目录

1、绪论

1.1 实习题目

基于 DSP 的交通灯设计。

1.2 课题研究的目的

1、熟悉使用 SEED-DEC5502 板控制 SEED-DEC—IO 上交通灯的方法；

2、掌握 DSP 扩展数字 I/O 接口的方法；

3、了解 SEED-DEC5502 的硬件系统。

通过本实验，了解 DSP 对 I/O 口的操作，实现交通灯模块的使用，并跟好的配置出各种模块的工作模式。

1.3 DSP 简介

数字信号处理（Digital Signal Processing，简称 DSP）是一门涉及许多学科而又广泛应用于许多领域的新兴学科。20 世纪 60 年代以来，随着计算机和信息技术的飞速发展，数字信号处理技术应运而生并得到迅速的发展。在过去的二十多年时间里，数字信号处理已经在通信等领域得到极为广泛 DSP 技术图解的应用。数字信号处理是利用计算机或专用处理设备，以数字形式对信号进行采集、变换、滤波、估值、增强、压缩、识别等处理，以得到符合人们需要的信号形式。

2、软件开发环境

CCS（Code Composer Studio）是一种针对 TMS320 系列 DSP 的集成开发环境，在 Windows 操作系统下，采用图形接口界面，提供有环境配置、源文件编辑、程序调试、跟踪和分析等工具。CCS 目前有三个不同时期的版本型号，又有针对 C2XX，CCS500,CCS6000 三个不同信号。CCS 有两种工作模式，即 软件仿真器模式：可以脱离 DSP 芯片，在 PC 机上模拟 DSP 的指令集和工作机制，主要用于前期算法实现和调试。

硬件在线编程模式：可以实时运行在 DSP 芯片上,与硬件开发板相结合在线编程和调试应用程序。

CCS 的开发系统主要由以下组件构成：

① TMS320C54x 集成代码产生工具；

② CCS 集成开发环境；

③ DSP/BIOS 实时内核插件及其应用程序接口 API；

④ 实时数据交换的 RTDX 插件以及相应的程序接口 API；

⑤ 由 TI 公司以外的第三方提供的各种应用模块插件。

CCS 的功能十分强大，它集成了代码的编辑、编译、链接和调试等诸多功能，而且支持 C/C++和汇编的混合编程，其主要功能如下：

① 具有集成可视化代码编辑界面，用户可通过其界面直接编写 C、汇编、.cmd 文件等；

② 含有集成代码生成工具，包括汇编器、优化 C 编译器、链接器等，将代码的编辑、编译、链接和调试等诸多功能集成到一个软件环境中；

③ 高性能编辑器支持汇编文件的动态语法加亮显示，使用户很容易阅读代码，发现语法错误；

④ 工程项目管理工具可对用户程序实行项目管理。在生成目标程序和程序库的过程中，建立不同程序的跟踪信息，通过跟踪信息对不同的程序进行分类管理；

⑤ 基本调试工具具有装入执行代码、查看寄存器、存储器、反汇编、变量窗口等功能，并支持 C 源代码级调试；

⑥ 断点工具，能在调试程序的过程中，完成硬件断点、软件断点和条件断点的设置；

⑦ 探测点工具，可用于算法的仿真，数据的实时监视等；

⑧ 分析工具，包括模拟器和仿真器分析，可用于模拟和监视硬件的功能、评价代码执行的时钟；

⑨ 数据的图形显示工具，可以将运算结果用图形显示,包括显示时域/频域波形、眼图、星座图、图像等，并能进行自动刷新；

⑩ 提供 GEL 工具。利用 GEL 扩展语言,用户可以编写自己的控制面板/菜单，设置 GEL 菜单选项，方便直观地修改变量，配置参数等；

3、系统方案设计

用 C 语言程序完成对交通灯的控制，实现相关控制。

DSP 系统中一般只有少量的数字 I/O 资源，而一些控制中经常需要大量的数字量的输入与输出。因而，在外部扩展 I/O 资源是非常有必要的。在扩展 I/O 资源时一般占用 DSP 的 I/O 空间。其实现方法一般有两种：其一为采用锁存器像 74LS273、741S373 之类的集成电路；另一种是采用 CPLD 在其内部做锁存逻辑，我们采用的是后者。

SEED-DEC5502 模板提供标准化的存储器扩展总线，以方便用户扩展其专用的电路。SEED-DEC5502 的存储器扩展总线，包含 3 个存储空间。子空间平均分成 3 块，分别分配为扩展总线的$\overline{\text{XCE1}}$、$\overline{\text{XCE2}}$、$\overline{\text{XCE3}}$，扩展总线的$\overline{\text{XCE1}}$、$\overline{\text{XCE2}}$、$\overline{\text{XCE3}}$可接口多种类型、多种数据宽度（8/16/32-位）的存储器，使用时，应根据它们接口的存储器类型和存储器数据宽度来动态调整 子空间的配置。接口不同存储器数据宽度时扩展总线的$\overline{\text{XCE1}}$、$\overline{\text{XCE2}}$、$\overline{\text{XCE3}}$在$\overline{\text{CE1}}$子空间的具体映射如下：

接口 8-位存储器：

	字节地址	字地址
$\overline{XCE1}$	0X4A0000~0X4BFFFF	0X250000~0X25FFFF
$\overline{XCE2}$	0X4C0000~0X4DFFFF	0X260000~0X26 FFFF
$\overline{XCE3}$	0X4E0000~0X4FFFFF	0X270000~0X27 FFF

接口 16-位存储器：

	字节地址	字地址
$\overline{XCE1}$	0X540000~0X57FFFF	0X280000~0X2BFFFF
$\overline{XCE2}$	0X580000~0X5BFFFF	0X2C0000~0X2D FFFF
$\overline{XCE3}$	0X5C0000~0X5FFFFF	0X2E0000~0X2F FFF

接口 32-位存储器：

	字节地址	字地址
$\overline{XCE1}$	0X680000~0X6FFFFF	0X340000~0X2BFFFF
$\overline{XCE2}$	0X700000~0X77FFFF	0X380000~0X2D FFFF
$\overline{XCE3}$	0X780000~0X7FFFFF	0X3C0000~0X2F FFF

扩展总线的$\overline{XCE3}$还可映射到’VC5502 的$\overline{CE3}$子空间，扩展总线的$\overline{XCE3}$映射的切换有系统控制寄存器 1（SYSCNTL1）中的 XCE3SEL 控制位决定。

当 XCE3SEL = 0 时，$\overline{CE3}$分配给板上 SDRAM 使用；当 XCE3SEL = 1 时，$\overline{CE3}$分配给扩展总线$\overline{XCE3}$使用，此时，应根据$\overline{XCE3}$接口的存储器类型来配置$\overline{CE3}$控制寄存器。

子空间的具体定位如下：

	字节地址	字地址
$\overline{CE3}$	0XC00000~0XFFFFFC	0X600000~0X7FFFFE

上电复位后，XCE3SEL = 0，$\overline{CE3}$缺省配置给 SDRAM 使用。在 EPD 实验箱中我们将 I/O 板映射到 SEED-DEC5502 模板的$\overline{CE3}$空间，地址映射关系如下：

实验箱 I/O 板对应的起始地址为：0x600000（字地址）；

TRAFFIC LED 的偏移地址为：0x000000；即 TRAFFIC LED 的地址为：0x600000；

SEED-DTK5502 系统中数字 IO 所占资源如下：

D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
SR	SY	SG	WR	EG	EY	WY	ER	WG	NR	NY	NG

NG：　方向北的绿灯控制位；

NY：　方向北的黄灯控制位；

NR：　方向北的红灯控制位；

WG：　方向西的绿灯控制位；

ER：　方向东的红灯控制位；

WY：　方向西的黄灯控制位；

EY：　方向东的黄灯控制位；

EG：　方向东的绿灯控制位；

WR：　方向西的红灯控制位；

SG：　方向南的绿灯控制位；

SY：　方向南的黄灯控制位；

SR：　方向南的红灯控制位；

当以上各位置“1”时，点亮各控制位所代表的交通灯状态的 LED 灯。

具体系统设计框图如图 3.1 所示。

图3.1系统设计框图

4、实验具体步骤和实现

4.1 CCS 软件的工程开发流程

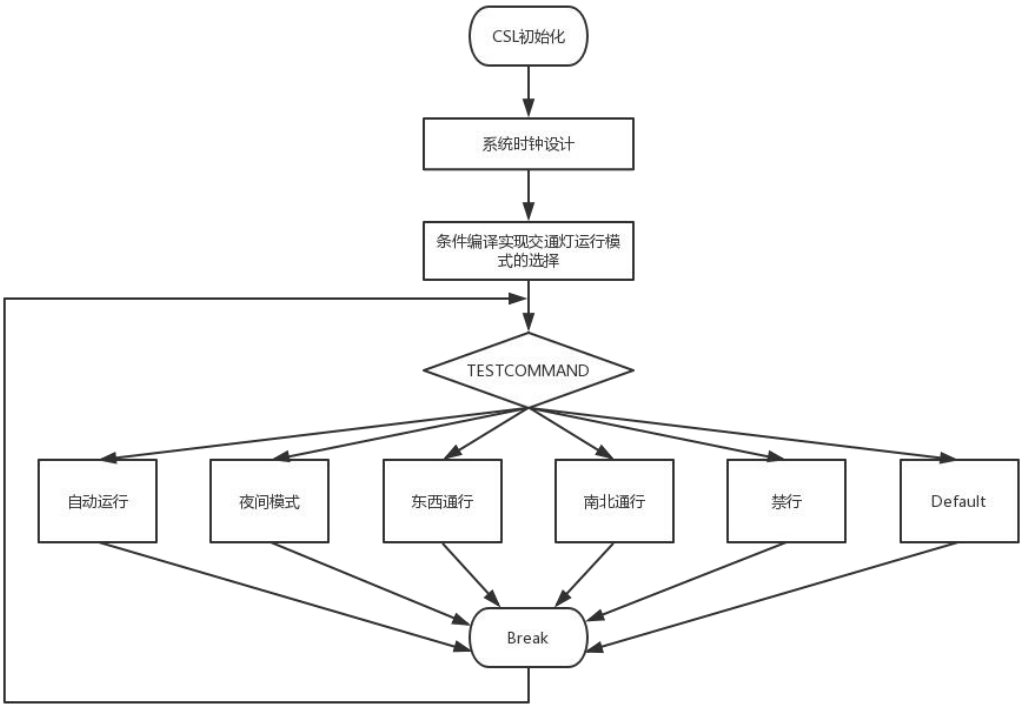


图 4.1.1 程序设计流程图

CSL 初始化以后，设计系统时钟，定义交通灯的具体闪烁方式如下所示：

交通灯闪烁方式：

闪烁方式	具体代码定义
EASTEWEST（东西通行）	0x88c
SOUTHNORTH（南北通行）	0x311
IOCHANGE（黄灯亮）	0x462
ALLFORBIN（全部红灯亮）	0x914

通过修改 TESTCOMMAND，可以修改交通灯的运行方式，具体运行方式和交通灯闪烁情况对应如下表所示：

交通灯运行模式对应的闪烁情况：

TESTCOMMAND	交通灯运行情况
1（自动运行）	NR、WG、EG、SR 亮 2 秒； NY、WY、EY、SY 亮 1 秒； NG、ER、WR、SG 亮 10 秒； NY、WY、EY、SY 亮 1 秒； NR、WG、EG、SR 亮 2 秒；
2（夜间模式）	NY、WY、EY、SY 亮 1 秒； 全部灯灭 6 秒；
3（东西通行）	NY、WY、EY、SY 亮 1 秒； NR、WG、EG、SR 亮 2 秒；
4（南北通行）	NY、WY、EY、SY 亮 1 秒； NG、ER、WR、SG 亮 10 秒；
5（禁　　行）	NY、WY、EY、SY 亮 1 秒； NR、ER、WR、SR 亮 2 秒；

4.2 创建工程与调试工程文件

4.2.1 配置运行环境

1、安装 TDS510 驱动

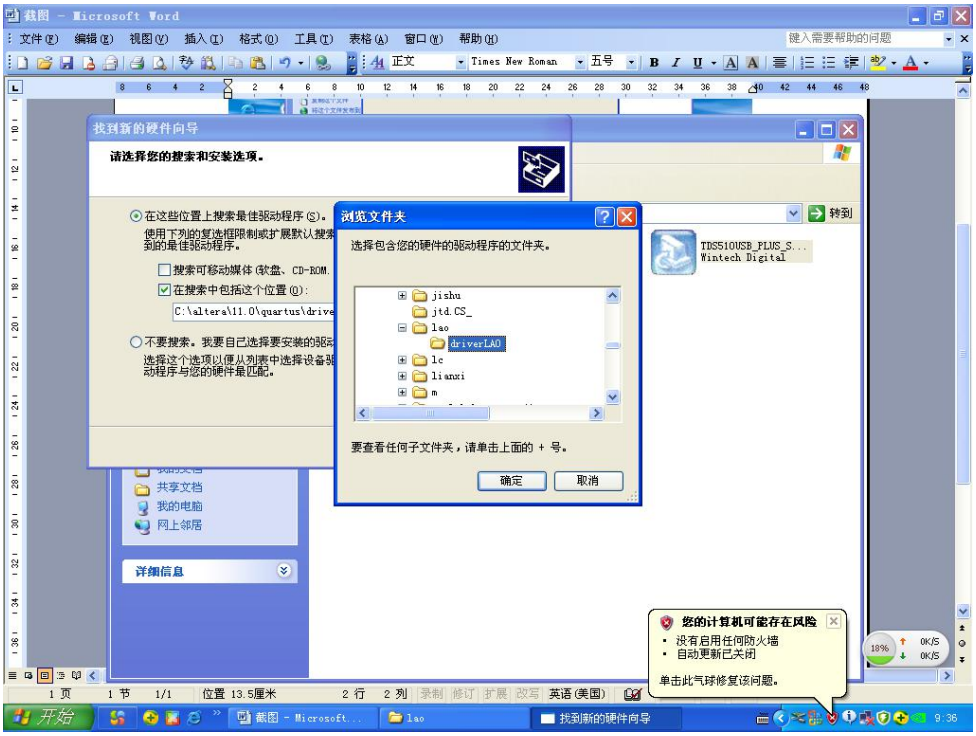


图 4.2.1.1 驱动安装图

2、配置 CCS

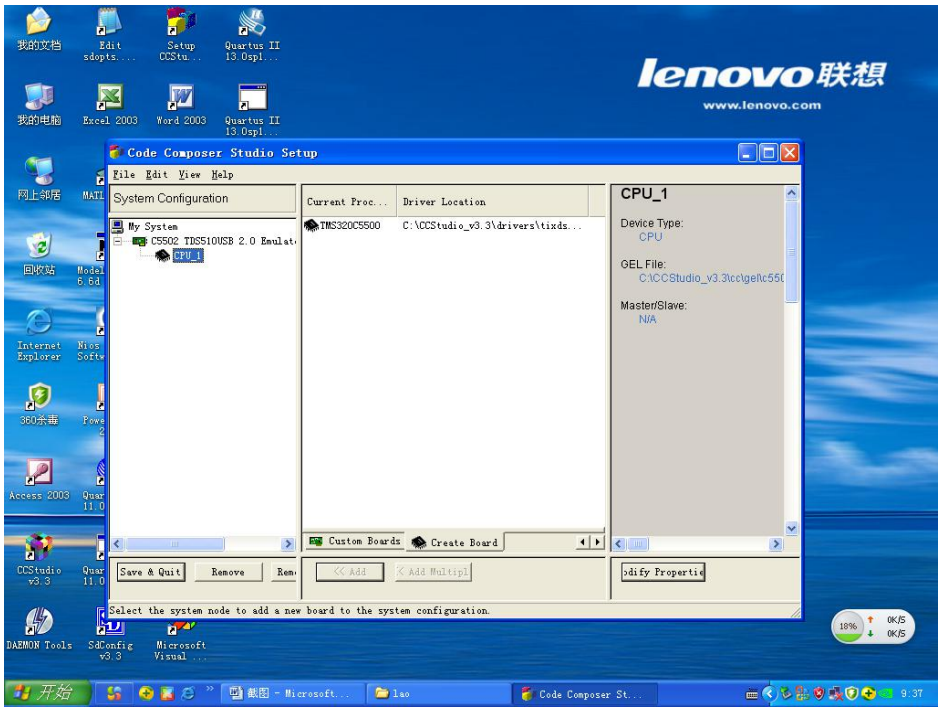


图 4.2.1.2 配置 CCS 图

3、连接实验箱

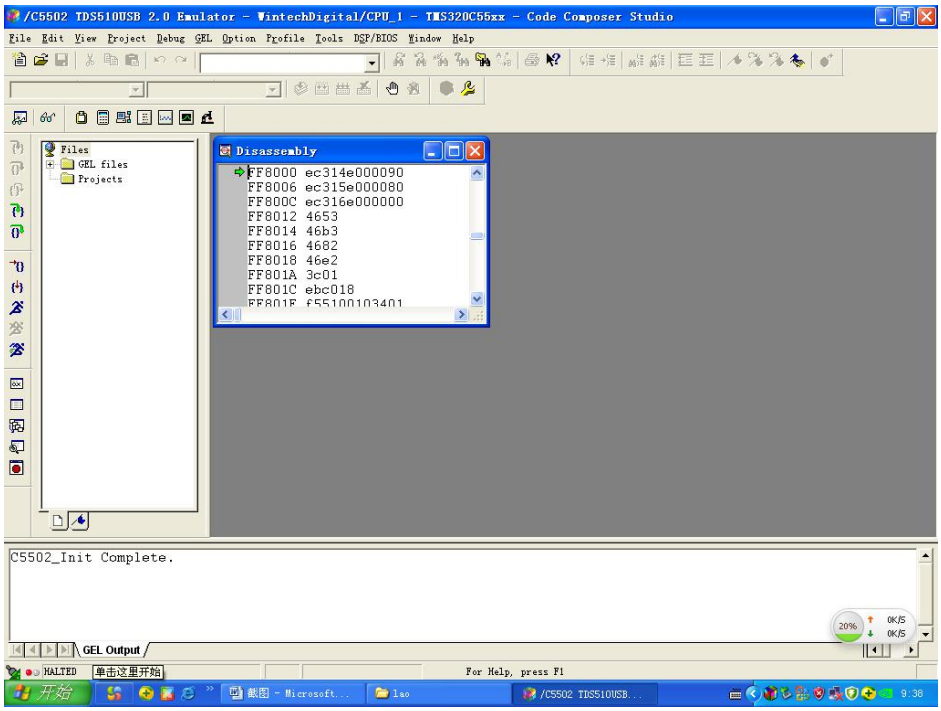


图 4.2.1.3 连接试验箱成功图

4.2.2 调试工程文件步骤

1、新建一个项目：点击 Project—New，将项目命名为 DEC5502_IO.pjt，并将项目保存在自己定义的文件夹下，注意文件夹一定要用英文名，不要将文件夹取名为中文名，因为 CCS 软件不能识别以中文命名的文件夹。

2、新建一个源文件：点击 File—New—Source File 可以打开一个文本编辑窗口，点击保存按键，保存在和项目相同的一个文件夹下面(DEC5502_IO)，保存类型选择*.C，我们在这里将保存名字命名为 AD7822.asm，同样建立并命名一个为 5502_IO.c。

3、在项目中添加源文件：在新建立了一个源文件以后，要想使用 CCS 编译器对该源文件进行编译还需要将源文件添加到项目中去。添加方法是在工程管理器中右键单击 traffic.pjt，在弹出的菜单中选择 Add Files，然后将刚才建立的 5502_IO.c 文件和 emif.c 添加到该项目中去。

程序运行过程如下：

1、开电，运行 CCS 软件，打开项目文件 DEC5502_IO.pjt，编译运行并下载。

2、设置断点，再点击“运行”按钮，如下图所示。这里的相关算法是在程序中编写的。

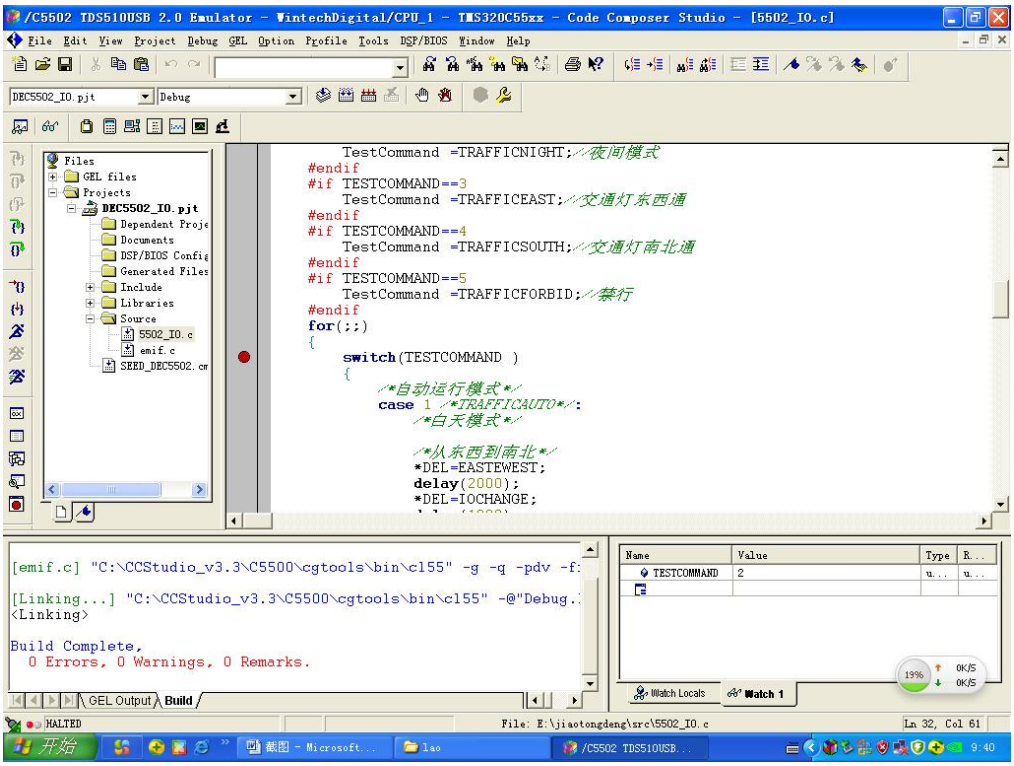


图 4.2.2.1 添加断点图

3 、点击 File—>Load progrram 打开图形对话框，如下。

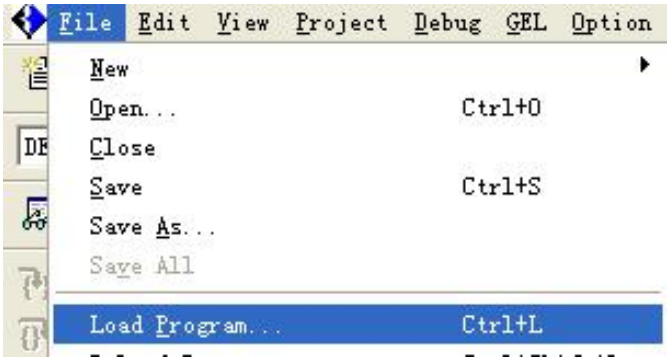


图 4.2.2.2 Load progrram

4、下一步

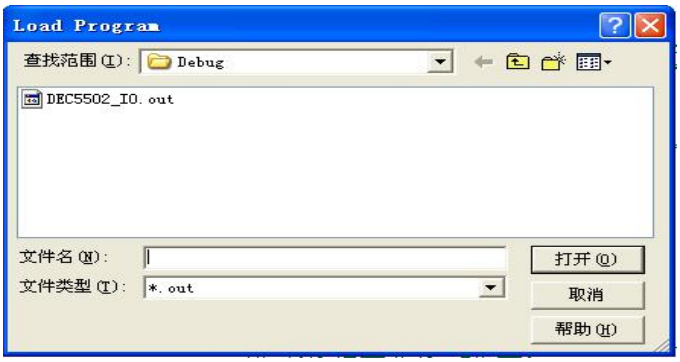


图 4.2.2.3 添加.out 文件

5、点击 View—>Watch Window 打开图形属性对话框，并修改 TESTCOMMAND

的值如下。

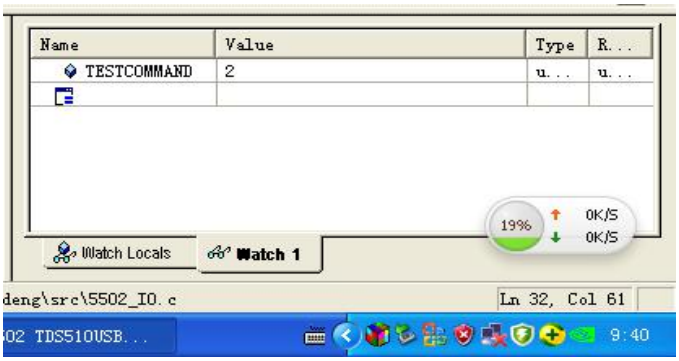


图 4.2.2.4 打开 Watch Window

程序分析：

5502_IO.c：这是实验的主程序，包含了系统初始化，并完成控制交通灯按照所选择的不同模式输出显示，以及 LED 灯按照可输入的 8 位二进制数显示结果；

Emif.c:包含 DSP 对 EMIF 外部接口的初始化；

SEED_DEC5502.cmd: 声明了系统的存储器配置与程序各段的连接关系。

本次实习有两种方案，C 语言和汇编语言实现，由于考虑到算法的特点，以及 C 语言的简练，我采用 C 语言版。

结论：

通过在 watch window 中修改 TESTCOMMAND 的值来实现对交通灯的控制，并根据不同的模式来切换闪烁时间、延迟时间或者控制那种灯亮等等。

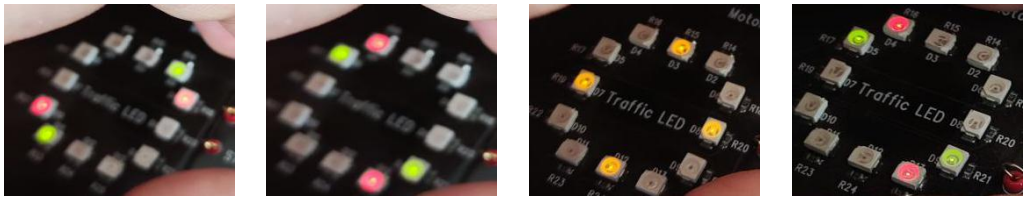


图 4.2.2.5 实验运行结果

5、实习心得

通过两周的 DSP 实习，使我收获到了特别多的知识与经验。作为一名大三的学生，我觉得能做类似的课程设计是十分有意义的。尤其是这次实习的模式基本为一人一组。此次实习正是因为我自己一人一组，没有其他同学的帮助，所以此次实习质量特别的高。本次实习我也是查遍了各个资料，就是为了能够更好的完成本次交通灯的任务。例如在做交通灯实习时我们用 5502 芯片设计时出现遇到了许多难题，不懂得 EMIF 接口设置问题和 EMIF 外部存储器接口究竟连接那些外部设备，经过老师的细心分析与讲解很快的为我解决了这些问题，我们熟知了 EMIF 是与 CPLD 连接以及交通灯。虽然本次实习只是完成了简单的交通灯，但是也有挺多的不足之处。没有能特别完美的完成任务。自己也会在实习后的课余时间补全自己的不足之处。这次实习不但让我了解到 DSP 课程的重要性而且还锻炼了我的自学能力和动手能力在以后的学习和生活中给予了很大的帮助，总的来说这次实习让我受益非浅。

6、教师评语

教师评语：

成绩：

教师签字：

附录 A 代码

CCS 程序：

交通灯模块程序：

```
#include <csl.h>
#include <csl_pll.h>
#include <csl_emif.h>
#include <csl_chip.h>
#include <stdio.h>
//实验操控：
//选择 TESTCOMMAND:1 为自动运行,2 为夜间模式,3 为交通灯东西通,
// 4 为交通灯南北通,5 为禁行。
unsigned int TESTCOMMAND =0 ;
//#define TESTCOMMAND   1  //交通灯操作命令选择
unsigned int TestCommand =0;//无操作
#define TRAFFICAUTO      0xAA14//自动运行模式
#define TRAFFICNIGHT 0xAA16//夜间模式
#define    TRAFFICEAST       0xAA1A//东西通行
#define TRAFFICSOUTH 0xAA1B//南北通行
#define TRAFFICFORBID    0xAA1C//禁行
#define  EASTEWEST     0x88c    //交通灯东西通（南北禁行）
#define  SOUTHNORTH    0x311    //交通灯南北通（东西禁行）
#define  IOCHANGE      0x462    //交通灯各方向黄灯亮
#define  ALLFORBIN     0x914    //交通灯各方向均禁行
void delay(int period);
volatile unsigned char* DEL = (volatile unsigned char *)0x600000;//交
通灯
volatile unsigned char* DECCTL = (volatile unsigned char *)0x280001;//
控制寄存器
main()
{
    /*初始化 CSL 库*/
    CSL_init();
    /*设置系统的运行速度为 300MHz*/
```

```
 PLL_setFreq(1, 0xF, 0, 1, 3, 3, 0);
/*初始化 DSP 外部 EMIF*/
 Emif_Config();
 *DECCTL=0x40;
/* #if TESTCOMMAND==1
    TestCommand =TRAFFICAUTO;//自动运行
#endif
#if TESTCOMMAND==2
    TestCommand =TRAFFICNIGHT;//夜间模式
#endif
#if TESTCOMMAND==3
    TestCommand =TRAFFICEAST;//交通灯东西通
#endif
#if TESTCOMMAND==4
    TestCommand =TRAFFICSOUTH;//交通灯南北通
#endif
#if TESTCOMMAND==5
    TestCommand =TRAFFICFORBID;//禁行
#endif    */
for(;;)
{
    switch(TESTCOMMAND)
    {
        /*自动运行模式*/
        case 0x01:
            /*白天模式*/
            /*从东西到南北*/
            *DEL=EASTEWEST;
            delay(2000);
            *DEL=IOCHANGE;
            delay(1000);
            *DEL=SOUTHNORTH;
            delay(8000);
            /*从南北到东西*/
```

```
    *DEL=SOUTHNORTH;
    delay(2000);
    *DEL=IOCHANGE;
    delay(1000);
    *DEL=EASTEWEST;
    delay(8000);
    break;
/*夜间模式*/
case 0x02:
    *DEL=IOCHANGE;
    delay(6000);
    *DEL=0;
    delay(6000);
    break;
/*交通灯东西通*/
case 0x03:
      *DEL=IOCHANGE;
    delay(1000);
    *DEL=EASTEWEST;
     delay(2000);
    break;
/*交通灯南北通*/
case 0x04:
      *DEL=IOCHANGE;
    delay(1000);
    *DEL=SOUTHNORTH;
    delay(2000);
    break;
/*禁行*/
case 0x05:
    *DEL=IOCHANGE;
    delay(1000);
    *DEL=ALLFORBIN;
    delay(2000);
```

```
                break;
            default:
                break;
        }
    }
}
void delay(int period)
{
    int i, j;

    for(i=0; i<period; i++)
    {
        for(j=0; j<0x1000; j++);
    }
}
```

EMIF 程序：

```
#include <csl.h>
#include <csl_emif.h>
/*FLASH 的 EMIF 设置*/
EMIF_Config MyEmifConfig = {
EMIF_GBLCTL1_RMK(               // EMIF Global Control Register 1
  EMIF_GBLCTL1_NOHOLD_HOLD_ENABLED,    // Hold enable
  EMIF_GBLCTL2_EK2HZ_HIGHZ,         //  EMIF_GBLCTL1_EK1HZ_EK1ENHigh-Z
control
  EMIF_GBLCTL1_EK1EN_ENABLED        // ECLKOUT1 Enable
  ),
EMIF_GBLCTL2_RMK(               // EMIF Global Control Register 2
  EMIF_GBLCTL2_EK2RATE_1XCLK,       // ECLKOUT2 Rate
  EMIF_GBLCTL2_EK2HZ_HIGHZ,         // EMIF_GBLCTL2_EK2HZ_EK2ENEK2HZ =
0, ECLKOUT2 is driven with value specified by EKnEN during
  EMIF_GBLCTL2_EK2EN_DISABLED       //  ECLKOUT2  Enable  (enabled  by
default)
  ),
EMIF_CE1CTL1_RMK(               // CE1 Space Control Register 1
```

```
  EMIF_CE1CTL1_TA_OF(3),        // Turn-Around time
  EMIF_CE1CTL1_READ_STROBE_OF(6),  // Read strobe width
  EMIF_CE1CTL1_MTYPE_16BIT_ASYNC,  // Access type
  EMIF_CE1CTL1_WRITE_HOLD_MSB_LOW, // Write hold width
bitHIGH
  EMIF_CE1CTL1_READ_HOLD_OF(3)     // Read hold width
  ),
EMIF_CE1CTL2_RMK(               // CE1 Space Control Register 2
  EMIF_CE1CTL2_WRITE_SETUP_OF(4),  // Write setup width
  EMIF_CE1CTL2_WRITE_STROBE_OF(10),    // Write strobe width
  EMIF_CE1CTL2_WRITE_HOLD_OF(2),   // Write hold width
  EMIF_CE1CTL2_READ_SETUP_OF(2)    // Read setup width
  ),
EMIF_CE0CTL1_RMK(               // CE0 Space Control Register 1
  EMIF_CE0CTL1_TA_DEFAULT,
  EMIF_CE0CTL1_READ_STROBE_DEFAULT,
  EMIF_CE0CTL1_MTYPE_DEFAULT,
  EMIF_CE0CTL1_WRITE_HOLD_MSB_DEFAULT,
  EMIF_CE0CTL1_READ_HOLD_DEFAULT
  ),
EMIF_CE0CTL2_RMK(               // CE0 Space Control Register 2
  EMIF_CE0CTL2_WRITE_SETUP_DEFAULT,
  EMIF_CE0CTL2_WRITE_STROBE_DEFAULT,
  EMIF_CE0CTL2_WRITE_HOLD_DEFAULT,
  EMIF_CE0CTL2_READ_SETUP_DEFAULT
  ),
EMIF_CE2CTL1_RMK(               // CE2 Space Control Register 1
  EMIF_CE2CTL1_TA_DEFAULT,         // Not use for SDRAM (asynchronous
memory types only)
  EMIF_CE2CTL1_READ_STROBE_DEFAULT,    // Read strobe width
  EMIF_CE2CTL1_MTYPE_32BIT_SDRAM,  // 32-bit-wide SDRAM
  EMIF_CE2CTL1_WRITE_HOLD_DEFAULT, // Write hold width
  EMIF_CE2CTL1_READ_HOLD_DEFAULT   // Read hold width
  ),
```

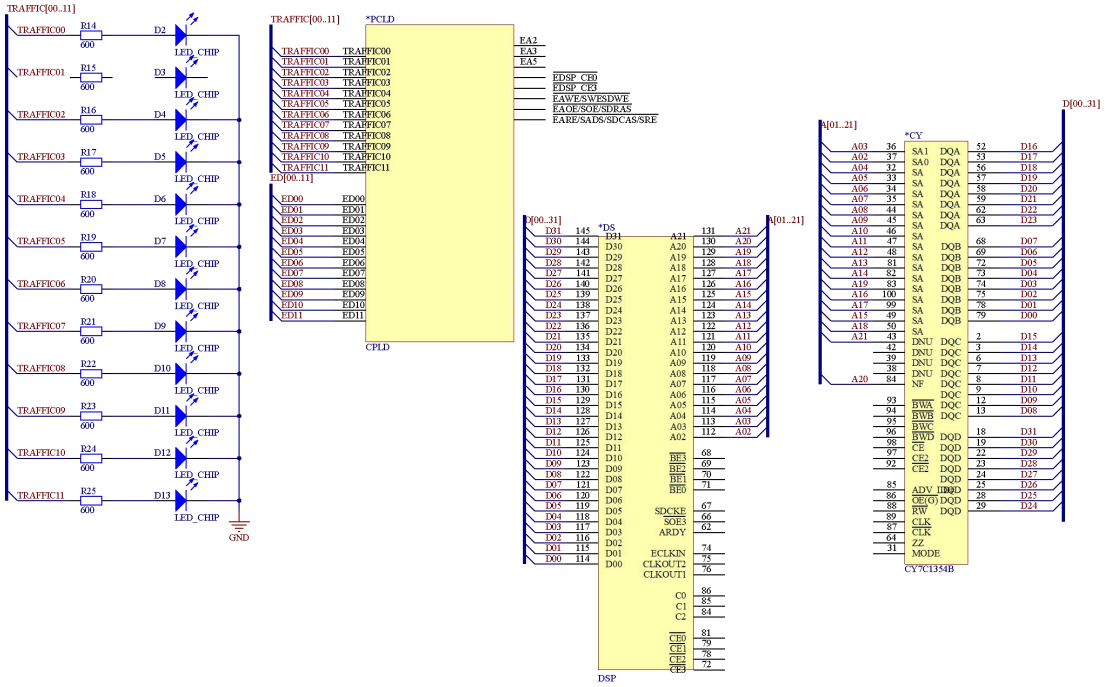
```
EMIF_CE2CTL2_RMK(              // CE2 Space Control Register 2
  EMIF_CE2CTL2_WRITE_SETUP_DEFAULT,    // Write setup width
  EMIF_CE2CTL2_WRITE_STROBE_DEFAULT,// Write strobe width
  EMIF_CE2CTL2_WRITE_HOLD_DEFAULT, // Write hold width
  EMIF_CE2CTL2_READ_SETUP_DEFAULT  // Read setup width
  ),
EMIF_CE3CTL1_RMK(              // CE3 Space Control Register 1
  EMIF_CE3CTL1_TA_OF(3),       // Turn-Around time
  EMIF_CE3CTL1_READ_STROBE_OF(6),  // Read strobe width
  EMIF_CE3CTL1_MTYPE_16BIT_ASYNC,  // Access type
  EMIF_CE3CTL1_WRITE_HOLD_MSB_LOW, // Write hold width MSB bitHIGH
  EMIF_CE3CTL1_READ_HOLD_OF(3)
  ),
EMIF_CE3CTL2_RMK(              // CE3 Space Control Register 2
  EMIF_CE3CTL2_WRITE_SETUP_OF(4),  // Write setup width
  EMIF_CE3CTL2_WRITE_STROBE_OF(10),    // Write strobe width
  EMIF_CE3CTL2_WRITE_HOLD_OF(2),   // Write hold width
  EMIF_CE3CTL2_READ_SETUP_OF(2)    // Read setup width
  ),
EMIF_SDCTL1_RMK(               // SDRAM Control Register 1
  EMIF_SDCTL1_TRC_OF(6),       // Specifies tRC value of the SDRAM in
EMIF clock cycles.
  EMIF_SDCTL1_SLFRFR_DISABLED      // Auto-refresh mode
  ),
EMIF_SDCTL2_RMK(               // SDRAM Control Register 2
  0x11,                            // 4 banks,11 row address, 8 column
address
  EMIF_SDCTL2_RFEN_ENABLED,        // Refresh enabled
  EMIF_SDCTL2_INIT_INIT_SDRAM,
  EMIF_SDCTL2_TRCD_OF(1),          // Specifies tRCD value of the SDRAM
in EMIF clock cycles
  EMIF_SDCTL2_TRP_OF(1)            // Specifies tRP value of the SDRAM
in EMIF clock cycles
  ),
```

```
0x61B,                          // SDRAM Refresh Control Register 1
0x0300,                             // SDRAM Refresh Control Register 2
EMIF_SDEXT1_RMK(                // SDRAM Extension Register 1
  EMIF_SDEXT1_R2WDQM_1CYCLE,
  EMIF_SDEXT1_RD2WR_3CYCLES,
  EMIF_SDEXT1_RD2DEAC_1CYCLE,
  EMIF_SDEXT1_RD2RD_1CYCLE,
  EMIF_SDEXT1_THZP_OF(1),           // tPROZ2=2
  EMIF_SDEXT1_TWR_OF(0),        //
  EMIF_SDEXT1_TRRD_2CYCLES,
  EMIF_SDEXT1_TRAS_OF(4),
  EMIF_SDEXT1_TCL_2CYCLES
  ),
EMIF_SDEXT2_RMK(                // SDRAM Extension Register 2
  EMIF_SDEXT2_WR2RD_0CYCLES,
  EMIF_SDEXT2_WR2DEAC_1CYCLE,
  0,
  EMIF_SDEXT2_R2WDQM_1CYCLE
  ),
EMIF_CE1SEC1_DEFAULT,               // CE1 Secondary Control Register 1
EMIF_CE0SEC1_DEFAULT,               // CE0 Secondary Control Register 1
EMIF_CE2SEC1_DEFAULT,               // CE2 Secondary Control Register 1
EMIF_CE3SEC1_DEFAULT,               // CE3 Secondary Control Register 1
EMIF_CESCR_DEFAULT                  // CE Size Control Register

  };
extern void Emif_Config(void)
{
    /*EMIF 为全 EMIF 接口*/
    CHIP_RSET(XBSR,0x0001);

    /*配置系统的 EMIF 接口*/
    EMIF_config(&MyEmifConfig);
}
```

附录 B 硬件电路图


TRAFFIC[00..11]
TRAFFIC00
TRAFFIC01
TRAFFIC02
TRAFFIC03
TRAFFIC04
TRAFFIC05
TRAFFIC06
TRAFFIC07
TRAFFIC08
TRAFFIC09
TRAFFIC10
TRAFFIC11
R14
R15
R16
R17
R18
R19
R20
R21
R22
R23
R24
R25
600
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
LED_CHIP
GND
*PCLD
ED[00..11]
ED00
ED01
ED02
ED03
ED04
ED05
ED06
ED07
ED08
ED09
ED10
ED11
EA2
EA3
EA5
EDSP_CE0
EDSP_CE1
EAWE/SWESDWE
EAOE/SOE/SDRAS
EARE/SADS/SDCAS/SRE
CPLD
D[00..31]
*DS
A[01..21]
BE3
BE2
BE1
BE0
SDCKE
SOE3
ARDY
ECLKIN
CLKOUT2
CLKOUT1
C0
C1
C2
CE0
CE1
CE2
CE3
DSP
*CY
BWA
BWB
BWC
BWD
CE
CE2
CE2
ADV_LD(W)
OE(G)
RW
CLK
CLK
ZZ
MODE
CY7C1354B