TF－VIA
32 位双 CPU 图形化开发嵌入式微控制器实验系统

实 验 指 导 书 v1.0

2012 年 10 月

目　录

注意事项

- 在移除或安装本实验箱内的各电路板或芯片前，必须断开实验箱电源，否则可能损坏芯片或元器件！

- 当使用 LM3S8962 核心板的虚拟串口对其供电时，务必关闭实验箱电源或断开 JP1 跳线，否则会造成芯片或其它元器件损坏！

- 当 LM3S8962 和 LM3S2110 两块核心板相互配合使用时，如使用 10 芯并口线（CAN 连接线）将它们连接时，必须断开 JP2 跳线，否则会造成芯片或其它元器件损坏！连接或断开操作必须在实验箱电源关闭的情况下操作。

- 当使用直流电机时，需注意死区保护，调试程序时，要注意 P1、P2、N3~N6 这几个器件，如发现这些器件中某个或某几个发热量较大，请立刻断开电源，查找并修改程序，待发热器件冷却之后再行调试，否则会造成器件损坏！

- 在直流电机工作前，请确定电机转盘上无障碍物或其它物品阻碍其正常转动，请勿用手或其它物品阻止其转动，否则可能造成器件损坏！

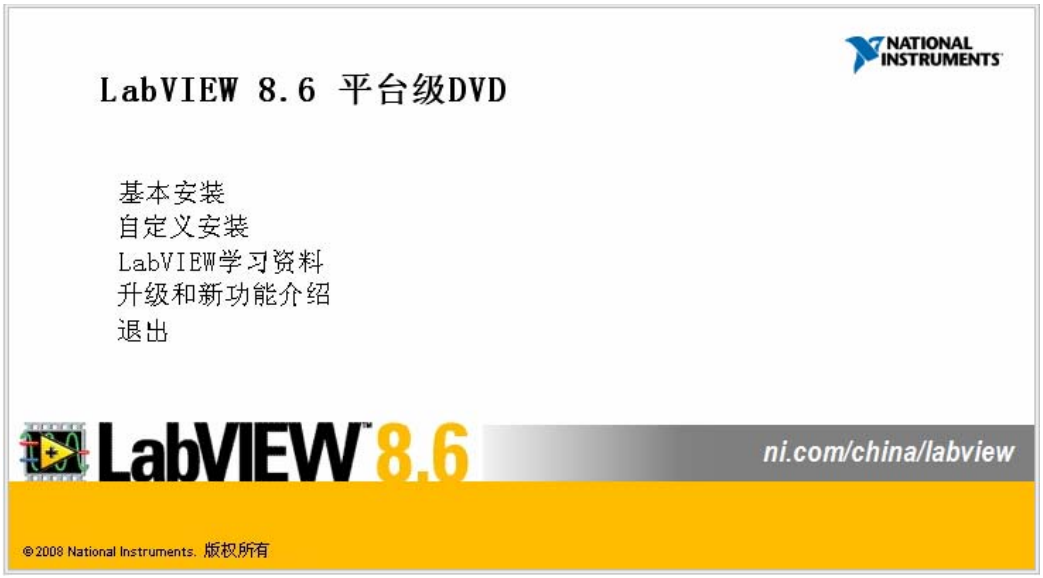
- 当使用实验箱上的串口 1 和串口 2 时，需选用**双端母头串口交叉线（2-3 交叉）**，否则可能无法观察到正确的结果，严重时可能导致器件损坏！

- 在 MDK 开发环境中打开实验程序时，因为软件的安装路径不同，可能会出现部分文件无法打开情况，这时只要重新指定文件路径即可。下面以 DriverLib.lib 文件为例，说明如何修正路径：在标有红色叉号的 DriverLib.lib 文件上右键单击，选择“Options for File ‘DriverLib.lib’”选项，这时会出现一个错误提示，“Could not successfully read archive ‘d:\Keil_v322\ARM\RV31\LIB\Luminary\DriverLib.lib’!”点击确定，该对框会再次出现，再次点击确定，直到出现标题为“Options for File ‘DriverLib.lib’”的设置页面。在该页面的“Path”选项里指定正确的文件路径，然后点击确定即可。

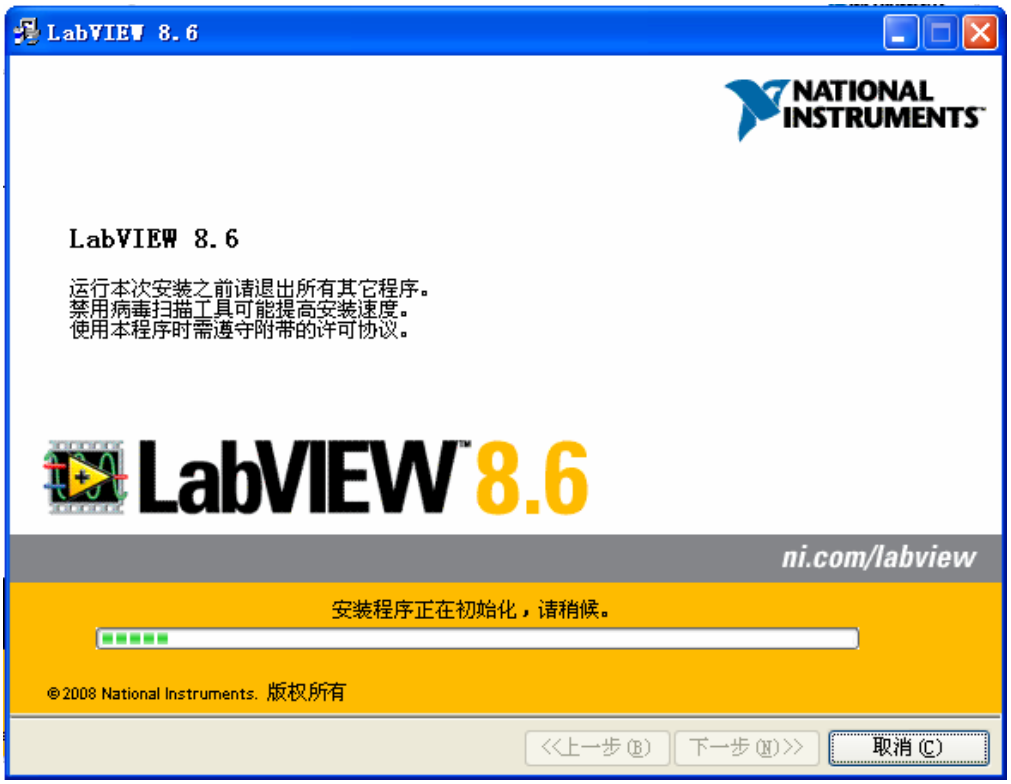
- 有关串口通信实验，传输波特率一般都设置为：115200bps；其余设置采用系统默认。

安装说明

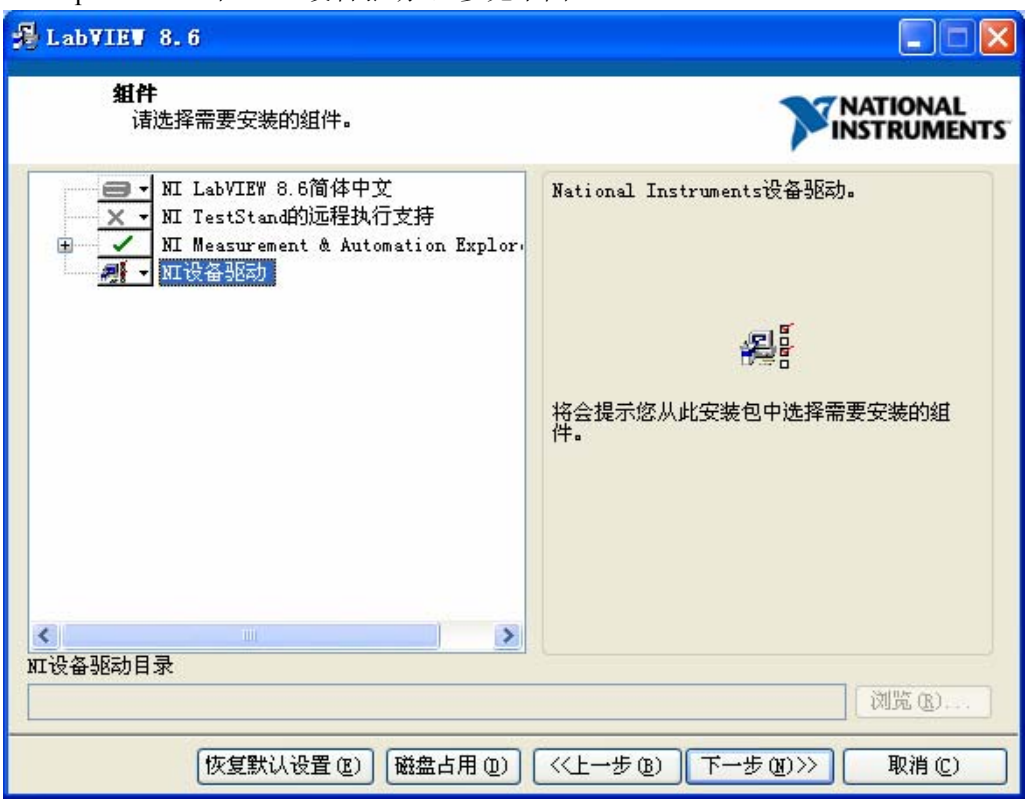
1、 将 LabVIEW8.6（或以上版本）安装光盘放入计算机光驱内，安装 LabVIEW 主程序，参见下图。



2、 点击“基本安装”(不同版本的光盘可能略有不同，如对 LabVIEW 的各组件比较熟悉，可根据自己的需求选择“自定义安装”)，进入安装界面，参见下图。



3、 待设定安装路径之后，来到安装组件选择，（不同版本光盘，其组件选项可能会不同），选择“LabVIEW 8.6 简体中文版”；选择“NI Measurement & Automation Explorer 4.5”和“NI 设备驱动”，参见下图。

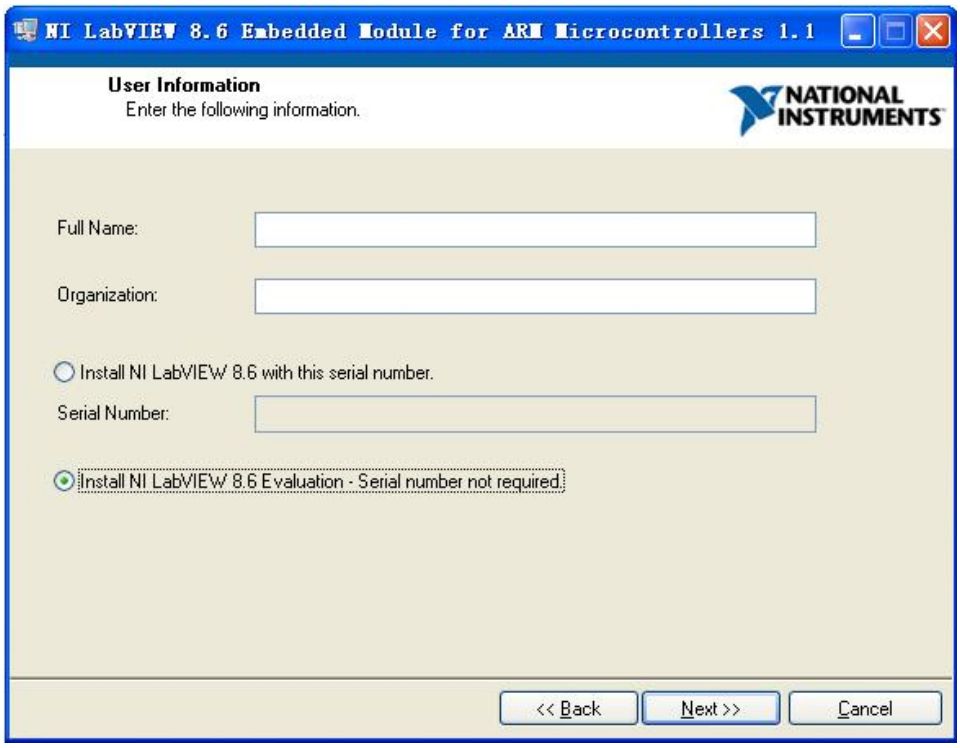


4、 安装完 LabVIEW 主程序后，继续安装 NI LabVIEW Embedded Module for ARM

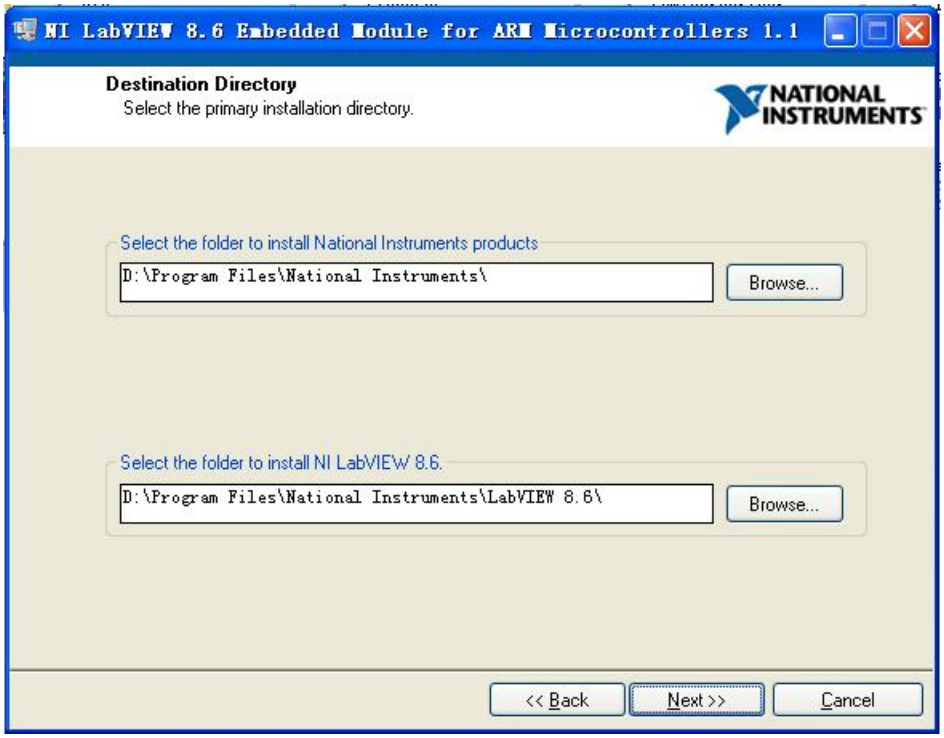
Microcontrollers 开发模块。在安装这个开发模块时，首先选择第一个选项 Install RealView Microcontroller Development Kit，安装 MDK 开发环境，参见下图。

5、 进入安装界面，如图 5，安装“RealView Microcontroller Development Kit”（MDK 开发环境），具体的安装步骤比较简单，这里就不再赘述，参见下图。

6、 安装 MDK 之后，回到步骤 4，选择第三个选项，进入用户信息界面，如用户姓名，单位等信息，这里先选择下面一个选项“Install NI LabVIEW 8.6 Evaluation – Serial number not required”，，参见下图。



7、 然后进入安装路径设定，这里的路径要设定为 LabVIEW 主程序同一路径，（建议采用系统默认路径），参见下图。



8、 待全部安装完成后，启动 LabVIEW 主程序，在主程序界面的左下方，有个选项“ARM project”点击开始即可新建一个工程，参见下图。至此开发软件安装过程

全部完成。


启动
文件(F) 操作(O) 工具(T) 帮助(H)
LabVIEW 8.6
专业版许可
文件
新建
VI
项目
基于模板的VI...
更多...
打开
浏览...
终端
ARM Project
开始
资源
LabVIEW新用户
LabVIEW入门指南
LabVIEW基础
LabVIEW文档指南
LabVIEW帮助
LabVIEW升级
自动整理程序框图
快速放置
多个对象的属性
新功能列表
网络资源
论坛
培训课程
LabVIEW Zone
范例
查找范例...

第一部分：MDK 环境实验

实验 1：LED 指示灯实验

一、 实验目的

- 了解实验箱的硬件环境；
- 初步了解如何使用 LM3S8962 核心板的 LED 指示灯。

二、 实验要求

编写一个 LED 指示灯闪烁程序。

三、 实验原理

将数据写到 GPIO PortF 输出端口。

四、 实验环境

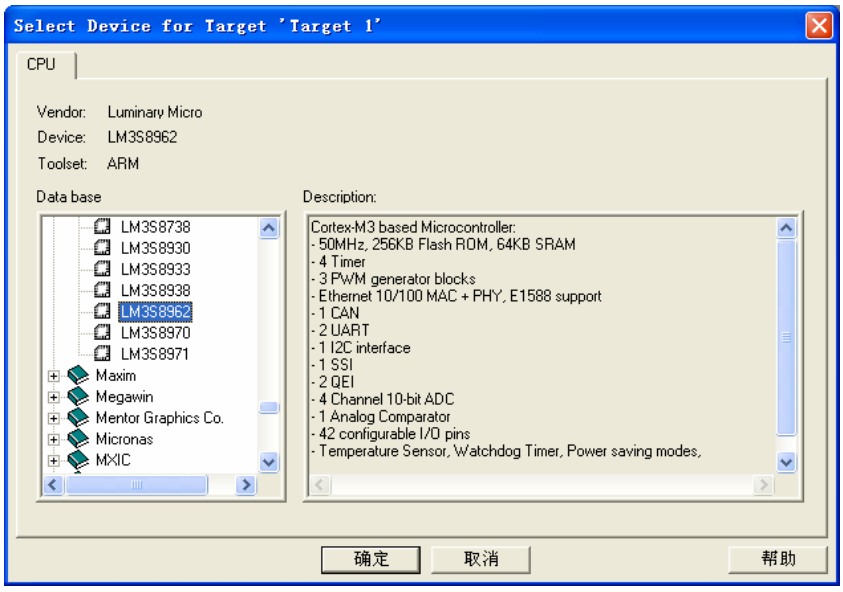
计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱。

五、 软件程序设计

根据实验要求，程序内容主要包括：定义 PortF.0 为输出端口，将数据写到 PortF.0 控制 LED 的亮灭。

整个工程包含 3 个源文件：Startup.s、DriverLib.lib 和 Blinky.c，其中 Startup.s 为启动代码；DriverLib.lib 文件是驱动库文件；Blinky.c 文件包含 main.c 函数，初始化 PortF.0 端口为输出端口，并写数据到 PortF.0 端口。具体程序清单见参考程序。

六、 建立工程



1、 创建工程并选择处理器。

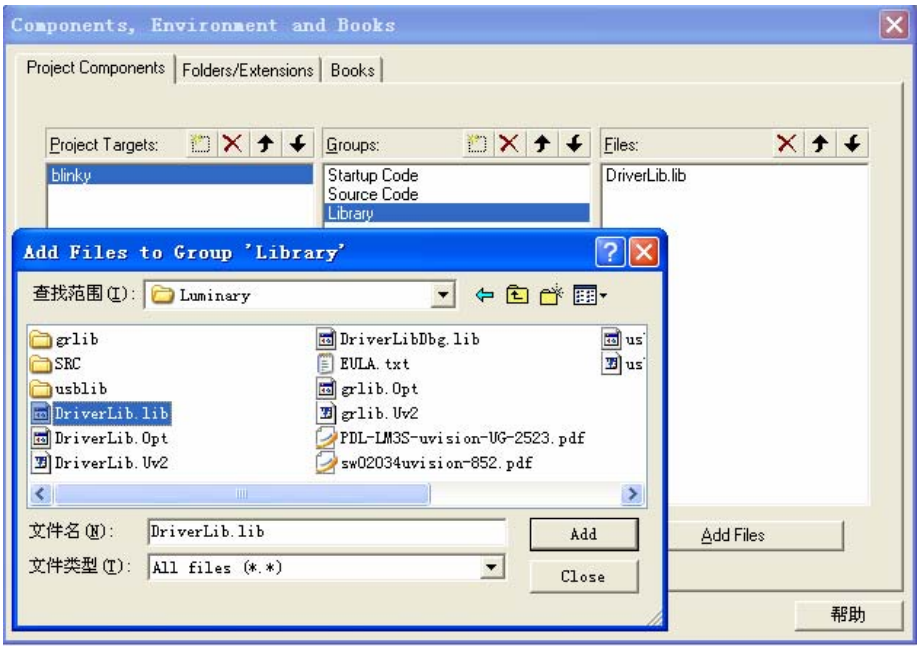
在 uVision 主界面在选择 **Project->New Project** 菜单项，打开一个标准对话框，输入 **Blinky**（建议对每个新建工程使用独立的文件夹，这里建立一个新的文件夹 **Keil01_Blinky**）。单击保存，出现选择处理器对话框：**Select Device for Target 'Target 1'**，选择 **Lunminary** 公司的 **LM3S8962** 控制器，如图所示。单击“**确定**”后，在弹出的对话框中单击“**是**”便可将启动代码加入工程。

2、 创建源文件及文件组

选择菜单项 **File->New**，根据程序清单创建新的源文件。在输入完源程序后，选择 **File->Save as** 菜单项保存源程序。

3、 加入驱动库文件

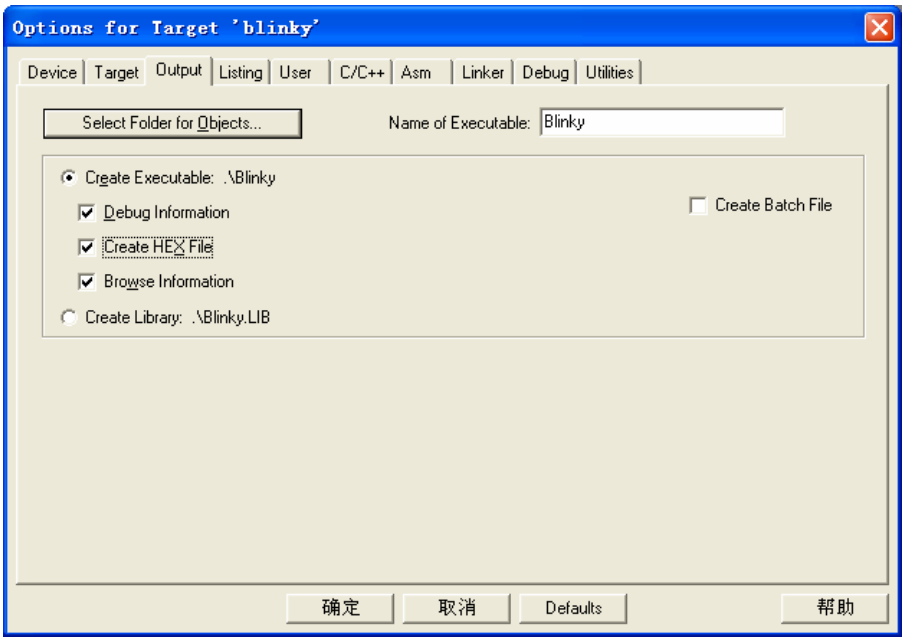
选择菜单项 **Project->Manage->Components，Environment，Books...** 在 **Project Components** 页中，可以创建文件夹，便于管理文件。在 **Groups** 区域中，点击刚创建的文件夹 **Library**，再在 **Files** 区域中点击 **Add Files**，在弹出的对话框中在，指定目录下（如 C:\Keil\ARM\RV31\LIB\Luminary）选择 **DriverLib.lib**，点击 **Add**，关闭对话框。如图所示：



4、 编译链接工程、调试程序

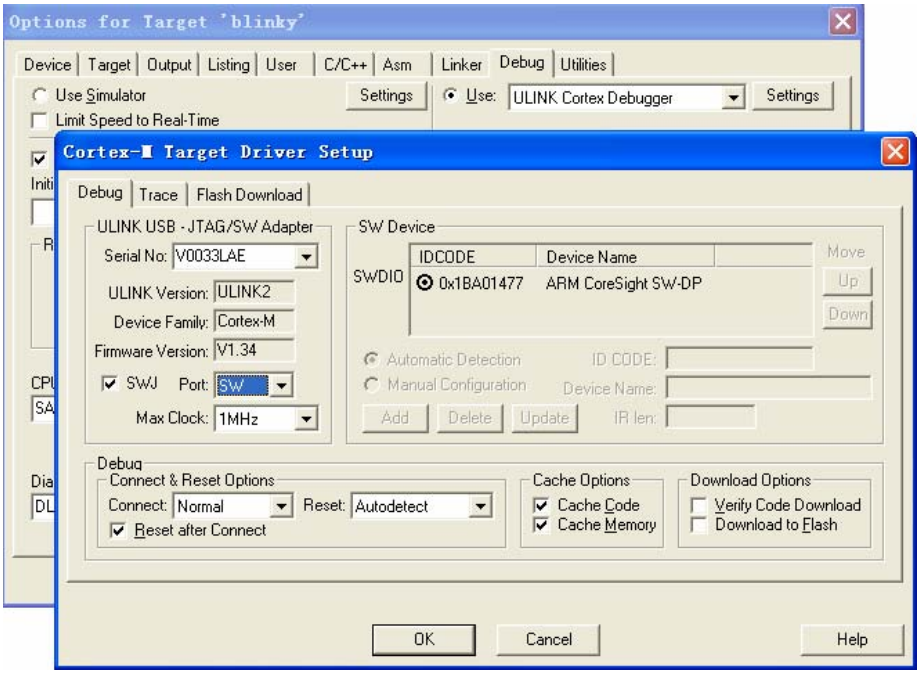
5、 建立 HEX 文件

应用程序在调试通过后，需要生成 Intel HEX 文件，用于下载到 E2PROM 编程器或仿真器中。在 **Options for Taget** 的 **Output** 页选择 **Create HEX File** 选项，如图所示：



6、 参数配置：

单击菜单栏中的 **Project->Options for Target 'blinky'**，在弹出的对话框中选择 **Debug** 页，在该页的右上角选择 **Use** 单选框，在该单选框右侧的下拉菜单中选择 **ULINK Cortex Debugger**；再点击 **Use** 单选框右侧的 **Settings** 按钮，在弹出的 **Cortex-M Target Driver Setup** 对话框中，在 **ULINK USB-JTAG/SW Adapter** 区域勾选 **SWJ** 项，在该项的 **Port** 中选择 **SW**，此时若在 **SW Sevice** 区域没有提示错误，则表示设备连接、配置正常。如下图所示：



七、 实验步骤与结果

1、 关闭实验箱电源，将仿真器(ULINK2)的数据接口排线插在核心板 1 的 JTAG 接口

上。

2、 打开 Keil01_Blinky 文件夹下的 Blinky.Uv2 工程文件，单击双箭头向下的全编译图标，完成程序的全编译并且生成可以下载烧写到 ARM 芯片的 AXF 文件，如果编译通不过或提示程序有错误，请修改程序错误，调试程序直到编译通过。

3、 全编译通过后，单击标有 load 的双箭头向下的图标，完成 AXF 文件通过仿真器下载和烧写到 ARM 芯片中,下载完成后按 S1 键(即 RESET 复位键)就可以在核心板上运行程序了。

4、 在 KEIL 界面下左端的 Project Workspace 窗口中，双击 Source Code 文件夹下的 ledblinkymain.c 程序文件，右端窗口中就会显示相应的程序内容，因为 main 函数就在其中，程序就是从 main 函数开始执行的，找到其中的 main 函数，可以查看程序的执行流程。

5、 在 KEIL 状态栏的 Debug 下，选择 Start/Stop Debug Session, 开始调试程序。

6、 程序正确运行后可见核心板上 LED 指示灯一亮一灭闪烁。

实验 2：OLED 字符显示实验

一、 实验目的

1、 了解实验箱的硬件环境；

2、 初步了解如何使用 LM3S8962 核心板的 OLED 显示器。

二、 实验要求

编写一个 OLED 显示程序，程序的功能就是在 OLED 指定位置上显示字符“Hello World!”。

三、 实验原理

将数据送到 OLED 中显示。

四、 实验环境

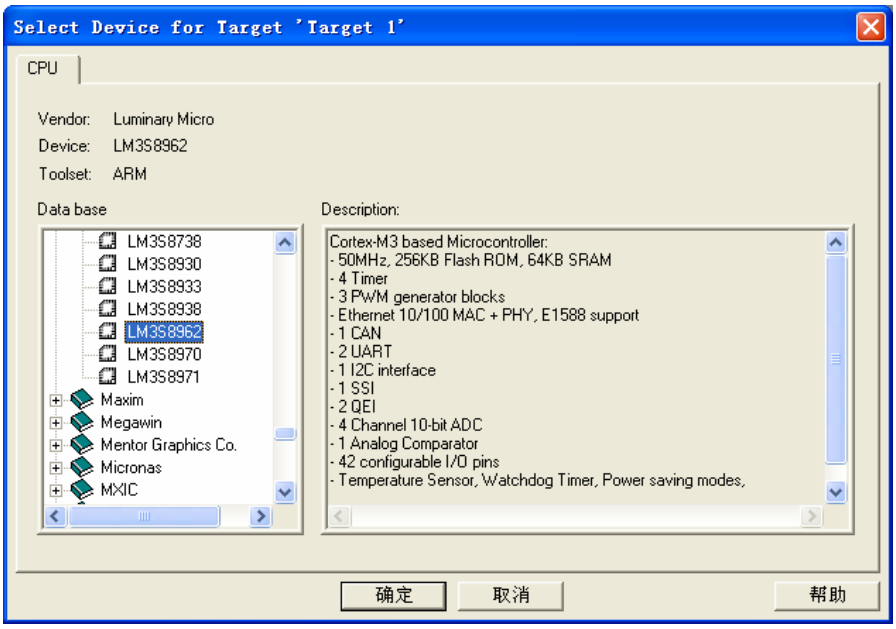
计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

根据实验要求，程序内容主要包括：初始化 OLED，使能 OLED，将文本字符串送到 OLED 显示器。

整个工程包含 5 个源文件：Startup.s、DriverLib.lib、rit128x96x4.c、uvision.c 和 hello.c，其中 Startup.s 为启动代码；DriverLib.lib 文件为驱动库；rit128x96x4.c 负责对 OLED 的操作，包括初始化、清屏、显示等功能；uvision.c 文件是对丢失调试器时的处理；hello.c 文件包含 main.c 函数。具体程序清单见参考程序。

六、 建立工程



1、 创建工程并选择处理器。

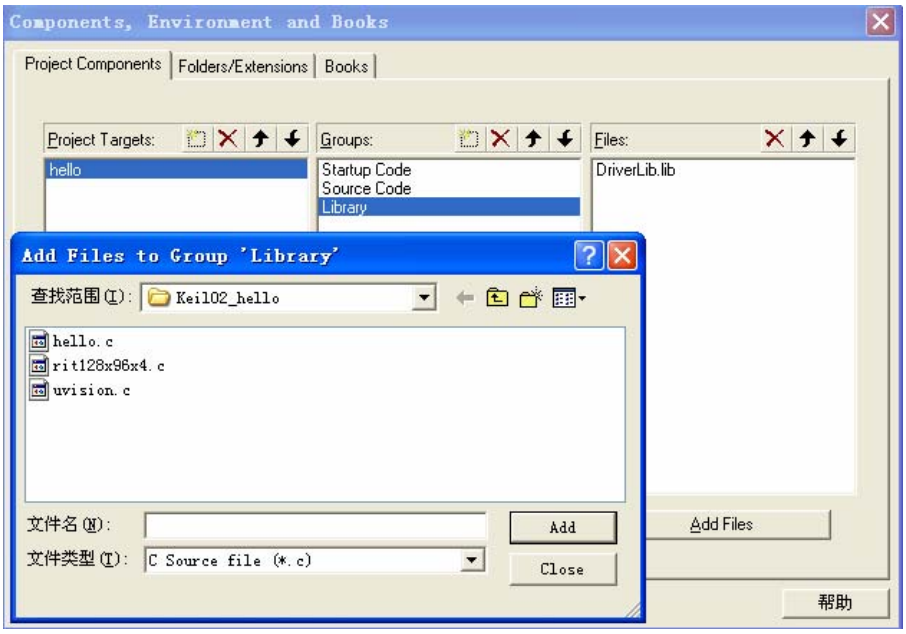
在 uVision 主界面在选择 **Project->New Project** 菜单项，打开一个标准对话框，输入 hello（建议对每个新建工程使用独立的文件夹，这里建立一个新的文件夹 Keil02_hello）。单击保存，出现选择处理器对话框：**Select Device for Target ‘Target 1’**，选择 **Lunimary** 公司的 LM3S8962 控制器，如图所示。单击“**确定**”后，在弹出的对话框中单击“**是**”便可将启动代码加入工程。

2、 创建源文件及文件组

选择菜单项 File->New，根据程序清单创建新的源文件。在输入完源程序后，选择 **File->Save as** 菜单项保存源程序。

3、 加入驱动库文件

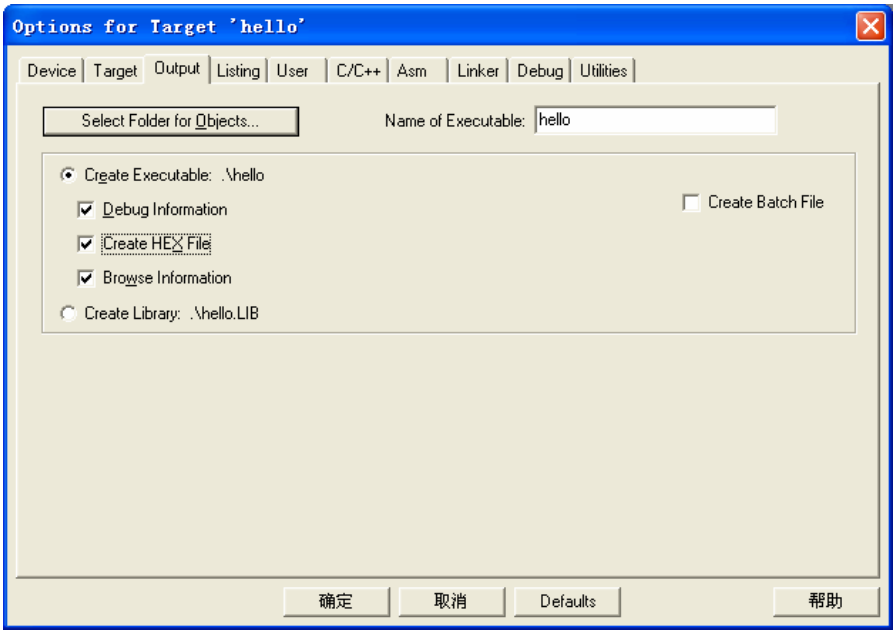
选择菜单项 **Project->Manage->Components**，Environment，Books...在 **Project Components** 页中，可以创建文件夹，便于管理文件。在 Groups 区域中，点击刚创建的文件夹 Library，再在 Files 区域中点击 Add Files，在弹出的对话框中在，指定目录下（如 C:\Keil\ARM\RV31\LIB\Luminary）选择 **DriverLib.lib**，点击 Add，关闭对话框。如图所示：



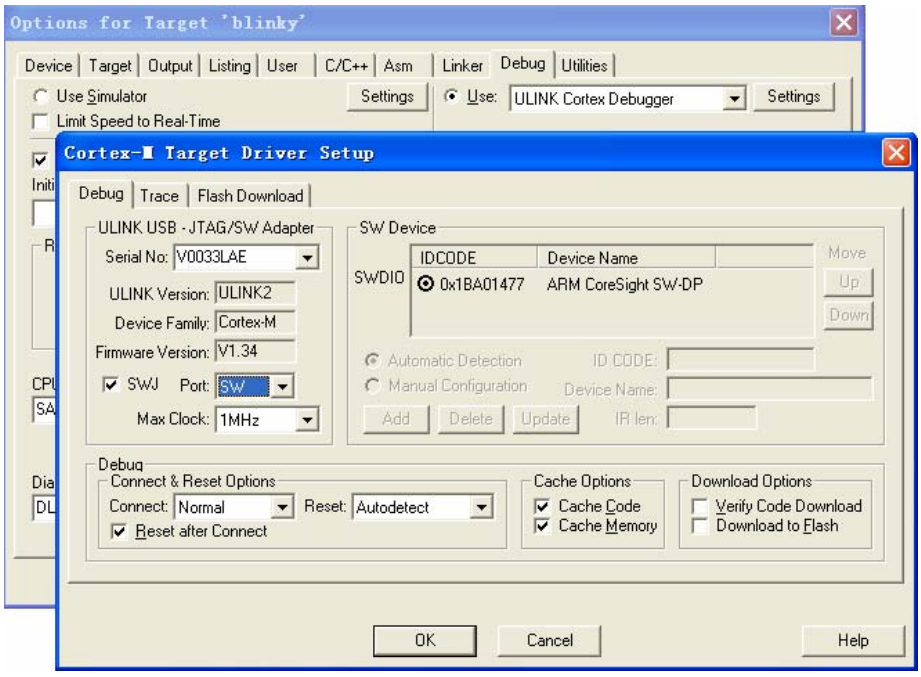
4、 编译链接工程、调试程序。

5、 建立 HEX 文件。

应用程序在调试通过后，需要生成 Intel HEX 文件，用于下载到 E2PROM 编程器或仿真器中。在 Options for Taget 的 Output 页选择 Create HEX File 选项，如图所示：



6、 参数配置：



单击菜单栏中的 Project->Options for Target ‘blinky’，在弹出的对话框中选择 Debug 页，在该页的右上角选择 Use 单选框，在该单选框右侧的下拉菜单中选择 ULINK Cortex Debugger；再点击 Use 单选框右侧的 Settings 按钮，在弹出的 Cortex-M Target Driver Setup 对话框中，在 ULINK USB-JTAG/SW Adapter 区域勾选 SWJ 项，在该项的 Port 中选择 SW，此时若在 SW Sevice 区域没有提示错误，则表示设备连接、配置正常。如下图所示：

七、 实验步骤与结果

1、 关闭实验箱电源，将仿真器(ULINK2)的数据接口排线插在核心板 1 的 JTAG 接口上。

2、 打开 Keil01_OLED 文件夹下的 lcedemo.Uv2 工程文件，单击双箭头向下的全编译图标，完成程序的全编译并且生成可以下载烧写到 ARM 芯片的 AXF 文件，如果编译通不过或提示程序有错误，请修改程序错误，调试程序直到编译通过。

3、 全编译通过后，单击标有 load 的双箭头向下的图标，完成 AXF 文件通过仿真器下载和烧写到 ARM 芯片中,下载完成后按 S1 键(即 RESET 复位键)就可以在核心板上运行程序了。

4、 在 KEIL 界面下左端的 Project Workspace 窗口中，双击 Source Code 文件夹下的 OLEDdemomain.c 程序文件，右端窗口中就会显示相应的程序内容，因为 main 函数就在其中，程序就是从 main 函数开始执行的，找到其中的 main 函数，可以查看程序的执行流程。

5、 在 KEIL 状态栏的 Debug 下，选择 Start/Stop Debug Session, 开始调试程序。

6、 程序正确运行后在 OLED 屏幕上显示“Hello World”字符。

实验 3：OLED 图形显示实验

一、 实验目的

1、 了解实验箱的硬件环境；

2、 学习 LM3S8962 核心板 OLED 显示器的使用。

二、 实验要求

编写一个图像显示程序,程序的功能就是在显示屏上显示 Luminary 公司的商业图标，并在图像上方显示滚动的字符串。

三、 实验原理

利用 OLED 显示器，根据 4 位位图格式文件，在 OLED 显示器中显示图像。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

整个工程包含 5 个源文件：Startup.c、DriverLib.lib、rit128x96x4.c、uvision.c 和 graphics.c，这里只介绍 graphics.c 源文件（其他的源文件在实验一和实验二中均已介绍）：包含 main.c 函数及 4 位位图格式文件中的部分数据。

六、 建立工程

参考实验一和实验二。

七、 实验步骤与结果

实验步骤参考实验一和实验二,程序正确运行后 OLED 屏幕上会出现动态显示的图标与文字。

八、 思考题

请尝试修改程序，使字符串（或其它点阵组成的图像）以垂直方向移动，或从屏幕的两边往中间集合。

实验 4：蜂鸣器实验

一、 实验目的

1、 了解实验箱的硬件环境；

2、 学习 LM3S8962 核心板 PWM 信号发生器的使用。

二、 实验要求

编写一个脉宽调制程序，程序的功能就是利用 PWM 发生器产生占空比 25%和 75%的 PWM 信号，输出到蜂鸣器中，发出蜂鸣声，并且输出过程是可以用按键控制的。

三、 实验原理

利用 PWM 发生器产生 PWM 信号激励蜂鸣器发出声音。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱。

五、 软件程序设计

整个工程包含 5 个源文件：Startup.c、DriverLib.lib、rit128x96x4.c、uvision.c 和 pwmgen.c，这里只介绍 pwmgen.c 源文件（其他的源文件在实验一和实验二中均已介绍）：包含 main.c 函数，完成对 PWM 发生器的端口设置、时钟设置、占空比设置，和使能 PWM 发生器。

六、 建立工程

参考实验一和实验二。

七、 实验步骤与结果

参考实验一和实验二；程序运行正确，蜂鸣器发出声响。

八、 思考题

请尝试修改程序，使 PWM 输出不同占空比的波形，看看声音有什么变化。

实验 5：内存检测实验

一、 实验目的

1、 了解实验箱的硬件环境；

2、 学习 LM3S8962 核心板存储器位段区的使用。

二、 实验要求

编写一个位操作程序,程序的功能就是对 SRAM 中的一个字节的 8 位进行位操作，使该字节为十六进制 0xdecafbad。并核对该字节的每一位是否正确。

三、 实验原理

访问 SRAM 中的位段。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱。

五、 软件程序设计

整个工程包含 5 个源文件：Startup.c、DriverLib.lib、rit128x96x4.c、uvision.c 和 bitband.c，这里只介绍 bitband.c 源文件（其他的源文件在实验一和实验二中均已介绍）：共有三个函数。

1） void Delay(unsigned long ulSeconds)：完成迟时一秒功能。

2） void PrintValue(unsigned long ulValue)：对 SRAM 位段操作，并显示位段中的一个字节。

3） main（）：主要是完成对处理器的初始化、系统定时器使能，并检验对位段位操作的结果，作出相应处理。

六、 建立工程

参考实验一和实验二。

七、 实验步骤与结果

参考实验一和实验二；内存检测正确，在 OLED 上显示字符“Success!”，否则显示“Errors!”。

实验 6：定时器中断实验

一、 实验目的

1、 了解实验箱的硬件环境；

2、 学习 LM3S8962 核心板定时器中断的使用。

二、 实验要求

编写定时器程序，程序的功能是定时器 0 每一秒产生一次中断，定时器 1，每 0.5 秒产生一次中断，在 OLED 上以 1 和 0 的跳变来表示中断过程。

三、 实验原理

利用定时器产生中断。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

整个工程包含 3 个源文件： DriverLib.lib、rit128x96x4.c 和 timers.c，这里只介绍 timers.c 源文件（其他的源文件在实验一和实验二中均已介绍）：主要说明以下 5 个函数。

1） Timer0IntHandler(void) ：定时间 0 中断处理程序。

2） SysCtlPeripheralEnable (SYSCTL_PERIPH_TIMER0)：使能定时器 0。

3） IntMasterEnable（）：使能处理器中断。

4） TimerConfigure (TIMER0_BASE, TIMER_CFG_32_BIT_PER)：定时器配置这个函数执行定时器模块的高级设置；也就是 说，它用来设置 32 或 16 位模式。

5） main（）：主要是完成对处理器的初始化、系统定时器使能，，作出相应处理。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

参考实验一和实验二；程序运行正确，我们可以在 OLED 屏幕上看到定时器 0 与 1 的中断次数。

八、 实验结果

程序运行时，在 OLED 上显示定时器 T1，T2 的中断，每次中断时，对其后面的布尔变量取反，并显示当前 T1 和 T2 中断次数，同时 LM3S8962 板上的 LED1 也与 T2 同步取反。

实验 7：看门狗定时器实验

一、 实验目的

1、 了解实验箱的硬件环境；

2、 学习看门狗使用。

二、 实验要求

编写一个看门狗程序，程序的是看门狗的简单使用。如果看门狗不定时喂食，它会重置系统。每次喂食的看门狗，用 OLED 的方式使我们看到，它正在喂，同时 LM3S8962 板上的用户 LED 也会同时闪烁。

三、 实验原理

在系统运行以后也就启动了看门狗的计数器，看门狗就开始自动计数，如果到了一定的时间还不去清看门狗，那么看门狗计数器就会溢出从而引起看门狗中断，造成系统复位。所以在使用有看门狗的芯片时要注意清看门狗。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

整个工程包含 3 个源文件： DriverLib.lib、rit128x96x4.c 和 wacthdog.c，这里只介绍 wacthdog.c 源文件（其他的源文件在实验一和实验二中均已介绍）：共有四个函数。

1） WatchdogIntHandler(void) ：为看门狗中断处理程序。

2） WatchdogReloadSet(WATCHDOG_BASE, SysCtlClockGet())：设置看门狗定时器的重装值

3） WatchdogResetEnable(WATCHDOG_BASE)：使能看门狗定时器的复位功能

4） main（）：主要是完成对处理器的初始化、使能看门狗，作出相应处理。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

参考实验一和实验二；

八、 实验结果

下载程序后复位 LM3S8962，OLED 屏幕上每隔 1 秒闪烁一次，闪烁的过程就就是看门狗复位的过程。

实验 8：直流电机实验

注意：本实验调试程序时需注意死区保护问题，注意 P1、P2、N3~N6 这几个器件，如发现这些器件中某个或某几个发热量较大，请立刻断开电源，查找并修改程序，待发热器件冷却之后再行调试，如若不然可能造成器件损坏。

一、 实验目的

1、 了解实验箱的硬件环境；

2、 学习直流电机的使用。

二、 实验要求

编写一个直流电机操作程序，让直流电机可以在正反向、快慢速，四种模式下进行切换，并注意观察直流电机的运行状态，并分析它和步进电机的工作状态有何不同，和各自的优缺点。

三、 实验原理

直流电机的转速计算公式如下：n=(U-IR)/KΦ，其中 U 为电枢端电压，I 为电枢电流，R 为电枢电路总电阻，Φ 为每极磁通量，K 为电动机结构参数。可 以看出，转速和 U、I 有关，并且可控量只有这两个，我们可以通过调节这两个量来改变转速。我们知道，I 可以通过改变电压进行改变，而我们常提到的 PWM 控 制也就是用来调节电压波形的常用方法，这里我们也就是用 PWM 控制来进行电机转速调节的。通过单片机输出一定频率的方波，方波的占空比大小绝对平均电压的大小，也决定了电机的转速大小。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

整个工程包含 4 个源文件： DriverLib.lib、rit128x96x4.c、Startup.s 和 dcmotor.c，这里只介绍 dcmotor.c 源文件（其他的源文件在之前的实验中均已介绍）。

在该程序中，请着重看电机控制部分，也就是 while(1)部分，重点分析该函数是如何控制直流电机的转速和方向的。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

本实验调试程序时需注意死区保护问题，注意 P1、P2、N3~N6 这几个器件，如发现这些器件中某个或某几个发热量较大，请立刻断开电源，查找并修改程序，待发热器件冷却之后再行调试，如若不然可能造

成器件损坏。

其它实验步骤可参考实验一、实验二和实验九。

八、 实验结果

下载程序后复位 LM3S8962，按下键盘“1”直流电机以顺时针方向慢速转动；按下“2”键以顺时针方向快速转动；按下“3”键以逆时针方向慢速转动；按下“4”键以逆时针方向快速转动；按下“0”键直流电机停止转动

九、 思考题

请仔细观察直流电机的工作原理，分析它和步进电机工作方式区别，并思考它们各自的优缺点和应用。

实验 9：小键盘实验 1

一、 实验目的

1、 了解实验箱的硬件环境；

2、 学习小键盘和七段数码管的使用。

二、 实验要求

编写一个键盘键入程序，程序的是当按下键盘的一个按键时，在 OLED 上显示该键的值，并同时在七段数码管上显示。

三、 实验原理

在键盘中按键数量较多时，为了减少 I/O 口的占用，通常将按键排列成矩阵形式，如图 1 所示。在矩阵式键盘中，每条水平线和垂直线在交叉处不直接连通，而是 通过一个按键加以连接。这样，一个端口（如 P1 口）就可以构成 4*4=16 个按键，比之直接将端口线用于键盘多出了一倍，而且线数越多，区别越明显，比如 再多加一条线就可以构成 20 键的键盘，而直接用端口线则只能多出一键（9 键）。由此可见，在需要的键数比较多时，采用矩阵法来做键盘是合理的。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

整个工程包含 3 个源文件： DriverLib.lib、rit128x96x4.c 和 sseg_kb.c，这里只介绍 sseg_kb.c 源文件（其他的源文件在实验一和实验二中均已介绍）：主要介绍三个函数。

1） disp(char x)：为七段数码管的显示处理程序。

2） GPIO_PORT_D_ISR(void)：中断服务函数 ISR。

3） main（）：主要是完成对处理器的初始化、使能 GPIO 外设，作出相应处理。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

参考实验一和实验二。

八、 实验结果

下载程序后复位 LM3S8962，按下小键盘上的按键，在 OLED 屏幕上显示当前按键，同时在七段数码管上显示该按键。

实验 10：小键盘实验 2

一、 实验目的

1、 了解实验箱的硬件环境；
2、 熟悉小键盘的使用。
3、 学习 LED 的使用。

二、 实验要求

编写一个键盘键入程序，程序的是当按下键盘的一个按键时，在 OLED 上显示该键的值，并同时在 LED 灯上以二进制的形式显示。

三、 实验原理

利用 LED 做二进制显示，左边为高位，右边为低位。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

整个工程包含 5 个源文件： DriverLib.lib、rit128x96x4.c 和 binary.c，这里只介绍 binary.c 源文件（其他的源文件在实验一和实验二中均已介绍）。

该实验中，可对比实验 12 来学习，主要对比下，GPIO_PORT_D_ISR(void)中断服务函数 ISR 在两个实验中的区别，看看这两种方式是如何工作的，有和相同点和不同点。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

参考实验一、实验二和实验十二。

八、 实验结果

下载程序后复位 LM3S8962，点击矩阵键盘上的按键，在 OLED 上显示当前按下的按键字符，同时 LED0～LED3 以二进制的方式点亮(左为低位，右为高位)。如按下键盘上的“7”键，OLED 上显示“7”，同时 LED0～LED2 被点亮，二进制为“0111”。

实验 11：ADC 实验

一、 实验目的

1、 了解实验箱的硬件环境；

2、 学习 UA741CN 的使用。

二、 实验要求

编写一个程序，程序的是当实验板上的电位器旋转时，在 OLED 屏幕上实时的显示当前的电压值。

三、 实验原理

ADC 模拟/数字转换过程主要有两项，首先是要对要转换的数据进行采样与保存（Sampling and Holding），然后再将取到的数据加以量化（Quantization），这样就完成了数据的转换。其中采样的目的在于将原始模拟数据取得其中若干个点，因此采样率（Sampling rate）越高则讯号越不易失真，也就是分辨率越高；量化的目的在于将由采样所获得的数据以 0 和 1 的组合进行编码，同样量化的位数越高则分辨率越高。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

整个工程包含 5 个源文件： DriverLib.lib、rit128x96x4.c、define.h、Startup.s 和 ADC.c，这里只介绍 ADC.c 源文件（其他的源文件在实验一和实验二中均已介绍）：主要介绍三个函数。

1） ADC_Sequence_0_ISR(void)：为 ADC 转换队列处理程序。

2） update() (void)：实时更新数据。

3） main（）：主要是完成对处理器的初始化、使能 ADC 模块，作出相应处理。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

参考实验一和实验二。

八、 实验结果

转动实验板上电位器，则在 LM3S8962 开发板的 OLED 上显示当前 ADC 转换后的电压值，范围在 0～3V 之间变化。

实验 12：优先级处理实验

一、 实验目的

1、 了解实验箱的硬件环境；

2、 熟练 LED 灯的使用

3、 了解优先级的处理方式。

二、 实验要求

编写一个优先级处理程序，由实验板上的拨动开关来充当中断请求，由实验板上的 LED 灯来充当优先级响应机制，来实现优先级处理。

三、 实验原理

多级中断的处理原则：当多级中断同时发生时，CPU 按照由高到低的顺序响应。高级中断可以打断低级中断处理程序的运行，转而执行高级中断处理程序。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

整个工程包含 5 个源文件： DriverLib.lib、Startup.s 和 i2c_trx.c，这里只介绍 i2c_trx.c 源文件（其他的源文件在实验一和实验二中均已介绍）。主要学习下面这几个函数。

1） UARTIntHandler(void)：串口中断处理程序。

2） update() (void)：实时更新数据。

3） main（）：主要是完成对处理器的初始化、使能 ADC 模块，作出相应处理。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

参考实验一和实验二。

八、 实验结果

下载程序后复位 LM3S8962，LED 灯从右向左依次为低级到高级变化，没有拨码开关被打开，全部 LED 灯都不亮，当只有一个拨码开关被打开，与其向对应的 LED 灯被点亮，若有多个拨码开关被打开，则只有被打开的拨码开关中最高级的 LED 灯被点亮，其它均被熄灭。

九、 思考题

请尝试修改程序，使该优先级处理方式改为右侧为最高优先级，左侧为最低优先级，并观察实验现象。

实验 13：串口传输实验 1

注意：请使用 2-3 交叉串口线连接实验箱串口 1 与 PC 机串口。

一、 实验目的

1、 了解实验箱的硬件环境；

2、 了解串口的工作方式。

二、 实验要求

编写两个程序，利用 LM3S8962 芯片控制串口 1，来向 PC 发送数据。发送波特率 115200，校验位 NONE，数据位 8，停止位 1；也可以轮询的方式来接收从 PC 发过来的字符，并将它显示在 OLED 上。

三、 实验原理

串口叫做串行接口，也称串行 通信接口 ，按电气标准及协议来分包括 RS-232-C、RS-422、RS485、USB 等。RS-232-C：也称标准串口，是目前最常用的一种串行通讯接口。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

该实验分为两个工程，这里主要介绍 LM3S8962 工程。包含 4 个源文件：DriverLib.lib、Startup.s、rit128x96x4.c 和 Serial_port.c，这里只介绍 Serial_port.c 源文件（其他的源文件在实验一、实验二和实验七中均已介绍）。主要学习以下几个函数。

1） UARTIntHandler(void)：串口中断处理程序。

2） 2）UARTSend(const unsigned char *pucBuffer, unsigned long ulCount)：发送 1 个字符到指定的 UART 端口（不等待）；ulBase：UART 端口的基址，取值 UART0_BASE、UART1_BASE 或 UART2_BASE ulData：要发送的字符。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

参考实验一、实验二和实验七。

八、 实验结果

下载程序之后，复位 LM3S8962，将看到串口的设置参数，这时按下小键盘上的“1”键，将进入发送模式，每隔一秒向外发送一个字符；按下“F”键将进入接收模式，可由 PC 端由串口工具之类的软件发送一个字节的字符，8962 将接收并显示。

实验 14：串口传输实验 2

注意：请使用 2-3 交叉串口线连接实验箱串口 1 与 PC 机串口；ULINK2 仿真器连接 LM3S2110（小板）的 JTAG 接口。

一、 实验目的

1、 了解实验箱的硬件环境；

2、 了解串口的工作方式。

二、 实验要求

编写两个程序，利用 LM3S2110 芯片控制串口 2，来向 PC 发送数据。发送波特率 115200，校验位 NONE，数据位 8，停止位 1。

三、 实验原理

同实验 9。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

该实验分为两个工程，这里主要介绍 LM3S8962 工程。包含 4 个源文件：DriverLib.lib、Startup.s、rit128x96x4.c 和 Serial_port.c，这里只介绍 Serial_port.c 源文件（其他的源文件在实验一、实验二和实验七中均已介绍）。主要学习以下几个函数。

1）UARTIntHandler(void)：串口中断处理程序。

2）UARTSend(const unsigned char *pucBuffer, unsigned long ulCount)：发送 1 个字符到指定的 UART 端口（不等待）；ulBase：UART 端口的基址，取值 UART0_BASE、UART1_BASE 或 UART2_BASE　　ulData：要发送的字符。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

参考实验一、实验二和实验七。

下载程序之后，将双母头串口线连接实验箱的串口 2 到 PC 机的串口，然后复位 LM3S2110，在 PC 端用串口查看工具来查看 LM3S2110 发送的数据。

八、 实验结果

开启实验箱电源，在 PC 机通过串口助手软件，可以接收到 LM3S2110 发送过来的字符，0～9；a～z；A～Z。

九、 思考题

在学习了两个芯片串口应用之后，请尝试修改程序，使 LM3S8962 在做接收端使用时，可一次接收多个字符，并将其显示在 OLED 上。

实验 15：秒表实验

一、 实验目的

1、 了解实验箱的硬件环境；

2、 熟悉七段数码管的工作方式。

二、 实验要求

编写一个程序，利用实验箱的 6 个七段数码管类组成分位、秒位和百分秒位。并有小键盘来控制计时开始、暂停和清零。

三、 实验原理

利用定时器来产生计数，选通相应的数码管来显示数值，通过循环的方式来更新每个数码管。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

该实验工程，包含 4 个源文件： DriverLib.lib、Startup.s、rit128x96x4.c 和 sseg_kb.c，这里只介绍 sseg_kb.c 源文件（其他的源文件在实验一、实验二和其他中均已介绍）。主要学习以下几个函数。

1） timerset()：定时器设置。

2） segdisp(char x)：七段数码管处理程序。

3） GPIO_PORT_D_ISR(void)：GPIO 的 D 口中断服务。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

参考实验一、实验二和实验十一。

八、 实验结果

开始七段数码管显示为 00.00.00，从左到右分别为分、秒和百分秒。按下启动键“1”，开始计时，计时期间按“1”或“2”键，可暂停计时，再次按下可恢复计时，按键“3”为清零键。

九、 思考题

请分析程序，然后修改，将计时秒表的功能该为正常的时钟（以两个七段数码管为一组，分别为小时、分钟和秒）。

实验 16：流水灯实验 1

注意：请将 ULINK2 仿真器连接到 8962 核心板的 JTAG 口上。

一、 实验目的

1、 了解实验箱的硬件环境；

2、 熟悉 LED 灯管的工作方式。

二、 实验要求

编写一个程序，利用实验箱的 8 个 LED 灯管，使其从左往右或从右往左逐个点亮，并熄灭之前的 LED。

三、 实验原理

利用二进制的位对应关系，对应实验箱上的 8 个 LED，利用某一位置 1，来点亮该位对应 LED，然后利用左移和右移来分别实现逐个点亮的效果。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

该实验工程，包含 4 个源文件： DriverLib.lib、Startup.s、rit128x96x4.c 和 RLED.c，这里只介绍 RLED.c 源文件（其他的源文件在实验一、实验二和其他中均已介绍）。主要学习以下几个函数。

1） SimpleDelay(void)：延时程序（具体时间可以自己掌握）。

2） mian()：主程序。

3） GPIO_PORT_D_ISR(void)：GPIO 的 D 口中断服务。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

参考实验一、实验二、实验十二和实验十四。

八、 实验结果

开启电源后 LED 灯将从右向左逐个点亮，并熄灭之前的 LED 灯，按键“1”为向左循环；按键“2”为向右循环。任何时候都可以改变起循环方向。

九、 思考题

分析程序然后修改程序，实现跳跃式点亮模式（由左向右或由右向左，每隔一个 LED 点亮一次），和相向相背点亮模式（由左右两个方向往中间逐个点亮，在中间相遇后由中间向两边逐个点亮）。并观察是实验现象。

实验 17：流水灯实验 2

注意：请将 ULINK2 仿真器连接到 2110 核心板的 JTAG 口上。

一、 实验目的

3、 了解实验箱的硬件环境；

4、 熟悉 LED 灯管的工作方式。

二、 实验要求

编写一个程序，利用实验箱的 8 个 LED 灯管，使其从左往右或从右往左逐个点亮，并熄灭之前的 LED。

三、 实验原理

利用二进制的位对应关系，对应实验箱上的 8 个 LED，利用某一位置 1，来点亮该位对应 LED，然后利用左移和右移来分别实现逐个点亮的效果。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱

五、 软件程序设计

该实验工程，包含 4 个源文件： DriverLib.lib、Startup.s、rit128x96x4.c 和 RLED.c，这里只介绍 RLED.c 源文件（其他的源文件在实验一、实验二和其他中均已介绍）。主要学习以下几个函数。

1） SimpleDelay(void)：延时程序（具体时间可以自己掌握）。

2） mian()：主程序。

3） GPIO_PORT_D_ISR(void)：GPIO 的 D 口中断服务。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

参考实验一、实验二、实验十二和实验十四。

八、 实验结果

开启电源后 LED 灯将从右向左逐个点亮，并熄灭之前的 LED 灯，按键“1”为向左循环；按键“2”为向右循环。任何时候都可以改变起循环方向。

九、 思考题

分析程序然后修改程序，实现跳跃式点亮模式（由左向右或由右向左，每隔一个 LED 点亮一次），和相向相背点亮模式（由左右两个方向往中间逐个点亮，在中间相遇后由中间向两边逐个点亮）。并观察是实验现象。

实验 18：CAN 传输实验

注意：进行该实验时，必须在关闭实验箱电源的状态下先断开 JP2 跳线，再用 10 芯 CAN 线与 LM3S8962 核心板相连且实验过程中 JP2 要保持断开状态，否则会损坏芯片或其它元器件。实验完毕后移除 CAN 连接线时同样要先关闭实验箱电源。

一、 实验目的

1、 了解实验箱的硬件环境；

2、 了解 CAN 线的传输模式。

二、 实验要求

编写两个程序，利用 LM3S8962 和 LM3S2110 的 CAN 单元来完成数据传递，并将数据显示在 OLED 屏幕上。

三、 实验原理

CAN 是控制器局域网络(Controller Area Network, CAN)的简称，是 ISO国际标准化的 串行通信协议，属于现场总线的范畴,它是一种有效支持分布式控制或实时控制的串行通信网络。CAN 控制器工作于多主方式,网络中的各节点都可根据总线访问优先权(取决于报文标识符)采用无损结构的逐位仲裁的方式竞争向总线发送数据,且 CAN 协议 废除了站地址编码,而代之以对通信数据进行编码,这可使不同的节点同时接收到相同的数据,这些特点使得 CAN 总线构成的网络各节点之间的数据通信实时性 强,并且容易构成冗余结构,提高系统的可靠性和系统的灵活性。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、LM3S8962 和 LM3S2110 核心板。

五、 软件程序设计

该实验有两个工程，主要介绍 LM3S8962 工程，包含 4 个源文件：DriverLib.lib、Startup.s、rit128x96x4.c 和 simp_rx.c，这里只介绍 simp_rx.c 源文件（其他的源文件在实验一、实验二和其他中均已介绍）。主要学习以下几个函数。

1） CANIntHandler(void)：CAN 中断服务处理。

2） mian()：主程序。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

注意：进行该实验时，必须在关闭实验箱电源的状态下先断开 JP2 跳线，再用

10 芯 CAN 线与 LM3S8962 核心板相连且实验过程中 JP2 要保持断开状态，否则会损坏芯片或其它元器件。实验完毕后移除 CAN 连接线时同样要先关闭实验箱电源。

将程序分别写入 LM3S8962 和 LM3S2110 两块核心板，然后分别按下两块核心板上的复位键，在实验箱上的 OLED 屏幕上可以看到 LM3S2110 芯片从 CAN 线上发来的数据。

八、 实验结果

将 LM3S8962 和 LM3S2110 两块开发板的 CAN 口，用 10 芯的并口线连接起来，再开启实验的电源，LM3S8962 开发板上 OLED 屏上将显示由 LM3S2110 通过 CAN 总线发送的 ASCII 字符。

九、 思考题

目前是 LM3S2110 核心板发送数据，LM3S8962 核心板接收；试试看能否改为 LM3S8962 核心板发送数据到 LM3S2110 核心板后再回传回来？

实验 20：传感器实验

一、 实验目的

1、 了解实验箱的硬件环境；

2、 熟悉传感器的使用。

二、 实验要求

编写一个程序，获取传感器的 Vout 端输出的数值，经转化之后，经 OLED 显示。

三、 实验原理

传感器就是能感知外界信息并能按一定规律将这些信息转换成可用信号的装置；简单说传感器是将外界信号转换为电信号的装置。所以它由敏感元器件（感知元件）和转换器件两部分组成，传感器就是能感知外界信息并能按一定规律将这些信息转换成可用信号的装置；简单说传感器是将外界信号转换为电信号的装置,通常是由敏感元器件（感知元件）和转换器件两部分组成。

四、 实验环境

计算机、MDK 集成开发环境、ULINK2 适配器、实验箱。

五、 软件程序设计

该实验工程，包含 4 个源文件： DriverLib.lib、Startup.S、define.h、和 ADC.c，这里只介绍 ADC.c 源文件（其他的源文件在实验一、实验二和其他中均已介绍）。主要逐一介绍以下几个函数。

4） ADC_Sequence_0_ISR(void)：ADC Sequence 0 的中断服务程序 。

5） update()：OLED 显示。

6） update1(float fdata2)：数值转化。

7） GPIO_PORT_D_ISR(void)：端口 D 的中断服务 。

六、 建立工程

参考实验一和实验二。

七、 实验步骤

将传感器模块的拨动开关拨到“TO ADC”端，下载、运行程序，完成实验。

八、 实验结果

从实验箱 OLED 屏幕上能看到音频传感器采集到的声波的波形及电压值。

九、 思考题

结合串口通信程序，可以试着将音频数据上传到 PC 中保存、显示、分析、回放。

实验 21：μCOSII 系统移植 1

一、 实验目的

1、 熟悉实验箱的硬件环境；
2、 熟悉实验系统各部分的使用。
3、 理解 LM3S8962 的各个资源使用。
4、 熟悉 μCOSII 系统的工作特点。
5、 了解 μCOSII 系统移植相关知识。

二、 实验要求

将 μCOSII 系统移植到 LM3S8962 芯片上，并且执行一个任务控制 LM3S8962 开发板上的 LED1 闪烁。

三、 实验原理

μCOSII 的简介

i. μCOSII 的组成部分

μCOSII 可以大致分成核心、任务管理、时间管理、任务同步与通信，CPU 的移植等 5 个部分。

ii. 核心部分

是操作系统的处理核心，包括操作系统初始化、操作系统运行、中断进出的前导、时钟节拍、任务调度、事件处理等多部分。

iii. 任务管理

μCOSII 中最多可以支持 64 个任务，分别对应优先级 0～63，其中 0 为最高优先级。63 为最低级，系统保留了 4 个最高优先级的任务和 4 个最低优先级的任务，所有用户可以使用的任务数有 56 个。μCOSII 提供了任务管理的各种函数调用，包括创建任务，删除任务，改变任务的优先级，任务挂起和恢复等。系统初始化时会自动产生两个任务：一个是空闲任务，它的优先级最低，该任务仅给一个整形变量做累加运算；另一个是系统任务，它的优先级为次低，该任务负责统计当前 cpu 的利用率。

iv. 任务调度

μCOSII 采用的是可剥夺型实时多任务内核。可剥夺型的实时内核在任何时候都运行就绪了的最高优先级的任务。μCOSII 的任务调度是完全基于任务优先级的抢占式调度，也就是最高优先级的任务一旦处于就绪状态，则立即抢占正在运行的低优先级任务的处理器资源。为了简化系统设计，μCOSII 规定所有任务的优先级不同，因为任务的优先级也同时唯一标志了该任务本身。

v. 时间管理

μCOSII 的时间管理是通过定时中断来实现的，该定时中断一般为 10 毫秒或 100 毫秒发生一次，时间频率取决于用户对硬件系统的定时器编程来实现。中断

发生的时间间隔是固定不变的，该中断也成为一个时钟节拍。μCOSII 要求用户在定时中断的服务程序中，调用系统提供的与时钟节拍相关的系统函数，例如中断级的任务切换函数，系统时间函数。

μCOSII 的移植

本移植的层次结构如图1所示。它由用户层、中间件层、μCOSII源码层、μCOSII移植层和驱动库层等五个层次组成。

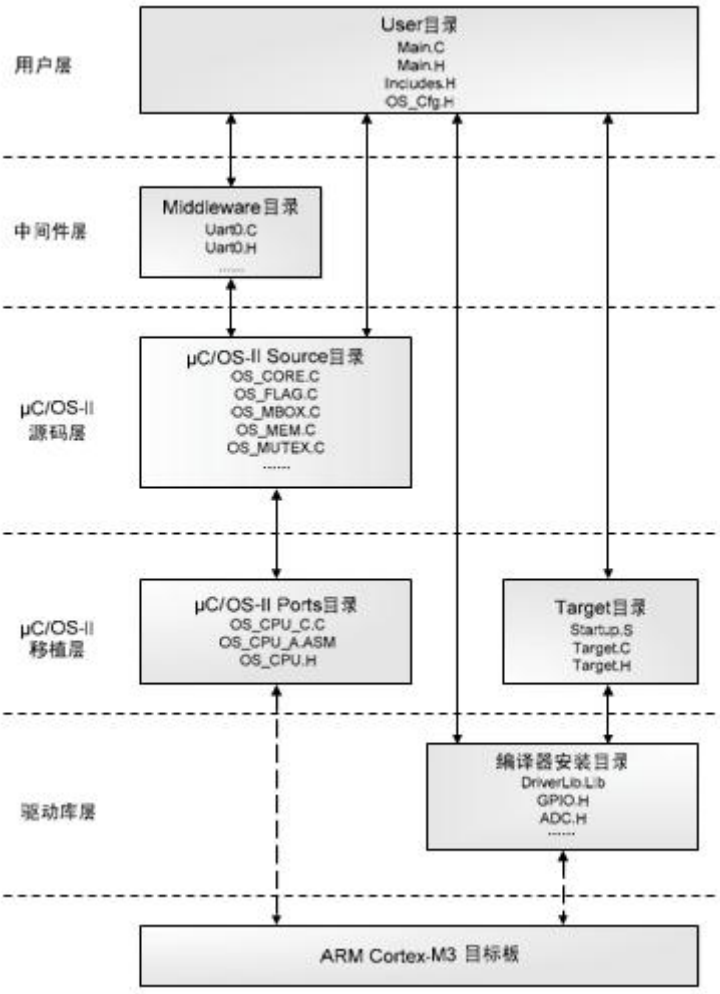


图1　模版层次结构

(1) 用户层的 User 目录存放用户代码与设置。其中 Main.C 文件是用户编写任务的地方；

Main.H 定义任务的堆栈大小、优先级等。OS_Cfg.H 是μCOSII 的配置文件，本模版提供了基于μCOSII2.52 的 OS_Cfg_V252.H 配置文件，用户可把其中的内容复制到 OS_Cfg.H 后，再根据需要修改。Includes.H 是总的头文件，除μCOSII 的源码外，所有.C 的文件都包含它，这样用户所需的头文件和其它声明只需在 Includes.H 中声明一次就行了。

(2) 中间件层的 Middleware 目录存放用户自己编写的中间件，如 Uart0.C、Uart0.H 串口通信中间件等。

(3) μCOSII 源码层的μCOSII\Source 目录存放μCOSII2.52 的源代码（除μCOSII.C 外的全部.C 和.H 的文件）。用户可在《μC/OS-II 嵌入式实时操作系

统》一书的 配套光盘中得到 2.52 的源码。用户只要把源码复制到此目录，不需对源码作任何的修改。

(4) μCOSII 移植层的 μCOSII\Ports 目录存放 μCOSII 基于 LM3S 单片机的移植代码，包括 OS_CPU_C.C、OS_CPU_A.ASM 和 OS_CPU.H 等三个必要的文件。Target 目录中 的 Startup 文件是单片机的启动代码和中断向量表，用户要在其中加入需要的中断服务函数的首地址（后面的实 验例子将详细说明）；Target.C 和 Target.H 提供单片机初始化函数 targetInit()和其它简单的外设控制 API 函数，包括 LED 控制、蜂鸣器控制、按键检测和定时器 0 中断服务等，方便用户调试程序。

(5) 驱动库层是直接面向硬件目标板的层。一般来说，除 μCOSII 外，其它代码都要直接或间接通过它访问硬件。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱、

五、 软件建立工程

该实验工程，包含多个源文件如图 2 所示

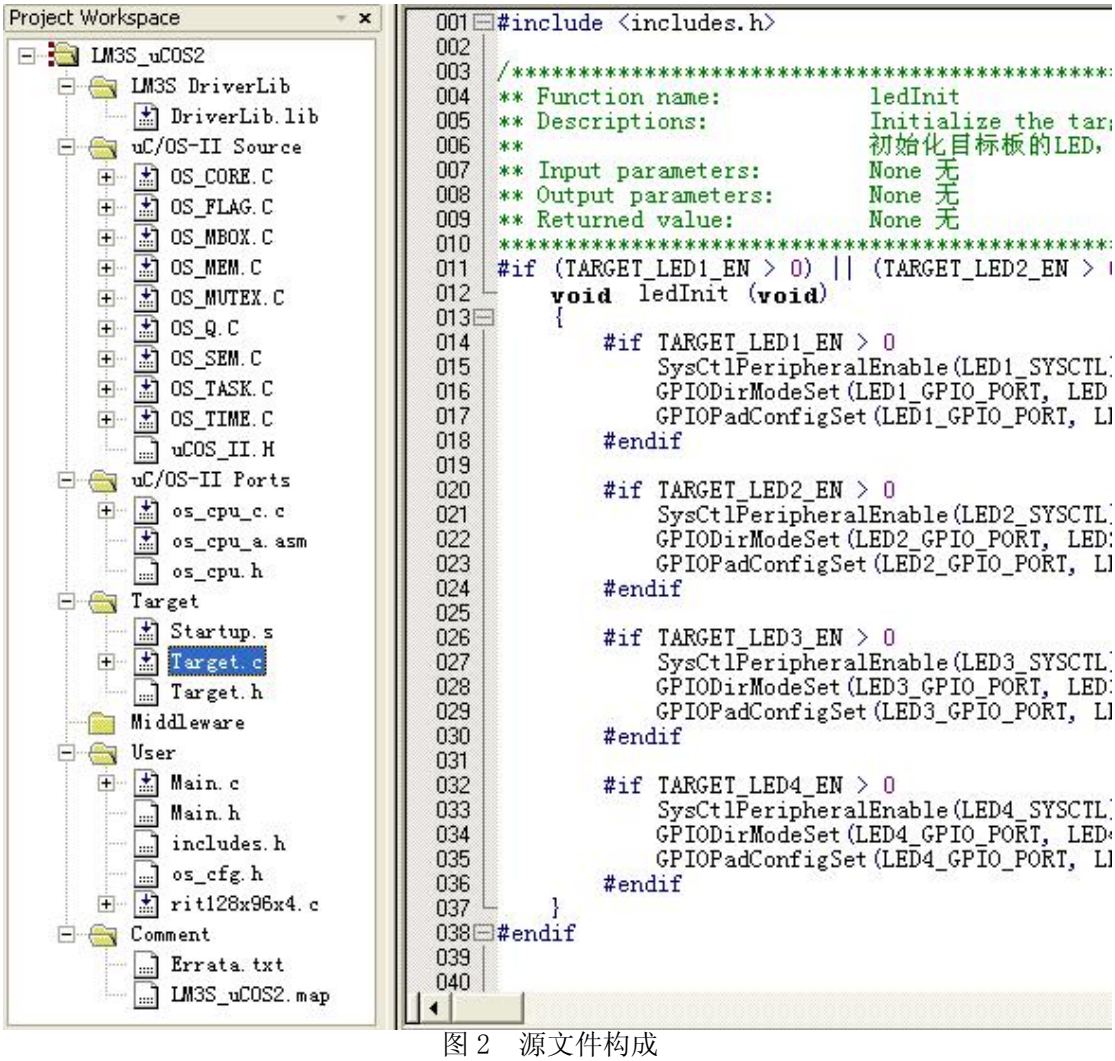


图 2 源文件构成

六、 程序设计

μCOSII 第一个运行的任务，首先要调用 Target.C 文件中的 targetInit()函数初始化工程 所需要的单片机硬件资源，它的函数原型如本实验例程所示。程序首先判断宏定义 PLL_EN 的值（在 Target.H 当中定义），如果为零，则系统时钟不使用 PLL，它等于晶振频率 EXT_CLK 除以 CCLK_DIV（均在 Target.H 文件中设定）；如果不为零，则系统时钟使用 PLL，如图 3 所示。接着初始化内核定时器，μCOSII 使用它的中断源作为时钟节拍。最后，加入用户所需要的其它初始化代码，如本实验中的初始化 LED 等。

```
/*********************************************************************************************************
** Function name:           targetInit
** Descriptions:            Initialize the target board 初始化目标板
** Input parameters:        None 无
** Output parameters:       None 无
** Returned value:          None 无
*********************************************************************************************************/
void  targetInit (void)
{
    #if PLL_EN == 0                                                 /*  Not use PLL  不使用PLL      */
        SysCtlClockSet(CCLK_DIV | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN | EXT_CLK);
                                                                    /*  System clock=               */
                                                                    /*  EXT_CLK/CCLK_DIV            */
                                                                    /*  系统时钟=EXT_CLK/CCLK_DIV   */

    #else                                                           /*  Use PLL  使用PLL            */
        SysCtlClockSet(CCLK_DIV | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN | EXT_CLK);
                                                                    /*  System clock=200MHz/CCLK_DIV*/
                                                                    /*  系统时钟=200MHz/CCLK_DIV    */
    #endif

    tickInit();                                                     /*  Initialize the uC/OS-II tick*/
                                                                    /*  interrupt,using the Kernal's*/
                                                                    /*   timer                      */
    /*
     *  Add you initialization code here.
     *  在这里加入你的初始化代码。
     */
    ledInit();                                                      /*  初始化LED的IO口             */
}
```

图 3　核心代码

七、 实验步骤

程序编译无错误后下载至 LM3S8962 开发板，复位后可观察到 LM3S8962 开发板上的 LED1 每隔 0.5 秒闪烁一次。

八、 思考题

1、 修改实验中的 LED1 闪烁任务，实现闪烁频率由快变慢、再有慢变快的循环。

2、 总结在该操作系统下的 LED1 闪烁任务与实验一有何不同。

实验 22：μCOSII 系统移植 2

一、 实验目的

1、 熟悉实验箱的硬件环境；

2、 熟悉实验系统各部分的使用。

3、 理解 LM3S8962 的各个资源使用。

2、 熟悉 μCOSII 系统的工作特点。

3、 熟悉 μCOSII 系统移植相关知识。

4、 理解 μCOSII 系统中多任务的调度。

二、 实验要求

在 μCOSII 系统上实现多任务协同工作，任务控制 LM3S8962 开发板上的 LED1 闪烁，同时控制 OLED 动态显示字符或图形。

三、 实验原理

μCOSII 可以简单的视为一个多任务调度器，在这个任务调度器之上完善并添加了和多任务操作系统相关的系统服务，如信号量、邮箱等。其主要特点有公开源代码，代码结构清晰、明了，注释详尽，组织有条理，可移植性好，可裁剪，可固化。内核属于抢占式，最多可以管理 60 个任务。

任务调度将在以下情况下发生

1） 高优先级的任务因为需要某种临界资源，主动请求挂起，让出处理器，此时将调度就绪状态的低优先级任务获得执行，这种调度也称为任务级的上下文切换。

2） 高优先级的任务因为时钟节拍到来，在时钟中断的处理程序中，内核发现高优先级任务获得了执行条件(如休眠的时钟到时)，则在中断态直接切换到高优先级任务执行。这种调度也称为中断级的上下文切换。

这两种调度方式在 μCOSII 的执行过程中非常普遍，一般来说前者发生在系统服务中，后者发生在时钟中断的服务程序中。调度工作的内容可以分为两部分：最高优先级任务的寻找和任务切换。其最高优先级任务的寻找是通过建立就绪任务表来实现的。μCOS 中的每一个任务都有独立的堆栈空间，并有一个称为任务控制块 TCB(Task Control Block)的数据结构，其中第一个成员变量就是保存的任务堆栈指针。任务调度模块首先用变量 OSTCBHighRdy 记录当前最高级就绪任务的 TCB 地址，然后调用 OS_TASK_SW()函数来进行任务切换。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱、

五、 软件程序设计

首先要在 main.c 中声明任务，和任务的堆栈。

```
/*******************************************************************************************
  VARIABLES 变量
*******************************************************************************************/
static OS_STK  Task_StartStk[TASK_START_STK_SIZE];                      /*  The stack of s
                                                                        /*  启动任务的堆栈
static OS_STK  Task_LedStk[TASK_LED_STK_SIZE];

static OS_STK  Task_OLEDbStk[TASK_OLEDb_STK_SIZE];

/*******************************************************************************************
   FUNCTION PROTOTYPES 函数声明
*******************************************************************************************/
static void taskStart (void  *parg);                                    /* 启动任务*/
static void taskLed (void  *parg);                                      /*  LED任务*/
static void taskOLEDb(void  *parg);                                     /* OLED任务*/
```

然后在 main.h 中定义任务优先级和堆栈的大小。

```
/*************************************************
  TASK PRIORITIES 任务优先级
*************************************************/
#define  TASK_START_PRIO                0
#define  TASK_LED_PRIO                  1
#define  TASK_OLEDb_PRIO                2
/*************************************************
  TASK STACK SIZES  任务堆栈大小
*************************************************/
#define  TASK_START_STK_SIZE            50
#define  TASK_LED_STK_SIZE              50
#define  TASK_OLEDa_STK_SIZE            50
#define  TASK_OLEDb_STK_SIZE            50

#ifdef __cplusplus
    }
#endif

#endif
/*************************************************
  END FILE
*************************************************/
```

在 main.c 中初始化该任务

```
/*******************************************************************************************
** Function name:           Task_Start
** Descriptions:            Start task
** input parameters:        *p_arg
** output parameters:       无
** Returned value:          无
*******************************************************************************************/
static  void  taskStart (void  *parg)
{
    (void)parg;
    targetInit();                                                       /*  Initialize the target's MCU
                                                                        /*  初始化目标单片机
    #if OS_TASK_STAT_EN > 0
        OSStatInit();                                                   /*  Enable statistics
                                                                        /*  使能统计功能
    #endif

    /*
     *  Create the other tasks here. 在这里创建其他任务
     */
    OSTaskCreate (taskLed, (void *)0,
                  &Task_LedStk[TASK_LED_STK_SIZE-1],
                  TASK_LED_PRIO);                                       /*  初始化taskLed任务

    OSTaskCreate (taskOLEDb, (void *)0,
                  &Task_OLEDbStk[TASK_OLEDb_STK_SIZE-1],
                  TASK_OLEDb_PRIO);                                     /*  初始化taskOLEDb任务

    while (1) {
        OSTaskSuspend(OS_PRIO_SELF);                                    /*  The start task can be pende
                                                                        /*  here. 启动任务可在这里挂起
    }
}
```

六、 建立工程

参考实验一和实验二，程序可参考实验 21。

七、 实验步骤

参考实验 21。程序编译无错误后下载至 LM3S8962 开发板，复位后可观察到 LM3S8962 开发板上的 LED1 每隔 0.5 秒闪烁一次，在 OLED 屏幕上以不同亮度循环显示字符“Hello World!”,同时还有一个“*”在屏幕上来回跳动。

八、 思考题

尝试自己添加几个任务，体会 μCOSII 操作系统上多任务调度。

实验 23：μCOSII 系统移植 3

一、 实验目的

1、 熟悉实验箱的硬件环境；
2、 熟悉实验系统各部分的使用。
3、 理解 LM3S8962 的各个资源使用。
4、 熟悉 μCOSII 系统的工作特点。
5、 熟悉 μCOSII 系统移植相关知识。
6、 理解 μCOSII 系统中多任务的调度。
7、 理解 μCOSII 系统中各任务之间消息的传递。

二、 实验要求

在 μCOSII 系统上实现多任务协同工作，任务控制 LM3S8962 开发板上的 LED1 点亮，同时发送消息控制 PWM 激励蜂鸣器，LED1 熄灭时发消息控制 PWM 停止激励蜂鸣器，同时在 OLED 上显示 PWM 当前工作状态。

三、 实验原理

μCOSII 可以简单的视为一个多任务调度器，在这个任务调度器之上完善并添加了和多任务操作系统相关的系统服务，如信号量、邮箱等。其主要特点有公开源代码，代码结构清晰、明了，注释详尽，组织有条理，可移植性好，可裁剪，可固化。内核属于抢占式，最多可以管理 60 个任务。对一个多任务的操作系统来说，任务间的通信和同步是必不可少的。uC/OS-II 中提供了多同步对象，这些同步对象都有创建，等待，发送，查询的接口用于实现进程间的通信和同步。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱、

五、 软件程序设计

可参考实验 21 和实验 22

如下图程序所示，蜂鸣器任务，当接收到消息时，蜂鸣器开始工作。

```
107 /*********************************************************************
108 ** Function name:           taskA
109 ** Descriptions:            The tasks A
110 ** input parameters:        *parg
111 ** output parameters:       无
112 ** Returned value:          无
113 *********************************************************************
114 static void taskB(void  *parg)
115 {
116     INT8U ucErr;
117     INT8U *ucMsg;
118     (void)parg;
119     while (1) {
120         ucMsg = OSMboxPend(DispMsg, 0, &ucErr);
121         buzOn (*ucMsg);
122     }
123 }
124
```

如下图程序所示，当 LED1 点亮时，发送消息，同时在 OLED 上显示。

```
/*********************************************************************************************************
** Function name:           taskB
** Descriptions:            The tasks B
** input parameters:        *parg
** output parameters:       无
** Returned value:          无
*********************************************************************************************************/
static void taskA(void  *parg)
{
    INT8U ucMsg;
    (void)parg;

    while (1) {
        ledOn(1);
        ucMsg = 2;
        RIT128x96x4StringDraw("PWMgen   ON ", 32, 40, 15);
        OSMboxPost(DispMsg, (void *)&ucMsg);                    /*  发送信号量          */
        OSTimeDly(OS_TICKS_PER_SEC / 1);                        /*  延时1秒             */
        ledOff(1);                                              /*  关闭LED1            */
        ucMsg = 1;
        RIT128x96x4StringDraw("PWMgen   OFF", 32, 40, 15);
        OSMboxPost(DispMsg, (void *)&ucMsg);                    /*  发送信号量          */
        OSTimeDly(OS_TICKS_PER_SEC / 1);                        /*  延时1秒             */

        OSTimeDly(2);
    }
}

/*********************************************************************************************************
  END FILE
*********************************************************************************************************/
```

六、 建立工程

参考实验 21 和实验 22。

七、 实验步骤

参考实验 21。程序编译无错误后下载至 LM3S8962 开发板，复位后可观察到 LM3S8962 开发板上的 LED1 每隔 1 秒闪烁一次，在 OLED 屏幕上显示“PWMgen ON”或者是“PWMgen OFF”，当 LED1 点亮时，蜂鸣器发出声音；LED1 熄灭时蜂鸣器停止发声，OLED 上会同步显示。

八、 思考题

1、 修改实验中的 LED1 闪烁频率，验证蜂鸣器的变化是否同步。

2、 修改实验中的 OLED 显示部分，使其单独一个任务的形式存在，以查询的方式，查询当前消息，并根据消息内容的不同，在 OLED 上显示不同的结果。

3、 添加按键任务，当按下某按键时，LED1 熄灭时，蜂鸣器发声；LED1 点亮时，蜂鸣器停止发声；按下另一按键时，恢复到之前的点亮方式。

4、 尝试自己添加几个任务，体会 μCOSII 操作系统上多任务调度，和任务之间是如何进行通讯。

实验 24：μCOSII 系统移植 4

一、 实验目的

1、 熟悉实验箱的硬件环境；
2、 熟悉实验系统各部分的使用。
3、 理解 LM3S8962 的各个资源使用。
4、 熟悉 μCOSII 系统的工作特点。
5、 熟悉 μCOSII 系统移植相关知识。
6、 理解 μCOSII 系统中多任务的调度。
7、 重点理解 μCOSII 系统中抢占式任务调度方式。

二、 实验要求

在实验 25 的程序基础上实现高优先级任务抢占式任务调度方式，低优先级任务控制 LM3S8962 开发板上的 LED1 闪烁，同时控制 OLED 动态显示字符或图形；高优先级任务激活方式为时间方式，每隔 10 秒启动一次，该任务启动时，停止其它任务的执行，该高优先级任务执行一轮密集计算，约 3 秒左右，执行完毕后任务挂起，系统可以响应其它低优先级任务。

三、 实验原理

μCOSII 可以简单的视为一个多任务调度器，在这个任务调度器之上完善并添加了和多任务操作系统相关的系统服务，如信号量、邮箱等。其主要特点有公开源代码，代码结构清晰、明了，注释详尽，组织有条理，可移植性好，可裁剪，可固化。内核属于抢占式，最多可以管理 60 个任务。

任务调度将在以下情况下发生

1） 高优先级的任务因为需要某种临界资源，主动请求挂起，让出处理器，此时将调度就绪状态的低优先级任务获得执行，这种调度也称为任务级的上下文切换。

2） 高优先级的任务因为时钟节拍到来，在时钟中断的处理程序中，内核发现高优先级任务获得了执行条件(如休眠的时钟到时)，则在中断态直接切换到高优先级任务执行。这种调度也称为中断级的上下文切换。

这两种调度方式在 μCOSII 的执行过程中非常普遍，一般来说前者发生在系统服务中，后者发生在时钟中断的服务程序中。调度工作的内容可以分为两部分：最高优先级任务的寻找和任务切换。其最高优先级任务的寻找是通过建立就绪任务表来实现的。μCOS 中的每一个任务都有独立的堆栈空间，并有一个称为任务控制块 TCB(Task Control Block)的数据结构，其中第一个成员变量就是保存的任务堆栈指针。任务调度模块首先用变量 OSTCBHighRdy 记录当前最高级就绪任务的 TCB 地址，然后调用 OS_TASK_SW()函数来进行任务切换。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱、

五、 软件程序设计

在 main.c 中声明任务，和任务的堆栈。

```
/*********************************************************************
** Function name:           taskCreate
** Descriptions:            CREATE APPLICATION TASKS
** input parameters:        无
** output parameters:       无
** Returned value:          无
*********************************************************************/
static  void  taskCreate (void)
{
    DispMsg = OSMboxCreate((void*)0);
    OSTaskCreate (task0, (void *)0,
                  &Task_0Stk[TASK_0_STK_SIZE-1], TASK_0_PRIO);

    OSTaskCreate (taskA, (void *)0,
                  &Task_AStk[TASK_A_STK_SIZE-1], TASK_A_PRIO);

    OSTaskCreate (taskB, (void *)0,
                  &Task_BStk[TASK_B_STK_SIZE-1], TASK_B_PRIO);
}

```

图 1

在 main.h 中定义任务优先级和堆栈的大小。

```
/*************************************************
  TASK PRIORITIES 任务优先级
*************************************************/
#define  TASK_START_PRIO                0
#define  TASK_0_PRIO                    1
#define  TASK_A_PRIO                    2
#define  TASK_B_PRIO                    3

/*************************************************
  TASK STACK SIZES  任务堆栈大小
*************************************************/
#define  TASK_START_STK_SIZE           50
#define  TASK_0_STK_SIZE               50
#define  TASK_A_STK_SIZE               50
#define  TASK_B_STK_SIZE               50

#ifdef __cplusplus
    }
#endif

#endif
```

图 2

六、 建立工程

参考实验一和实验二，程序可参考实验 21 和实验 23。

七、 实验步骤

参考实验 23。程序编译无错误后下载至 LM3S8962 开发板，复位后可观察到 LM3S8962 开发板上的 LED1 每隔 1 秒闪烁一次，在 OLED 屏幕上显示“PWMgen ON”或者是“PWMgen OFF”，当 LED1 点亮时，蜂鸣器发出声音；LED1 熄灭时蜂鸣器停止发声，OLED 上会同步显示。当第 10 秒来临时，任务 task0 准备完成，因为该任务优先级较其它两个任务要高（参看图 2），系统挂起其它两个任务，开始响

应 task0 任务，task0 任务约占用 3 秒钟时间，期间会在 OLED 的下方显示倒计时，PWM 和 LED1 任务均处于 Ready 状态，等待 task0 任务挂起后，方可再次运行。

八、 思考题

尝试将该实验程序中的 task0 任务的优先级修改为最低优先级，体会 μCOSII 操作系统上抢占式任务调度。

实验 30：自检程序

注意：在选择直流电机选项时需注意并注意 P1、P2、N3~N6 这几个器件，如发现这些器件中某个或某几个发热量较大，请立刻断开电源，查找并修改程序，待发热器件冷却之后再行调试，如若不然可能造成器件损坏。

一、 实验目的

a) 熟悉实验箱的硬件环境；
b) 熟悉实验系统各部分的使用。

二、 实验要求

以上个实验的总结和回顾。

三、 实验环境

计算机、RealVIEW MDK 集成开发环境、ULINK2 适配器、实验箱。

四、 软件程序设计

该实验工程，包含 4 个源文件： DriverLib.lib、Startup.S、define.h、和 selftest.c，这里只介绍 selftest.c 源文件,大家在这个源文件中可以看到是之前所做实验程序的集合，如有不明，可参照前面的实验。

五、 建立工程

参考实验一和实验二。

六、 实验步骤

参考前面的所有实验。

注意：在选择直流电机选项时需注意并注意 P1、P2、N3~N6 这几个器件，如发现这些器件中某个或某几个发热量较大，请立刻断开电源，查找并修改程序，待发热器件冷却之后再行调试，如若不然可能造成器件损坏。

第二部分：LabVIEW 环境实验

实验 1：模拟输入实验

一、 实验目的

1、了解实验板的硬件环境；

2、初步了解如何驱动实验板模拟输出口使用 。

二、 实验要求

通过编写一个显示器驱动小程序，初步了解 AI 的使用。

三、 实验原理

直接驱动 AI，将模拟输出电压在前面板和实验板 OLED 中显示。

四、 实验环境

计算机、RealVIEW MDK 集成开发环境、LabVIEW、LabVIEW Embedded Module for ARM Controllers、ULINK2 仿真器、实验板。

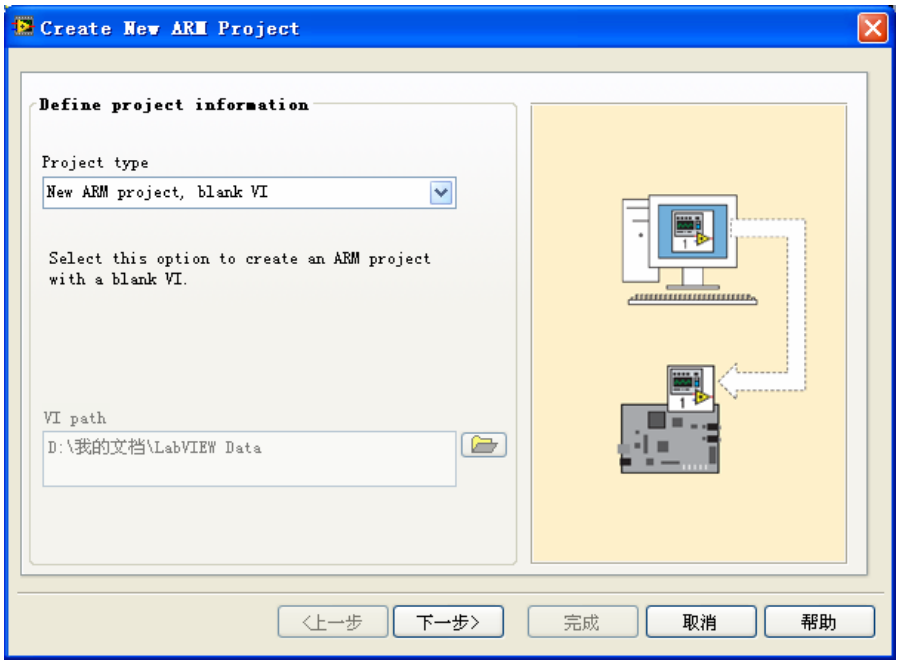
五、 实验步骤

1、 将实验板与计算机连接。

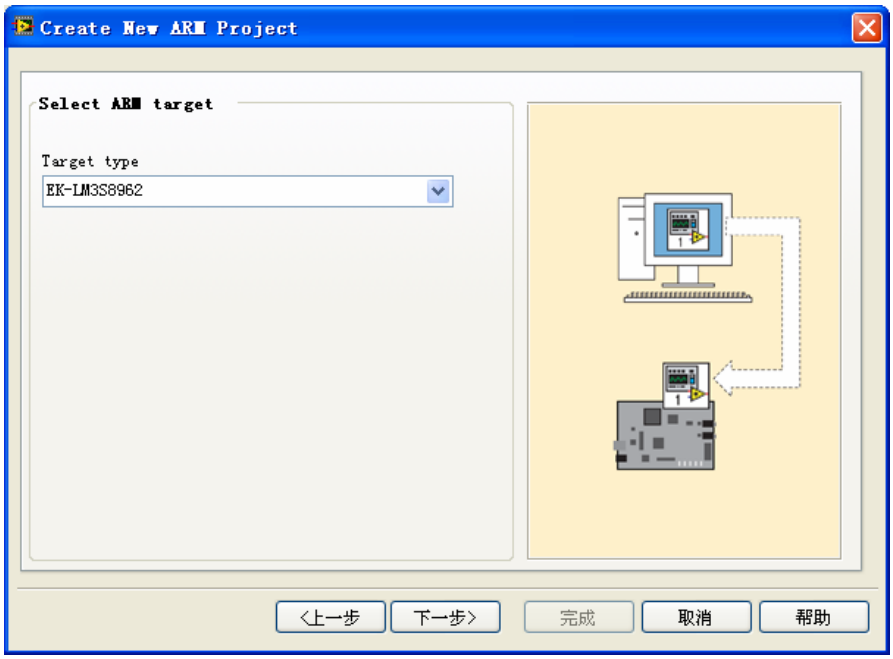
2、 通过 USB 将实验板与计算机连接提供电源。

3、 启动 LABVIEW，选择 **终端** 选择 **ARM Project**，单击 **开始** 新建工程（参见下图）：

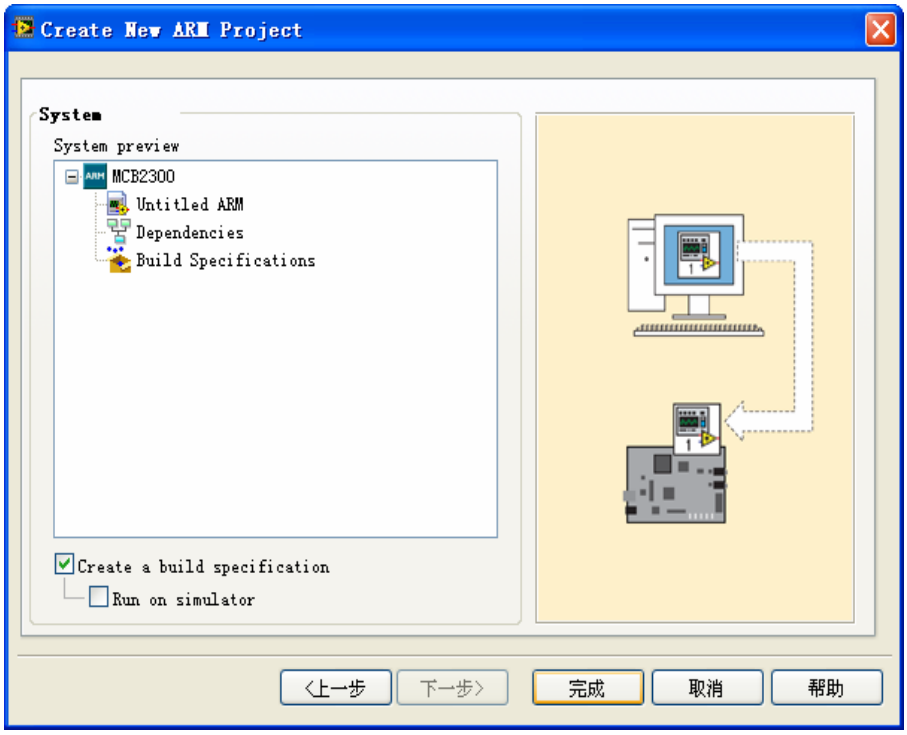
点击 开始 后，转入新项目创建页面（参见下图），然后点击 下一步 ，进入新页面：



在这个页面中选择实验板的类型（**Target type**），这里我们选择 **EK-LM3S8962**，然后点击 下一步 （参见下图），进入新页面：

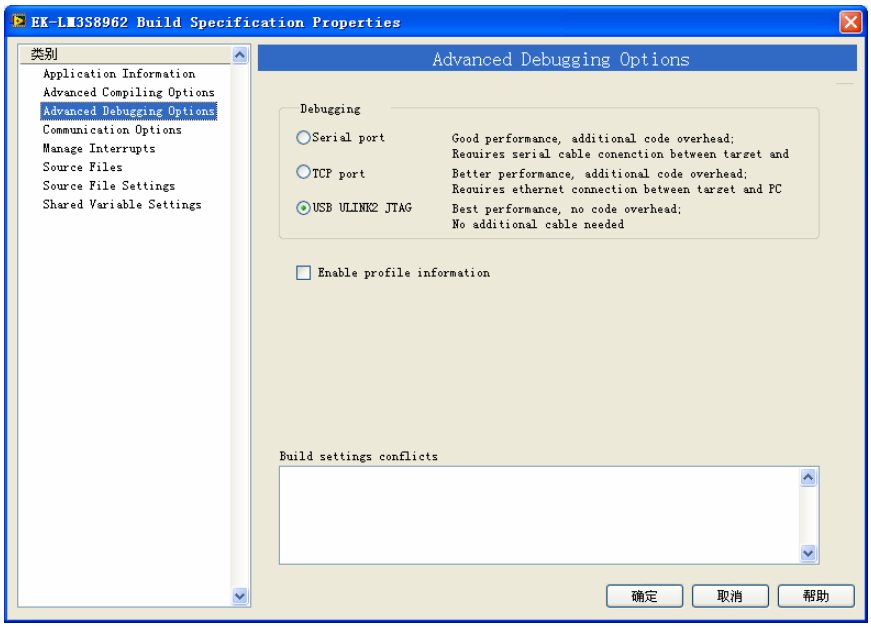


在这一页面中勾选 **Create a build specification**，（**Run on Simulator** 不能勾选）然后点击 **完成**（参见下图）。

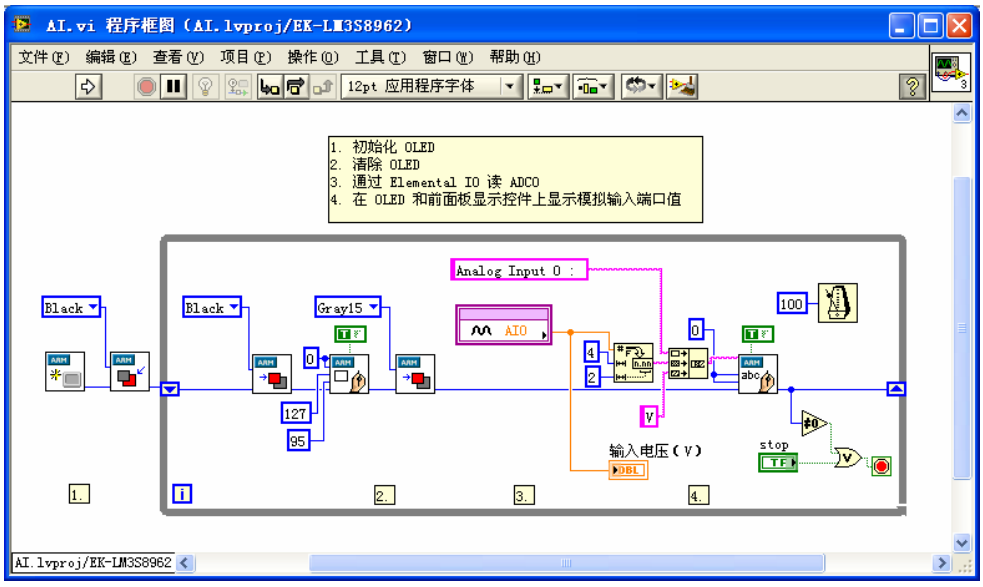


然后保存“项目”与“VI”，项目名称保存为“AI.lvproj”，VI 名称保存为“AI.vi”：

4、 重要参数设置：鼠标右键点击在项目浏览器中的 **程序生成规范** 下的 **Application**，选择 **属性**，在 **EK-3S8962 Build Specification** 页面下的 **Application Information** 页选择 **Run on target using ULINK2**，在下下页的 **Advanced Debugging Options** 中选择 **USB ULINK2 JTAG**，见下图：

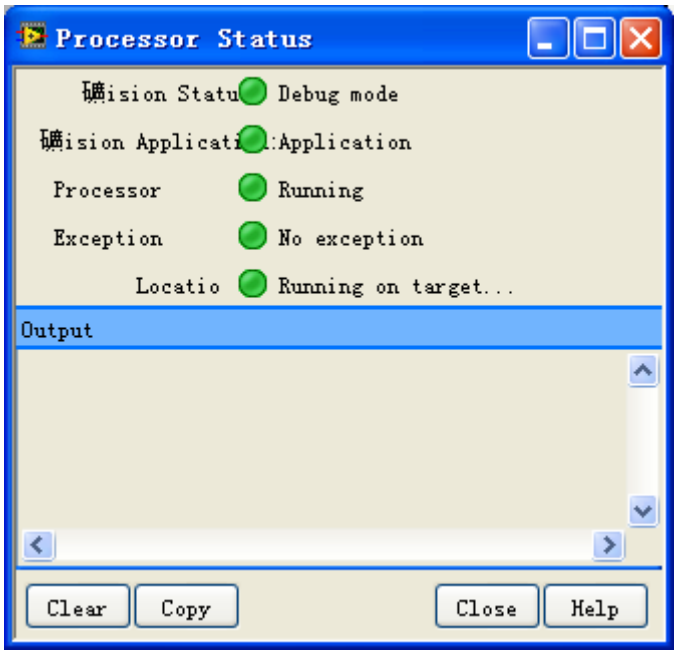


5、 在项目浏览器中双击 **AI.vi**，参考下图建立程序框图：



6、 单击运行按扭，程序将自动编译、链接并下载到实验板上运行，观察处理器状态，在目标板的 ADC0 引脚上输入电压，电压值在前面板显示控件和 OLED 中显示。

注意：模拟输入电压不能大于 3.3V，否则有可能烧毁开发板上 AD 芯片。


Processor Status
Debug mode
:Application
Running
No exception
Running on target...
Output
Clear
Copy
Close
Help

实验 2：数字量输入输出实验

一、 实验目的

初步了解数字输入/输出端口的使用。

二、 实验要求

通过编写数字输入/输出端口 DI0 实验程序，初步了解数字输入端口的使用方法与规则。

三、 实验原理

通过实验板上按键产生数字量，利用这个数字量控制实验板上 LED 指示灯的闪烁速度。

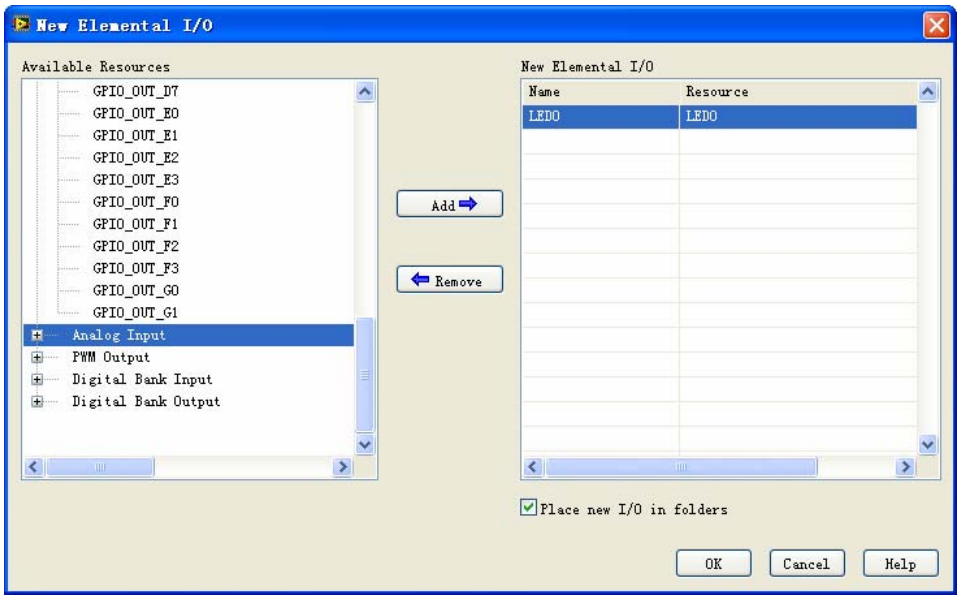
四、 实验环境

计算机、RealVIEW MDK 集成开发环境、LabVIEW、LabVIEW Embedded Module for ARM Controllers、ULINK2 仿真器、实验板。

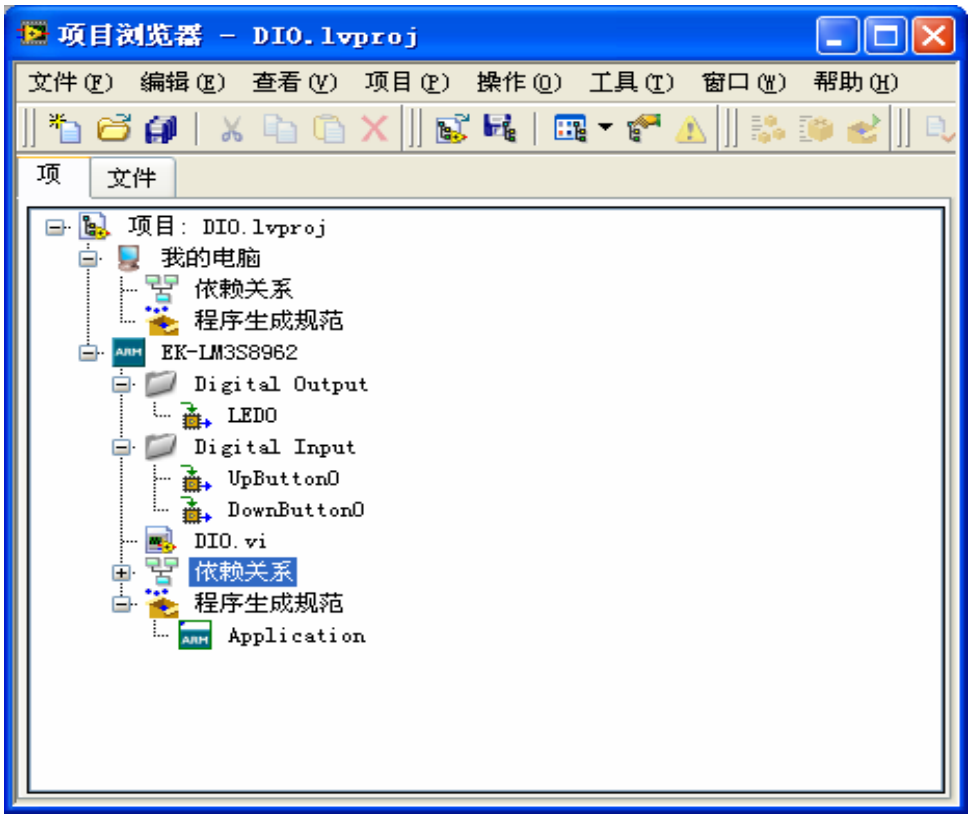
五、 实验步骤

1、 建立工程：“**DI0..lvproj**”；

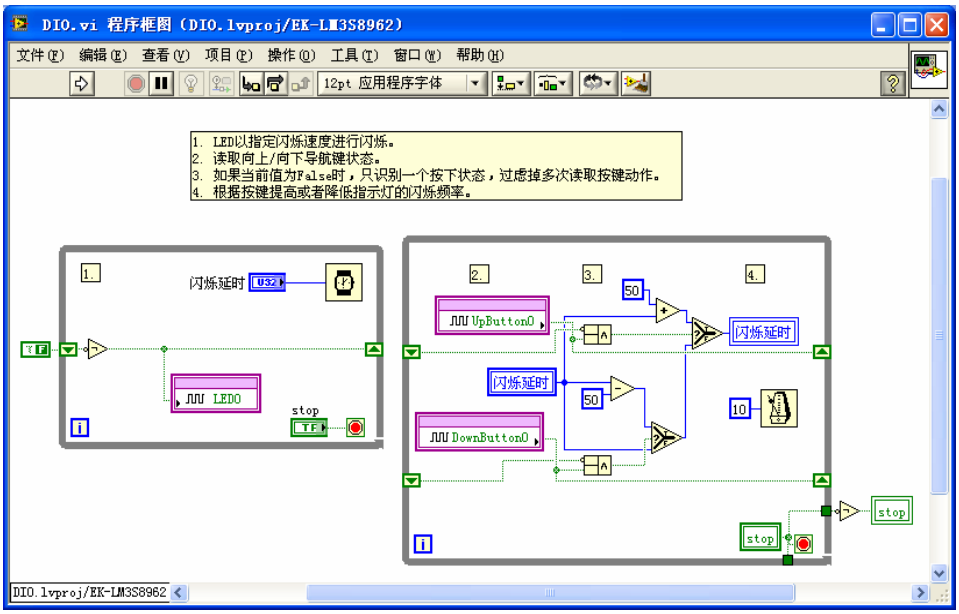
2、 在 **项目浏览器中** 右键单击 **EK-LM3S8982**，选择 **新建→Elemental I/O**，添加数字输入端口 **LED0** 和数字输入端口 **UpButton0 和 DownButton0**（参考下图）：



添加完后点击 **OK**，添加完端口后的 **项目浏览器** 显示如下图：



3、 打开**DI0.vi**程序框图，按住左键将项目浏览器中**LED0**和**UpButton0、DownButton0**拖动到程序框图中，并参考下图建立程序框图：



4、 程序编写完成后点击程序前面板上的运行按键，程序将开始编译、链接并下载到实验板上。

5、 运行程序，按下实验板的 **UpButton0**、**DownButton0**导航按键或直接在前面板中改变 **闪烁延时** 输入控件，**LED0**指示灯闪烁速度相应改变。

实验 3：LED 指示灯实验

一、 实验目的

初步了解数字输出端口的使用；

二、 实验要求

通过编写一个数字输出端口小程序，了解其使用规则。

三、 实验原理

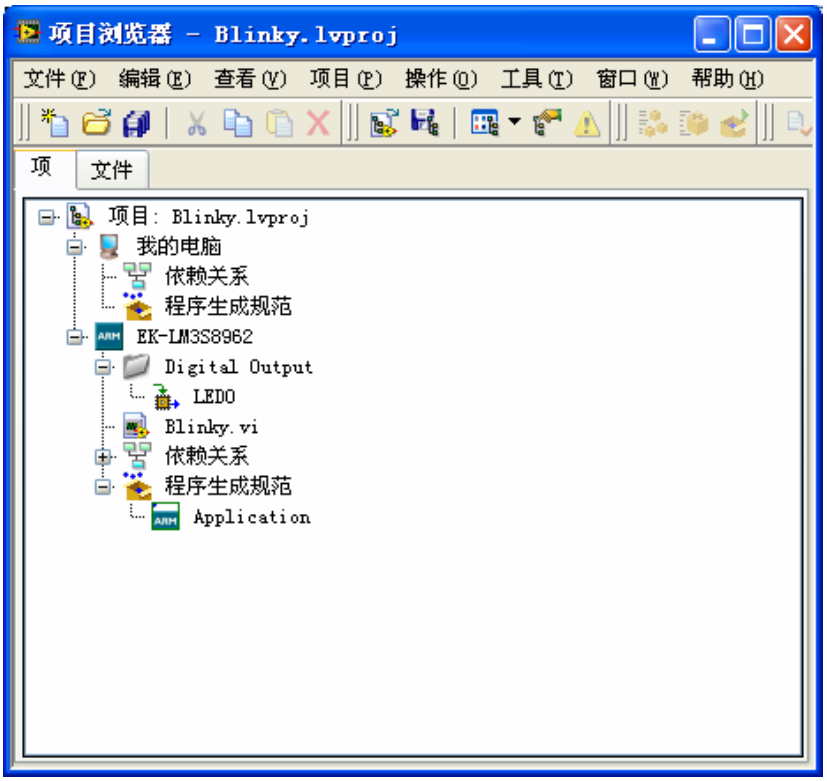
通过数字输出端口控制实验板上 LED 指示灯的工作。

四、 实验环境

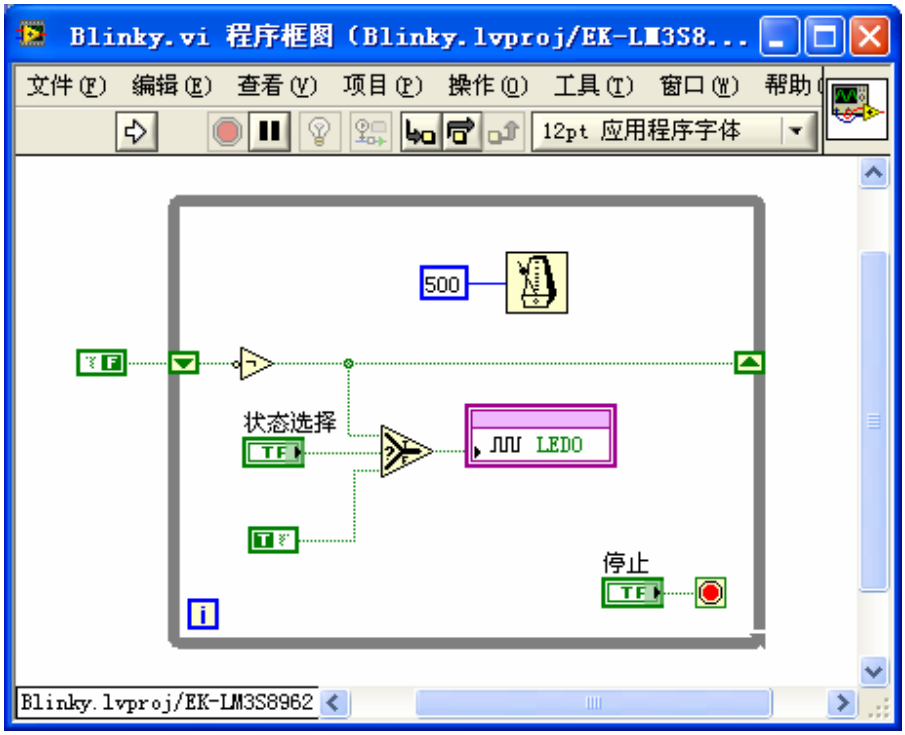
计算机、RealVIEW MDK 集成开发环境、LabVIEW、LabVIEW Embedded Module for ARM Controllers、ULINK2 仿真器、实验板。

五、 实验步骤

1、 参考下图建立工程：**“Blinky.lvproj”**：



2、 参考下图建立 **Blinky.vi**程序前面板与后面板：



3、 运行程序，可通过前面板控制指示灯的闪烁状态，观察开发板上的指示灯。

实验 4：OLED 显示实验

一、 实验目的

1、 学习实验板上 OLED 显示屏的使用；

2、 学习使用 LabVIEW 软件关于实验的 OLED 子 VI 使用。

二、 实验要求

通过编写波形程序，了解实验板显示器及 OLED 子 VI 的使用。

三、 实验原理

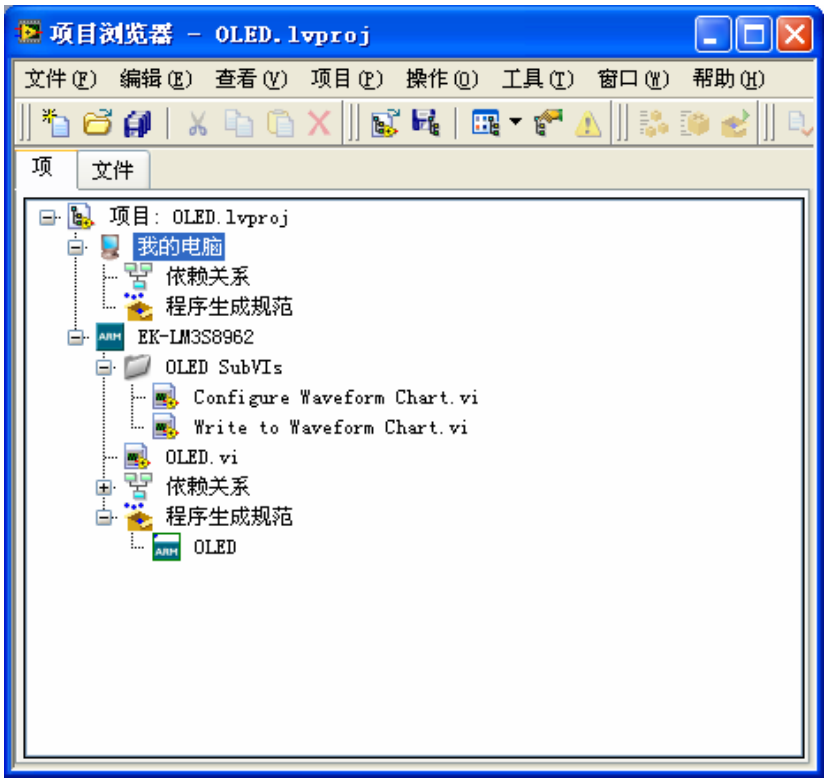
将程序产生的波形图送到实验板上 OLED 显示屏上显示出来。

四、 实验环境

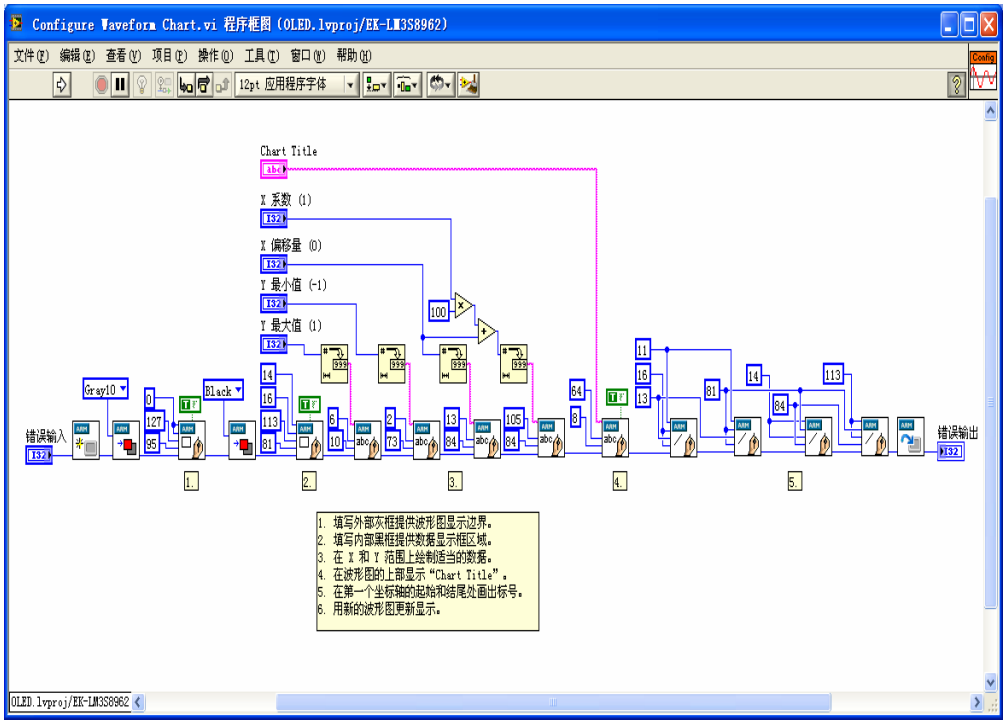
计算机、RealVIEW MDK 集成开发环境、LabVIEW、LabVIEW Embedded Module for ARM Controllers、ULINK2 仿真器、实验板。

五、 实验步骤

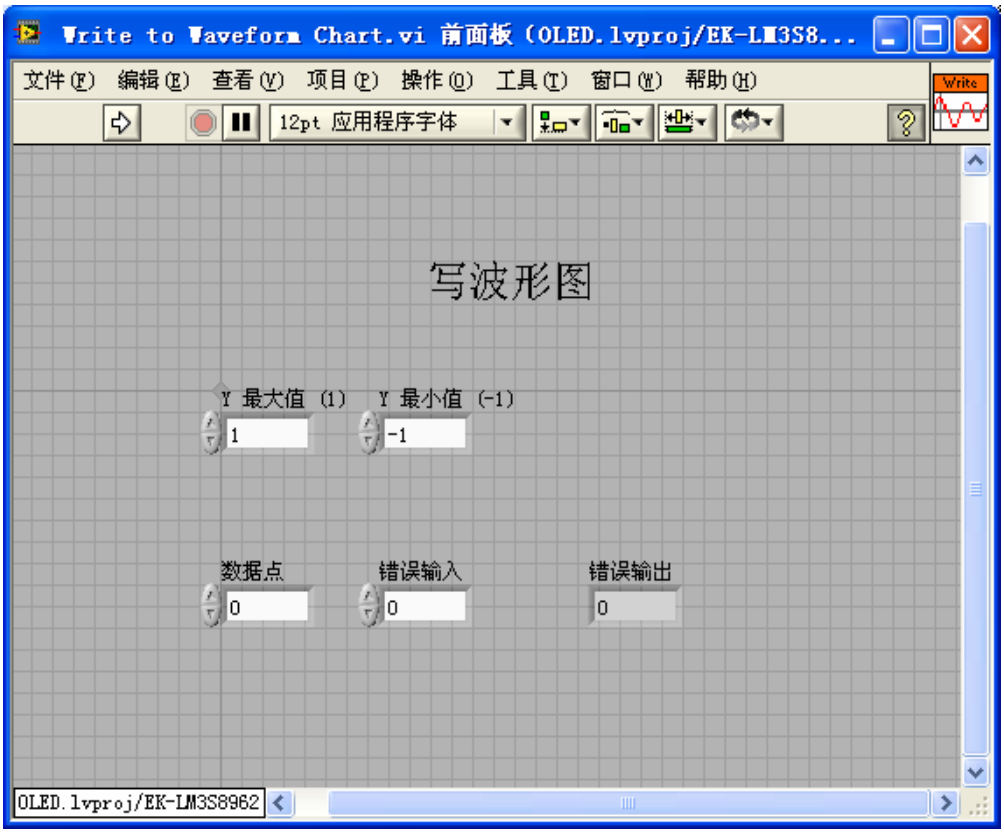
1、 参考下图建立工程：“**OLED.lvproj**”：

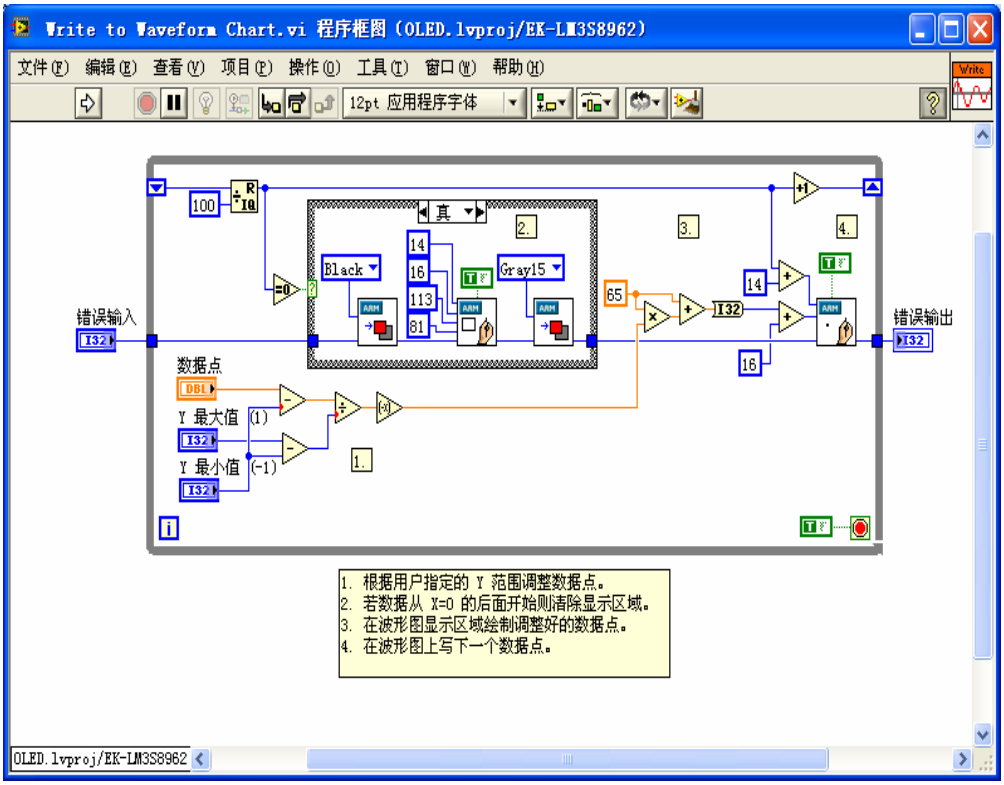


2、 参考下图建立子 VI **Configure Waveform Chart.vi**：

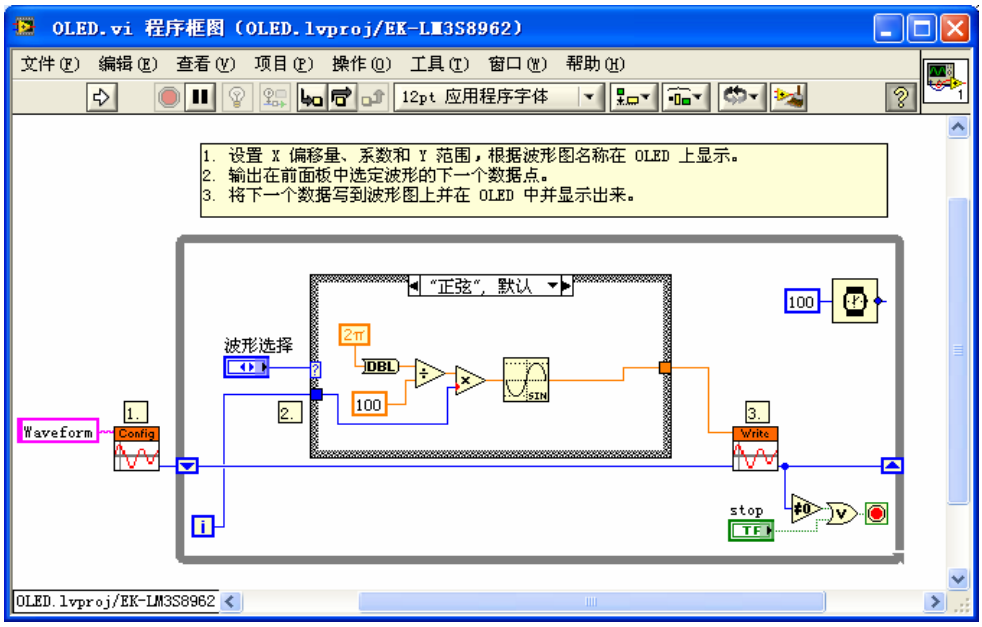


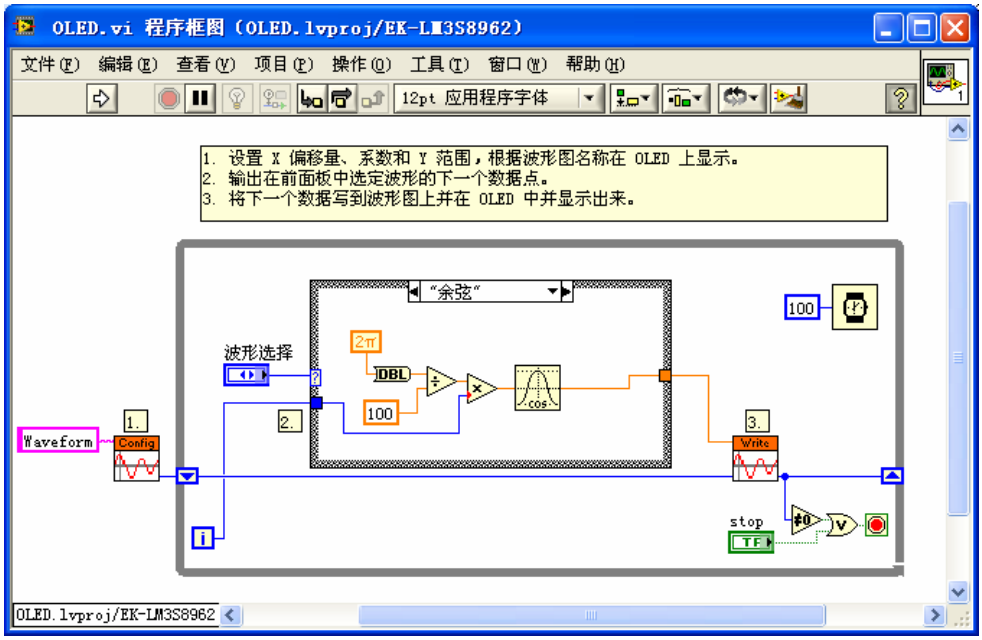
3、 参考下图建立子 VI **Configure Waveform Chart.vi**：

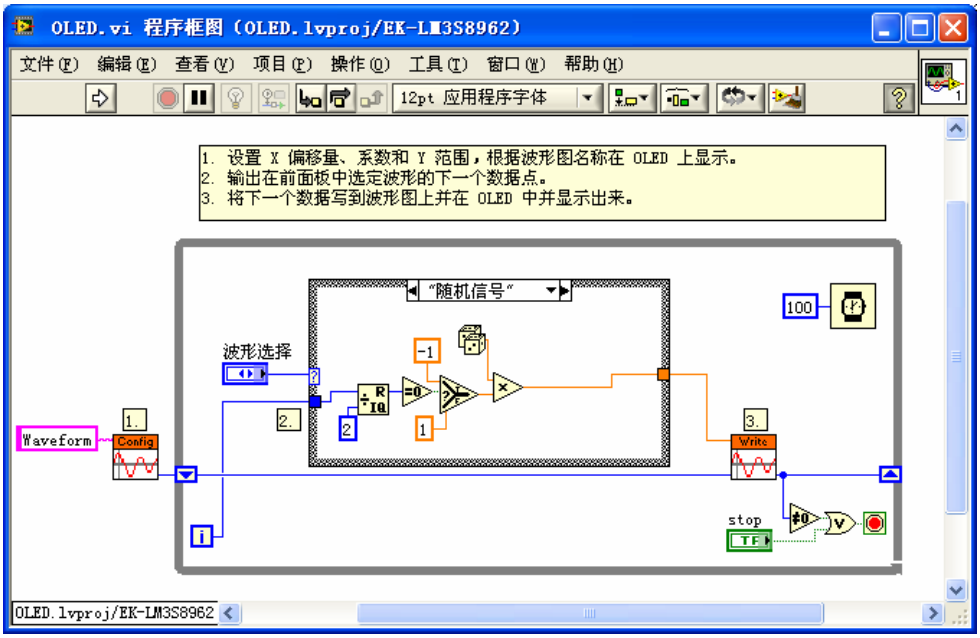


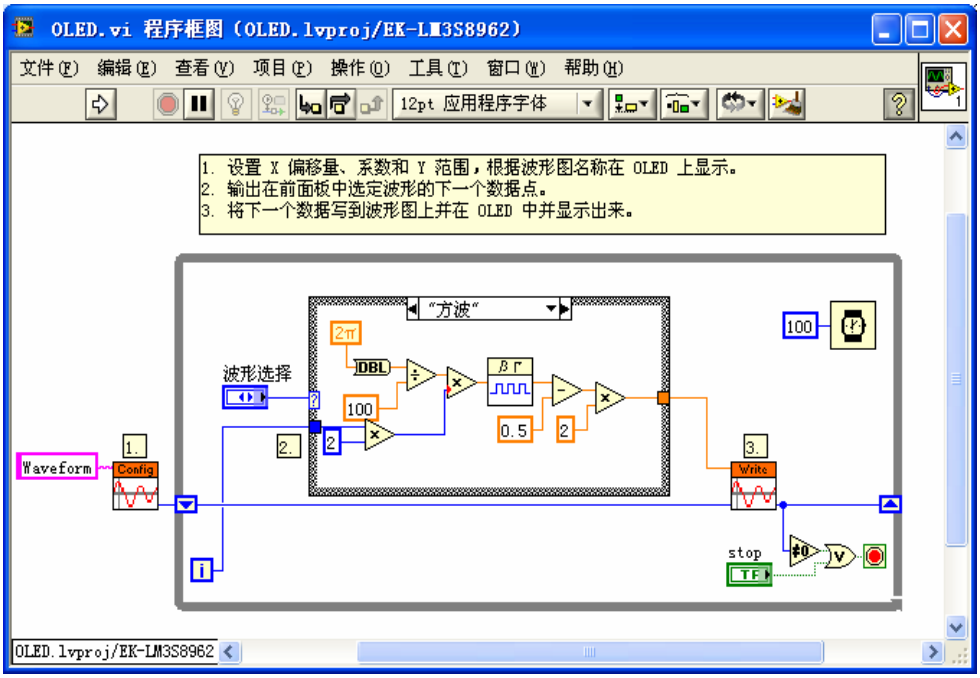


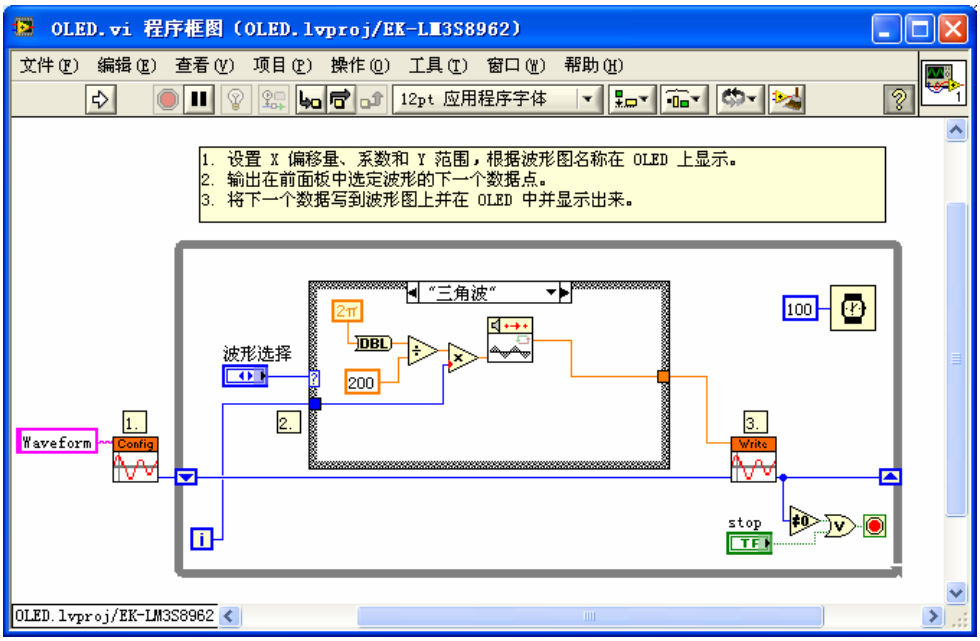
4、 参考下图建立主程序 **OLED.vi**：






OLED.vi 程序框图 (OLED.lvproj/EK-LM3S8962)
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
12pt 应用程序字体
1. 设置 X 偏移量、系数和 Y 范围，根据波形图名称在 OLED 上显示。
2. 输出在前面板中选定波形的下一个数据点。
3. 将下一个数据写到波形图上并在 OLED 中并显示出来。
"随机信号"
波形选择
Waveform
Config
Write
stop
TF
100
OLED.lvproj/EK-LM3S8962


OLED.vi 程序框图 (OLED.lvproj/EK-LM3S8962)
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
12pt 应用程序字体
1. 设置 X 偏移量、系数和 Y 范围，根据波形图名称在 OLED 上显示。
2. 输出在前面板中选定波形的下一个数据点。
3. 将下一个数据写到波形图上并在 OLED 中并显示出来。
"方波"
波形选择
Waveform
Config
Write
stop
TF
100
DBL
0.5
OLED.lvproj/EK-LM3S8962



5、 保存、编译、运行程序，通过在程序前面板控件中选择不同的信号种类，改变实验板 OLED 显示屏上显示的波形。

实验 5：PID 控制与 PWM 输出实验

一、 实验目的

初步了解 PID 算法与 PWM 的工作原理；

二、 实验要求

通过编写一个 PID 控制与 PWM 输出小程序，进一步掌握基于 LabVIEW 的嵌入式（ARM）模块工程的建立、运行；初步了解 PID 算法与 PWM 的使用。

三、 实验原理

通过 ADC0 端口输入电压，将它连接到 OLED 中并显示出来，主程序前面板输出控制计数值并将计数值通过 OLED 显示出来。

四、 实验环境

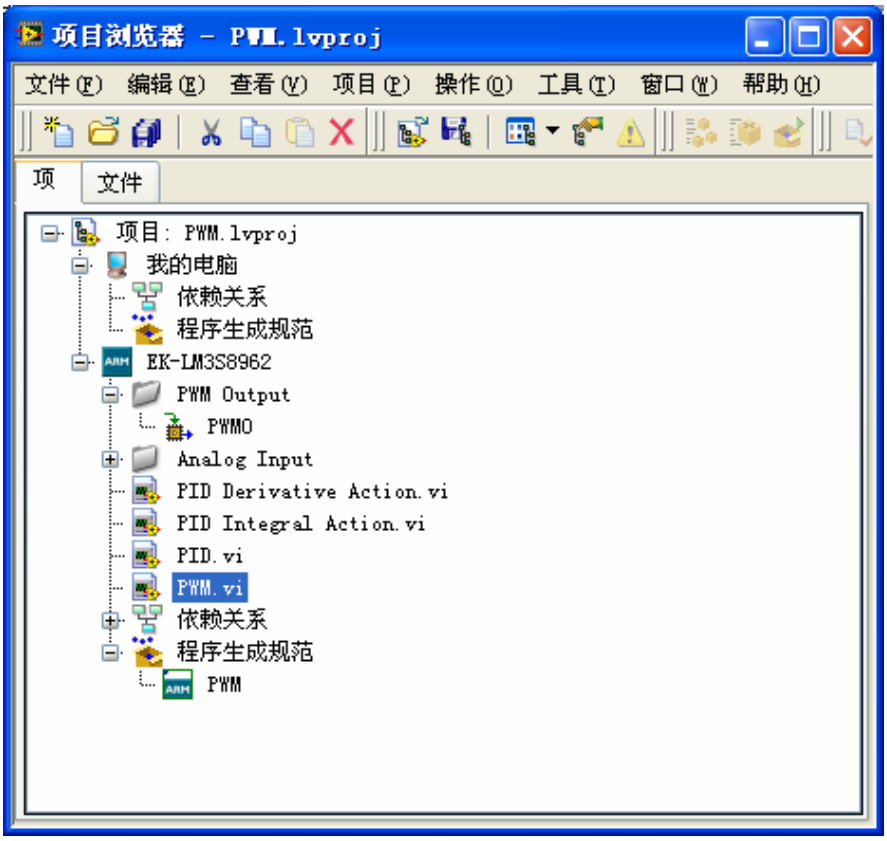
计算机、RealVIEW MDK 集成开发环境、LabVIEW、LabVIEW Embedded Module for ARM Controllers、ULINK2 仿真器、实验板。

五、 实验步骤

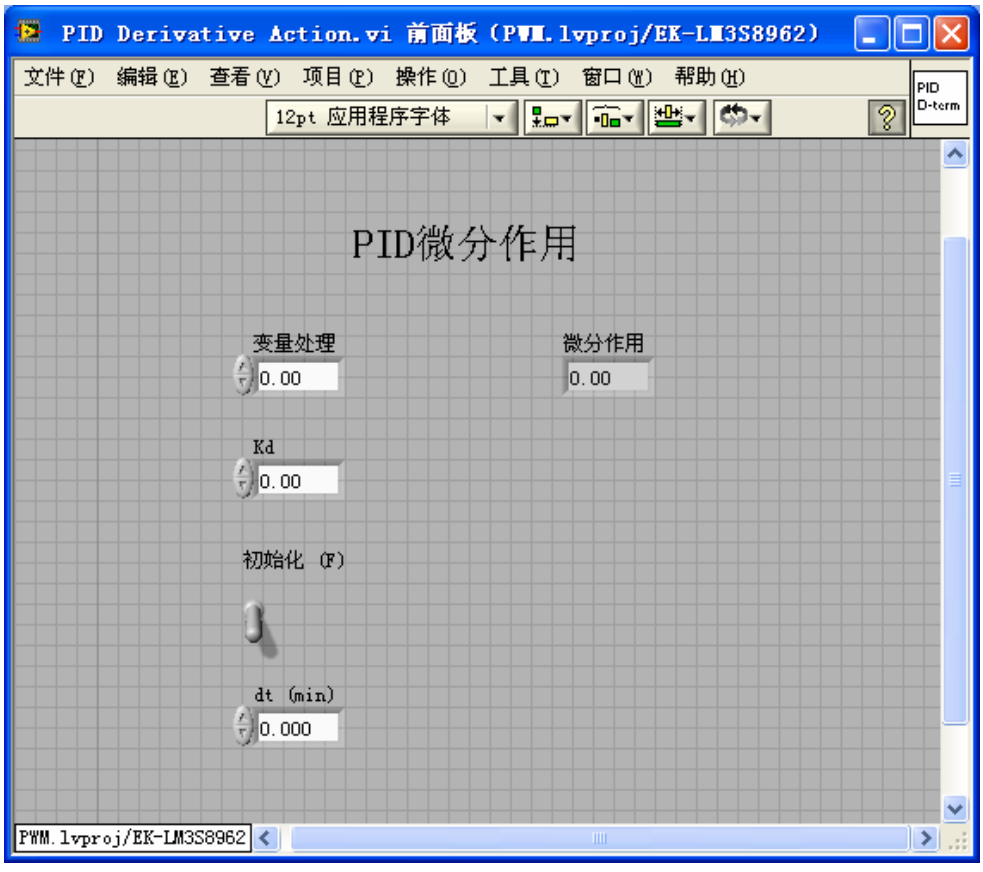
1、 建立工程：**“PWM.lvproj”**；

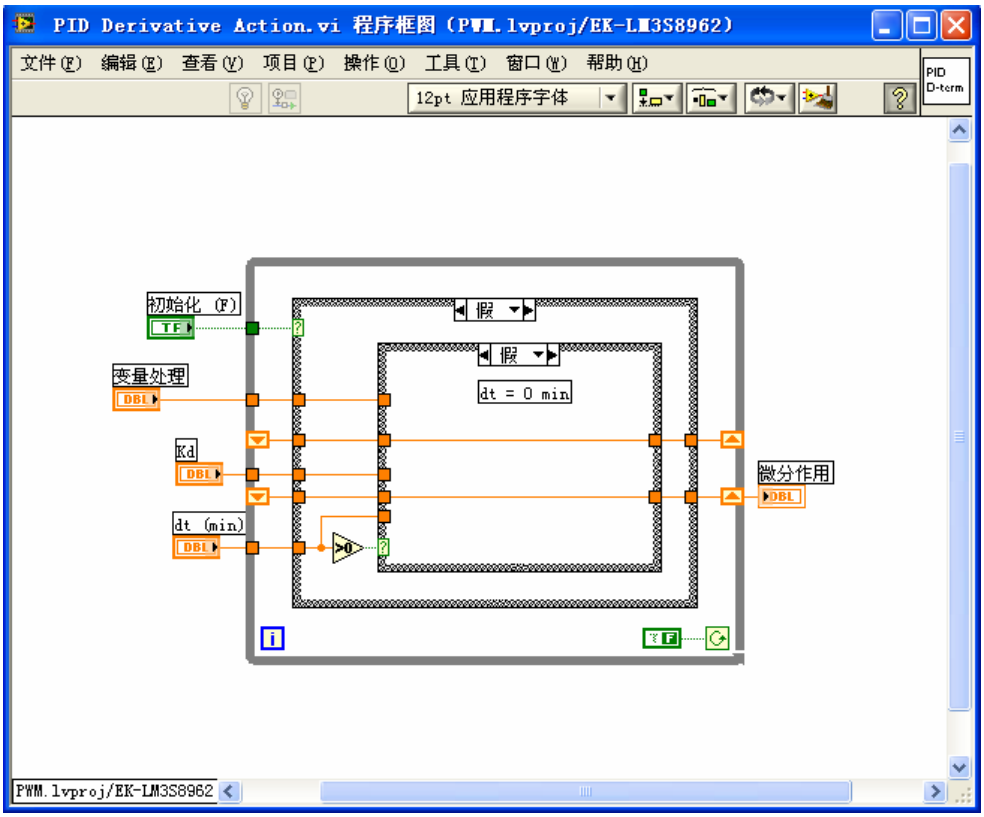
2、 添加模拟输入端口 **AI0**和脉宽调制端口 **PWM1**；

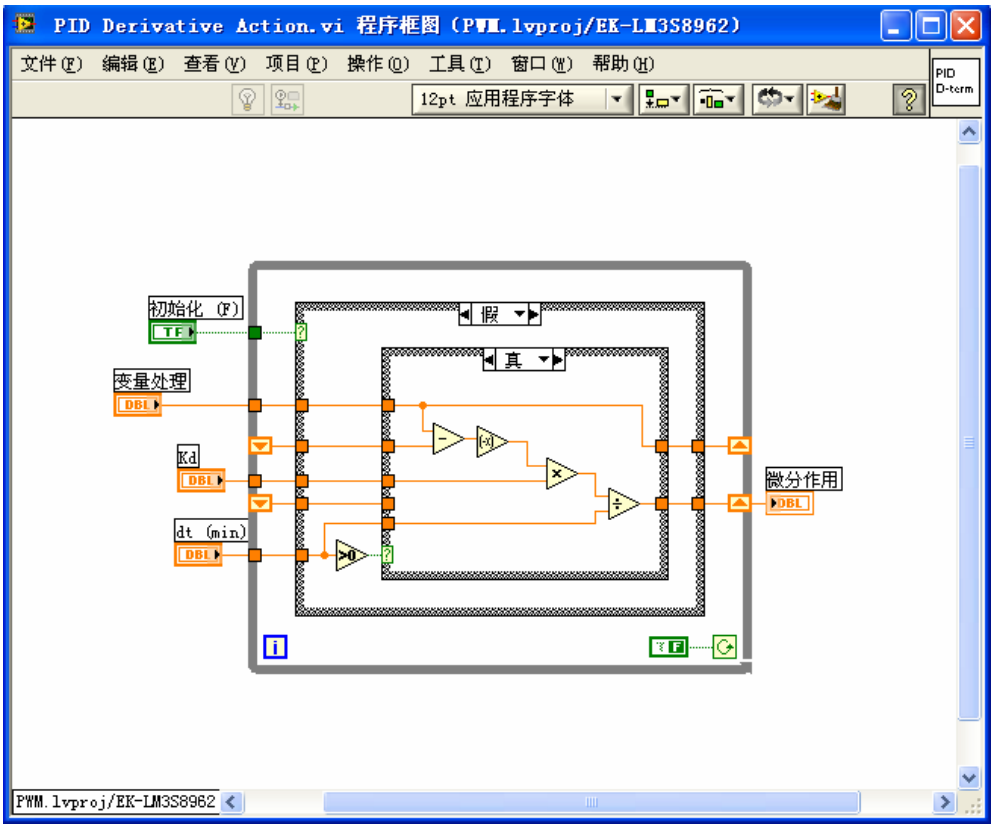
3、 项目浏览器中右击 **EK-LM3S8962**选择新建→虚拟文件夹并将其重命名为 **“PID SubVIs”**；右击 **PID SubVIs**虚拟文件夹，新建三个 VI 并分别保存为**“PID Derivative Action.vi”、“PID Integral Action.vi”和“PID.vi”**（参考下图）：

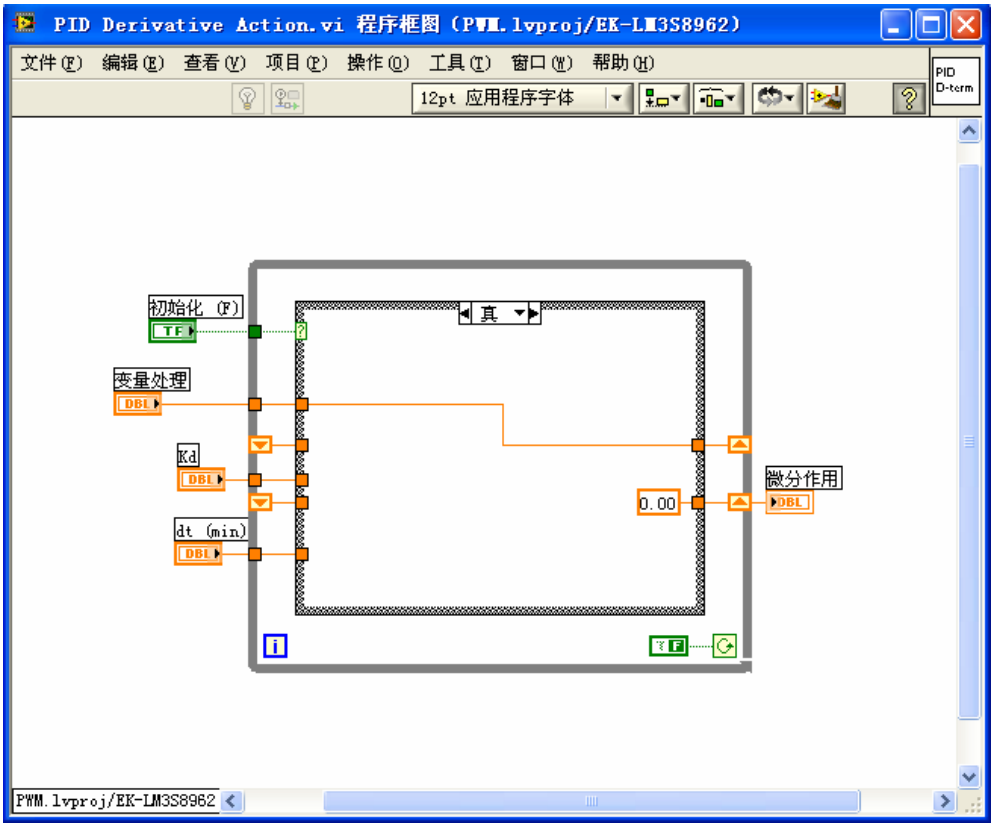


4、 参考下图建立子 VI **PID Derivative Action.vi**前面板和程序框图：

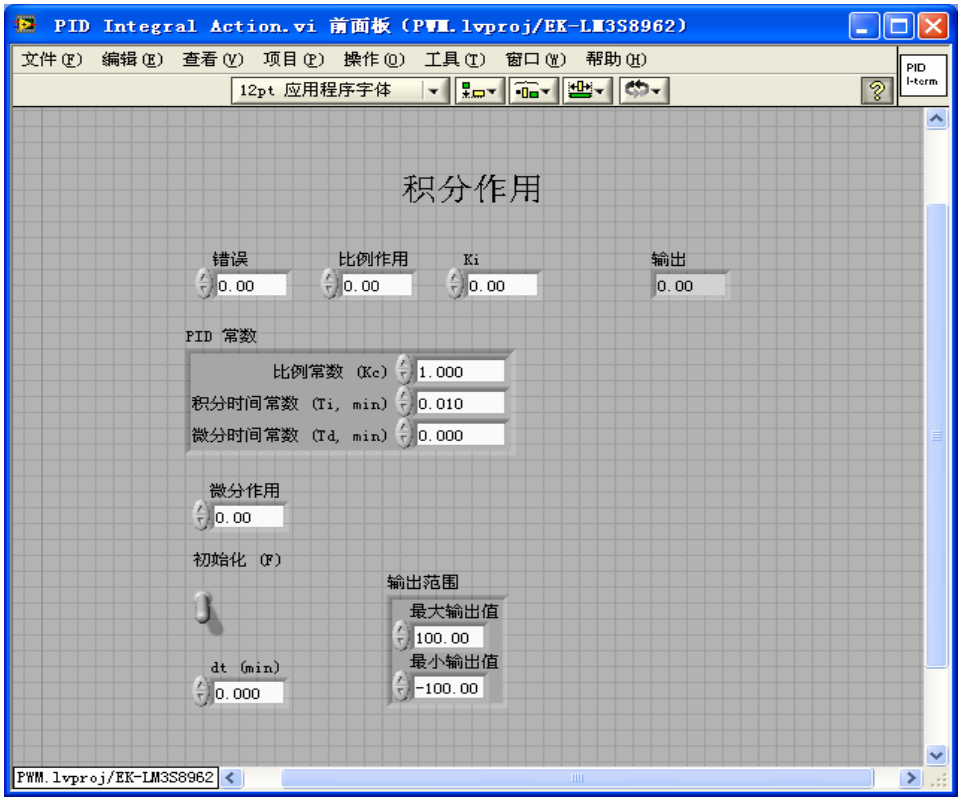


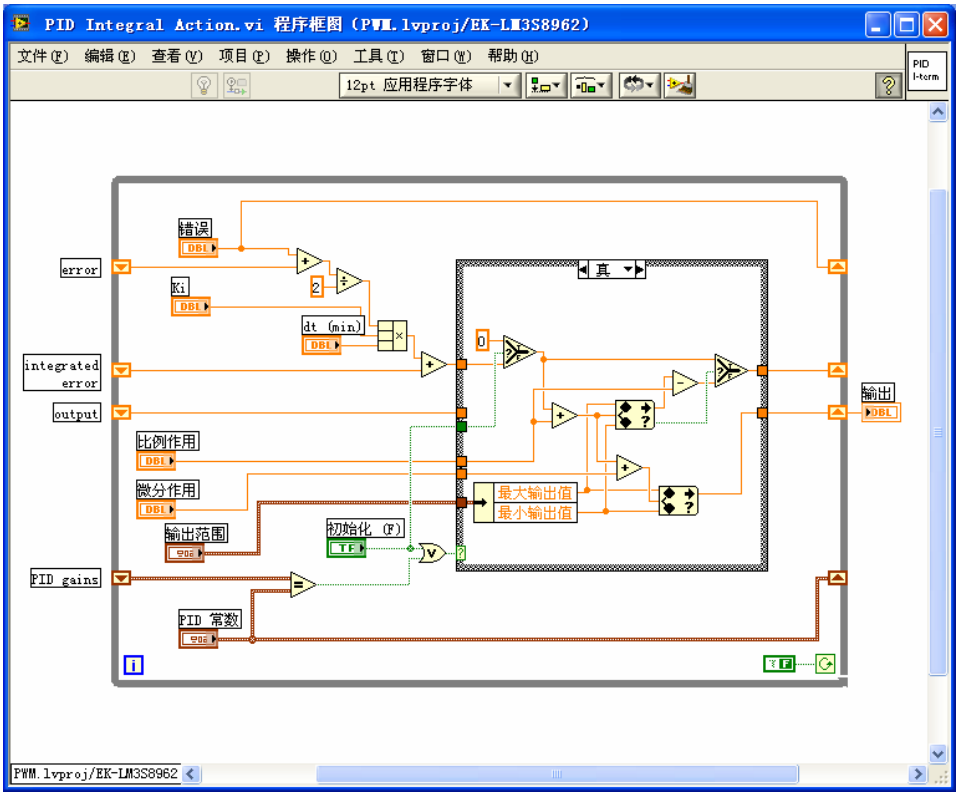

PID Derivative Action.vi 程序框图 (PWM.lvproj/EK-LM3S8962)
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
12pt 应用程序字体
PID
D-term
初始化 (F)
TF
变量处理
DBL
Kd
DBL
dt (min)
DBL
假
假
dt = 0 min
>0
微分作用
DBL
PWM.lvproj/EK-LM3S8962

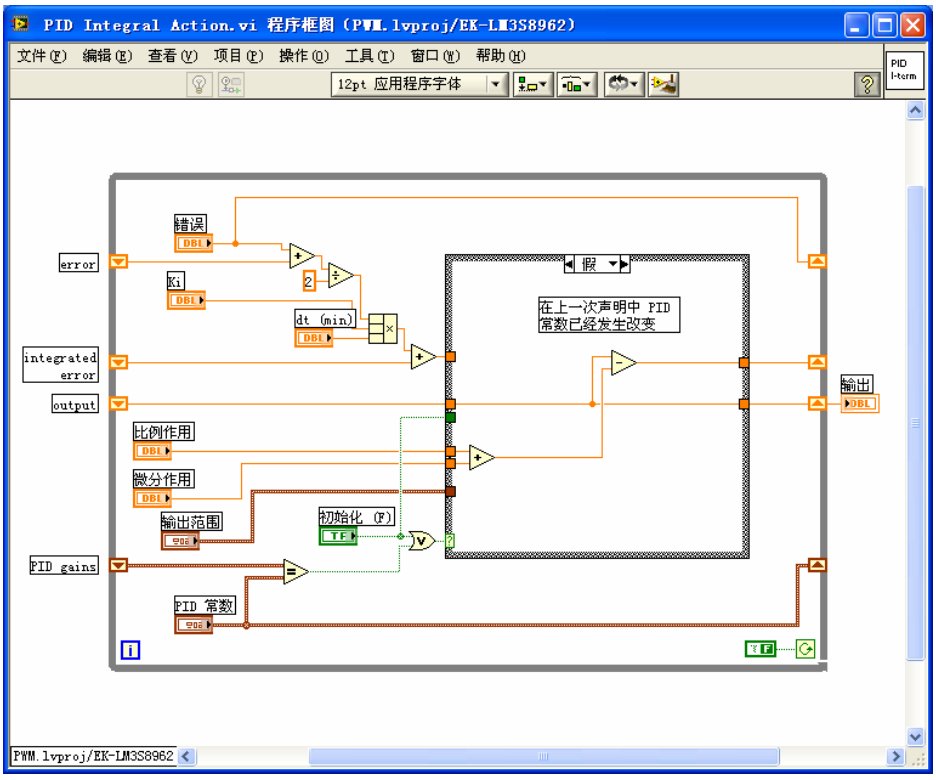

PID Derivative Action.vi 程序框图 (PWM.lvproj/EK-LM3S8962)
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
12pt 应用程序字体
PID
D-term
初始化 (F)
TF
变量处理
DBL
Kd
DBL
dt (min)
DBL
假
真
>0
微分作用
DBL
PWM.lvproj/EK-LM3S8962



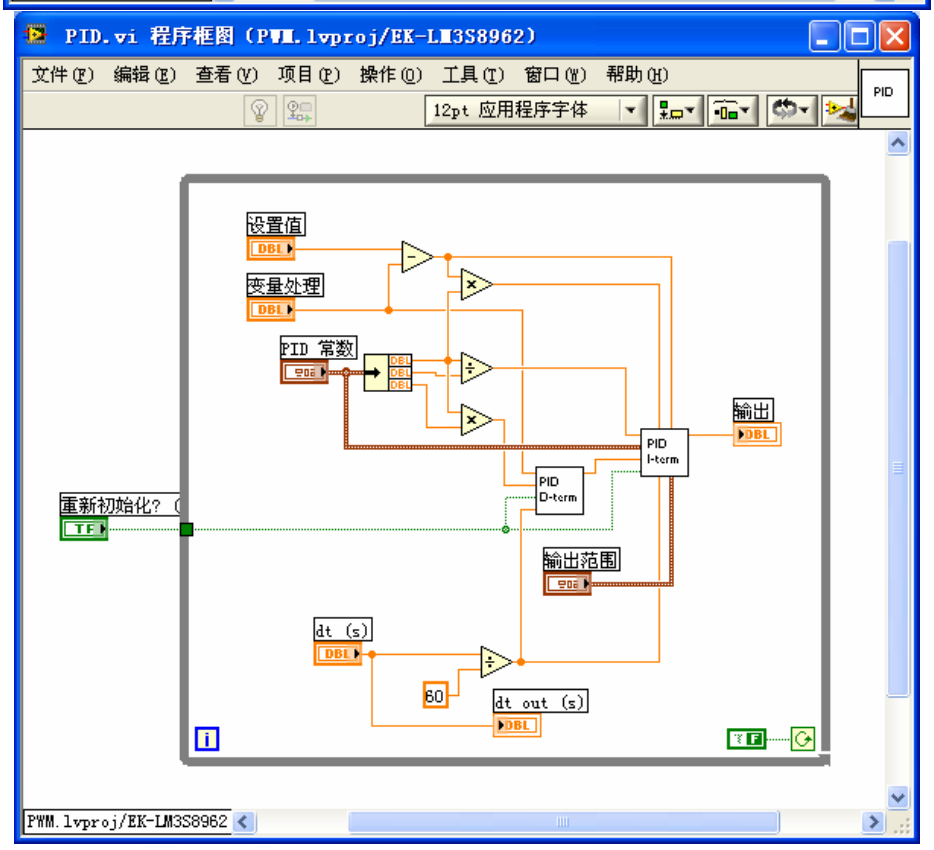
5、 参考下图建立子 VI **PID Integral Action.vi**前面板和程序框图；



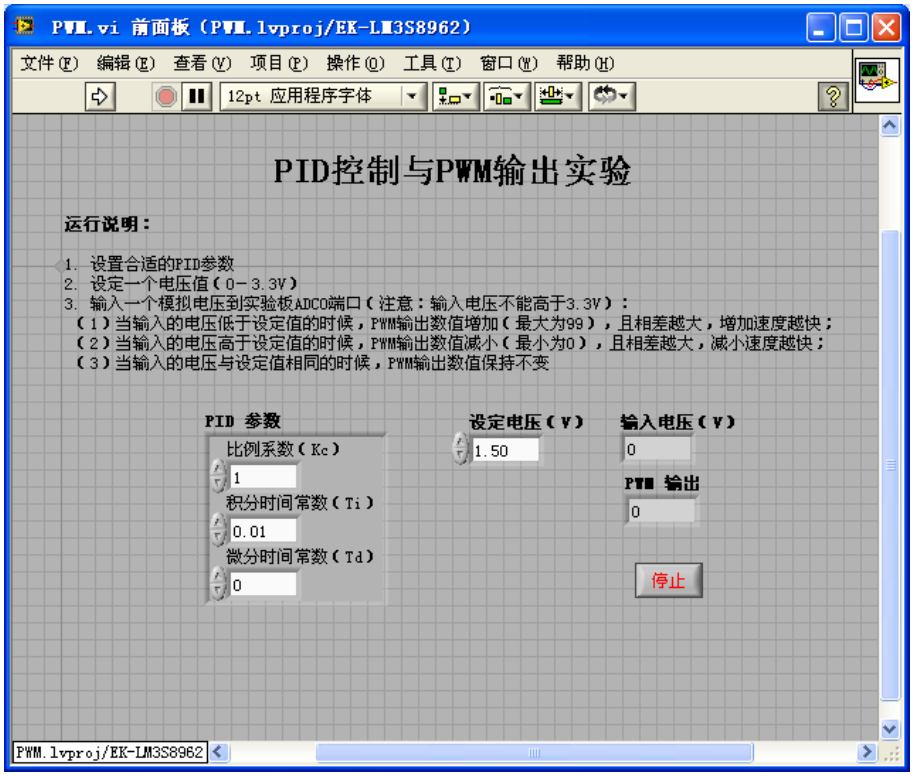


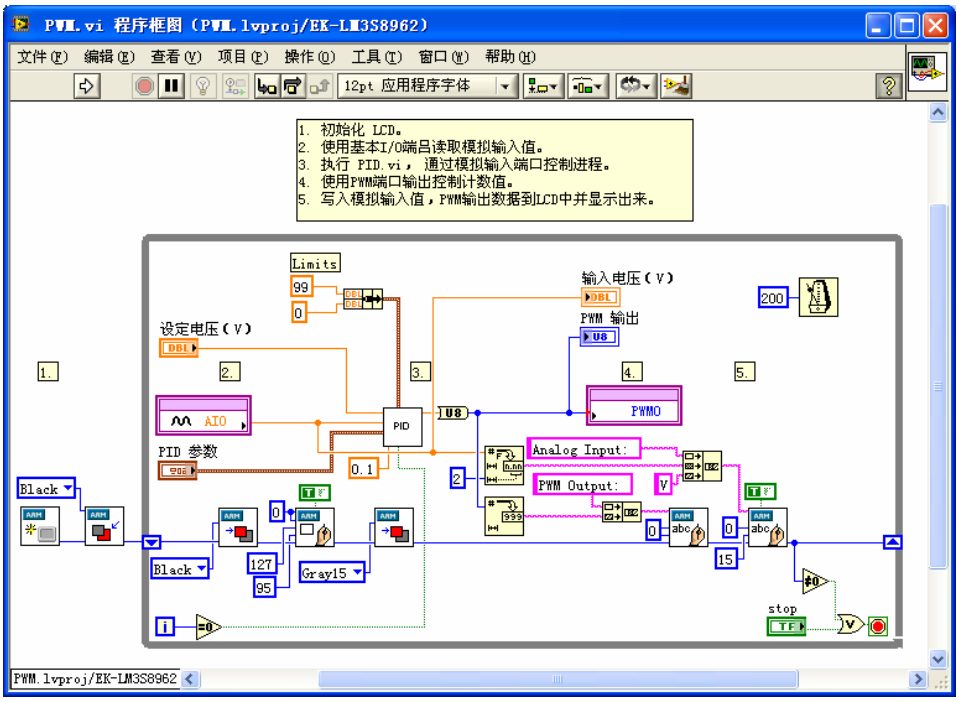


6、 参考下图建立子 VI **PID.vi**前面板和程序框图；



7、 参考下图建立主程序 **PWM.vi**的前面板和程序框图：





8、 单击运行按扭，观察处理器状态和实验板的运行；调节输入电压观察实验板上的 OLED 显示器。

9、 结果：模拟输入电压 AI0 低于设定值的时候，PWM 输出值增加（最大为 99），且相差越多，变化越快；模拟输入电压 AI0 高于设定值的时候，PWM 输出值减小（最小为 0），同样是相差越多，变化越快；模拟输入电压 AI0 与设定值相同的时候，PWM 输出值固定不变。

实验 6：TCP 通信实验

一、 实验目的

了解 TCP 通信原理。

二、 实验要求

通过这个实验，要求能够比较熟练的掌握 LabVIEW 语言中各个 TCP 控件的使用，了解 TCP 协议规程。

三、 实验原理

传输控制协议（Transmission Control Protocol，TCP）是一种面向连接的、可靠的、基于字节流的运输层通信协议 TCP 建立连接之后，通信双方都同时可以进行数据的传输，因此 TCP 是全双工通信。

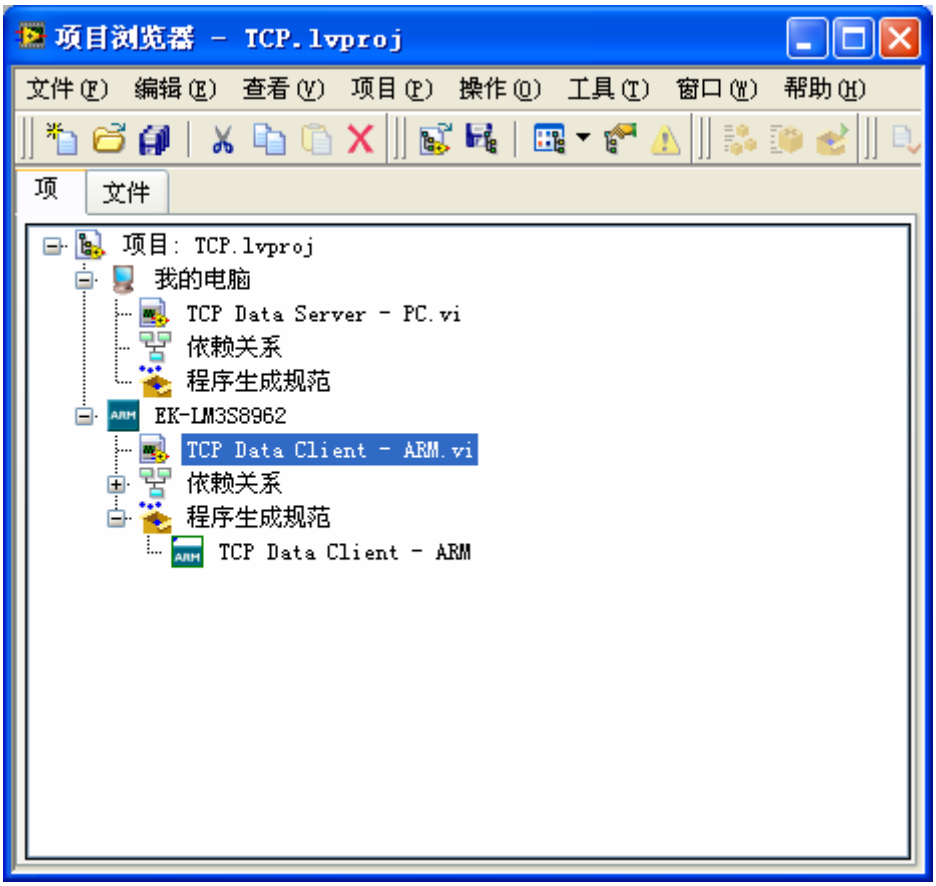
四、 实验环境

计算机、RealVIEW MDK 集成开发环境、LabVIEW、LabVIEW Embedded Module for ARM Controllers、ULINK2 仿真器、实验板。

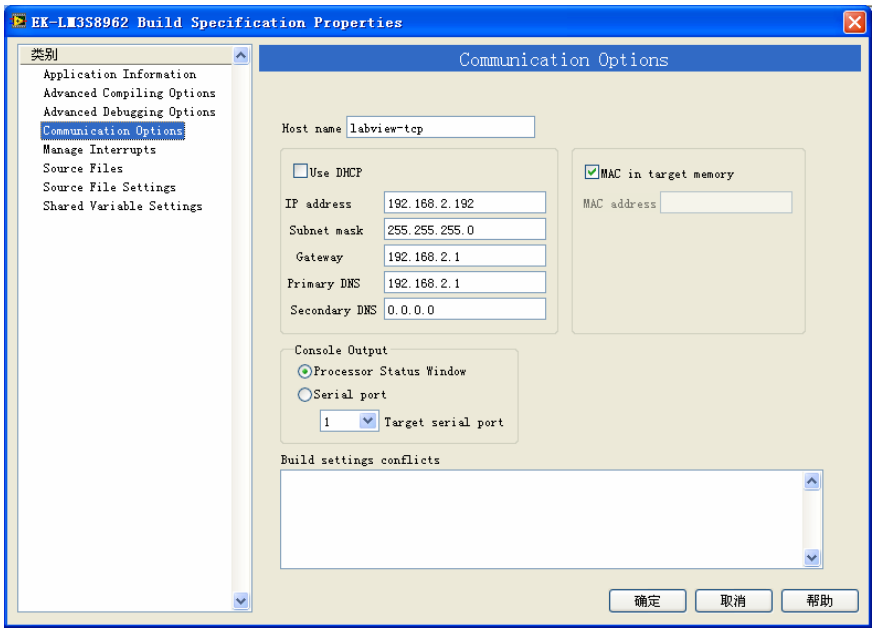
五、 实验步骤

1、 连接好开发板与计算机，用网线将开发板与计算机相连；

2、 参考下图，建立工程：“**TCP. Lvproj**”：



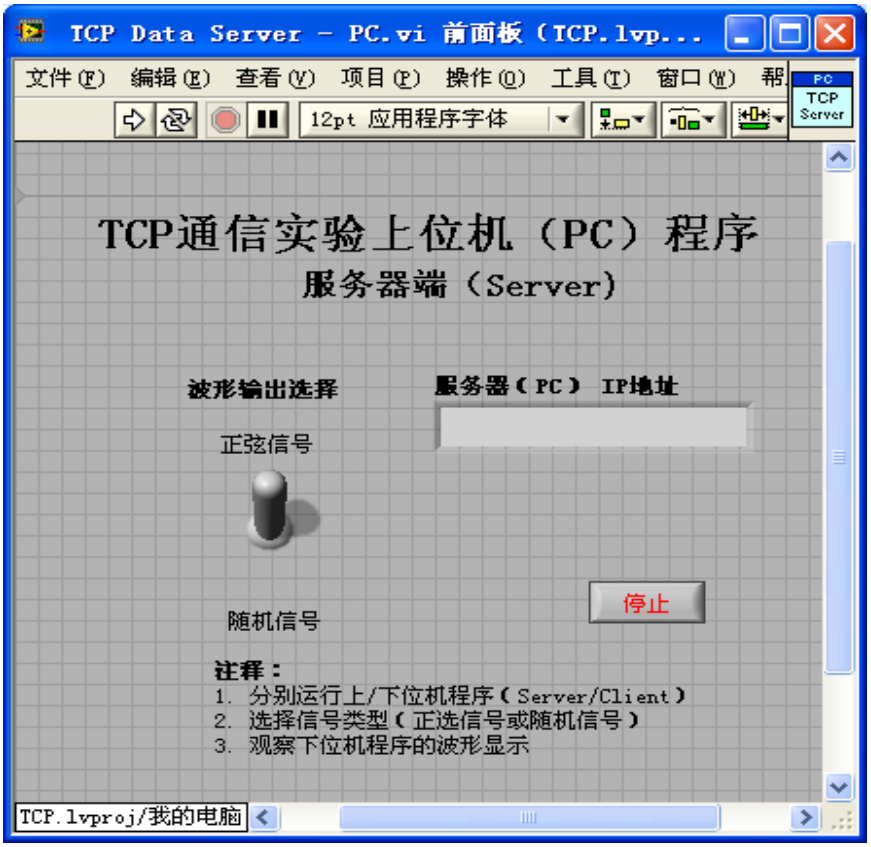
3、参数设置：在 **EK-LM3S8962 Build Specification Properties** 页面中的 **Communication Options**页中去掉 **Use HDCP**键入固定 IP 地址（参见下图，实验板 IP 地址默认设置是 192.168.2.192）：

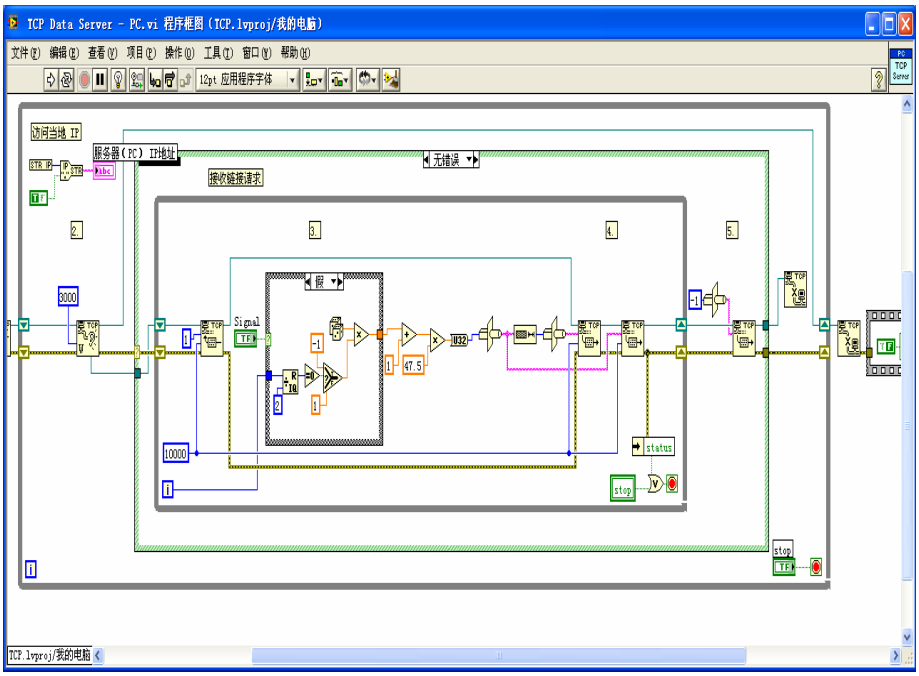


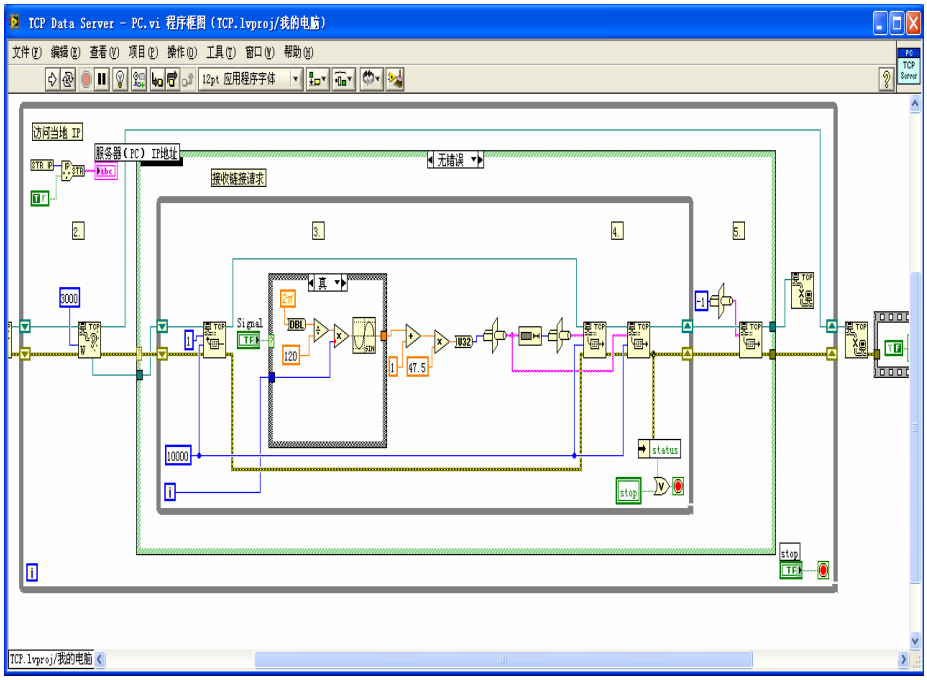
4、将上位机（PC）的 IP 地址设置在同一网段内，如：192.168.2.114；

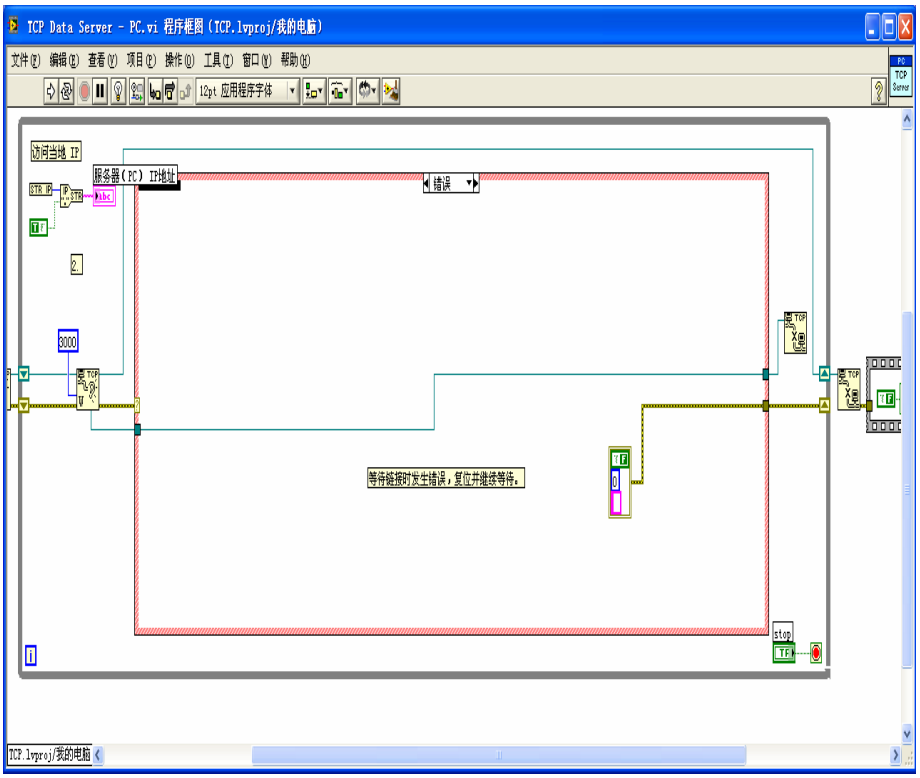
5、参考下图，建立上位机（PC）程序 **TCP Data Server - PC.vi** （程序框图中只包含

主要框架部分）：

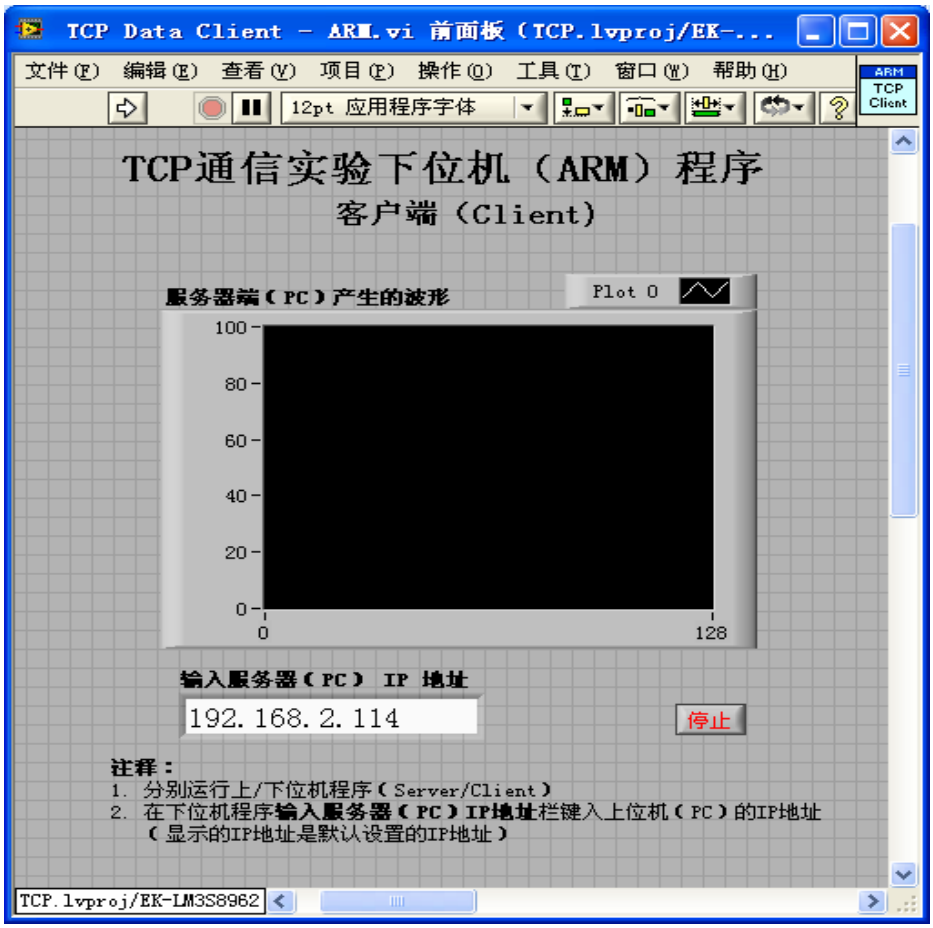

TCP Data Server - PC.vi 前面板 (TCP.lvp...
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮
12pt 应用程序字体
PC
TCP
Server
TCP通信实验上位机（PC）程序
服务器端（Server）
波形输出选择
正弦信号
随机信号
服务器（PC） IP地址
停止
注释：
1. 分别运行上/下位机程序（Server/Client）
2. 选择信号类型（正选信号或随机信号）
3. 观察下位机程序的波形显示
TCP.lvproj/我的电脑

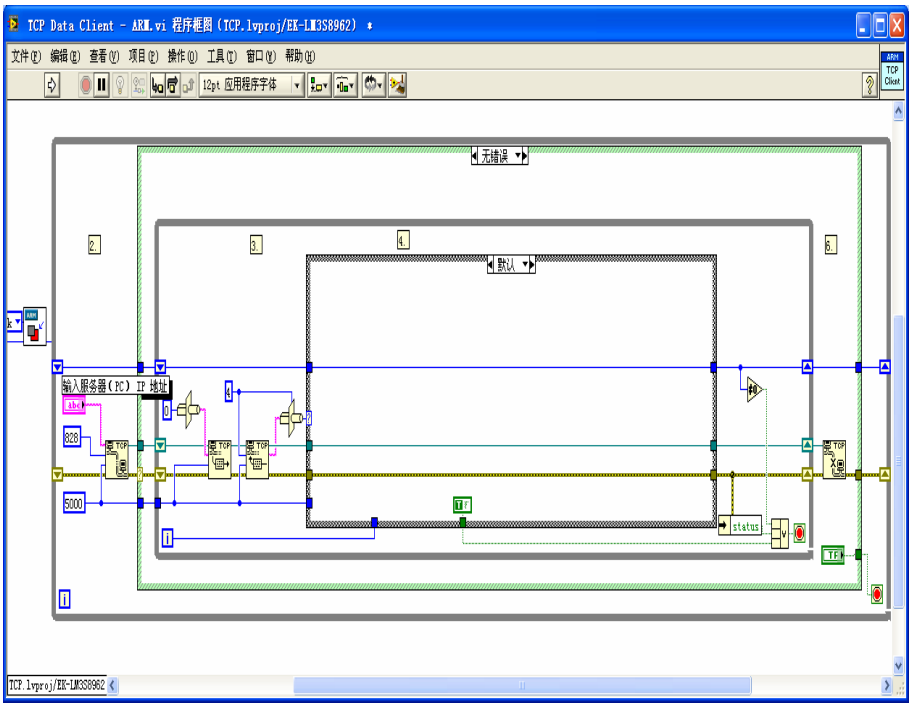

TCP Data Server - PC.vi 程序框图 (TCP.lvproj/我的电脑)
访问当地 IP
服务器(PC) IP地址
接收链接请求
无错误
Signal
status
stop
TCP.lvproj/我的电脑

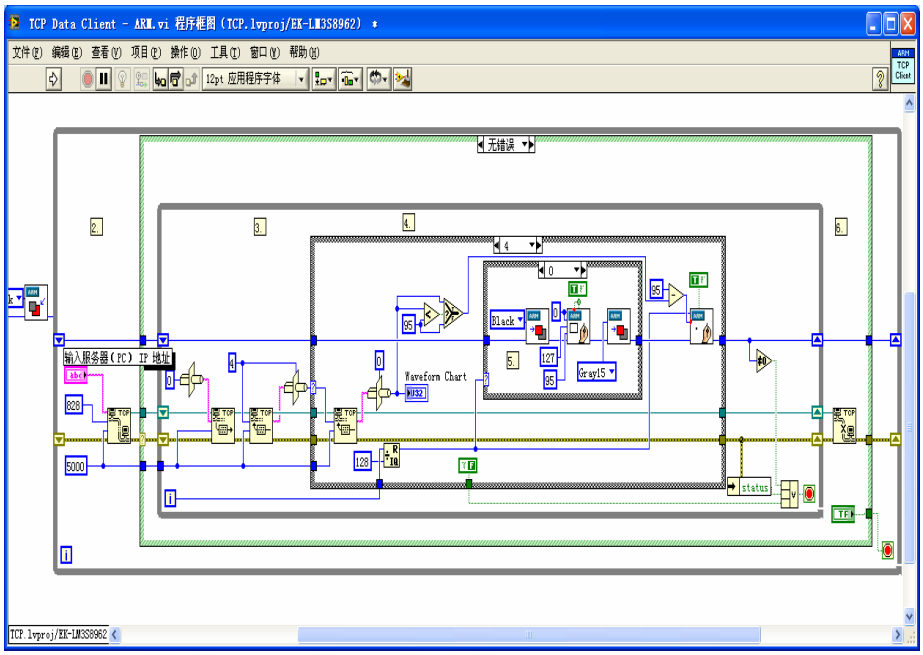


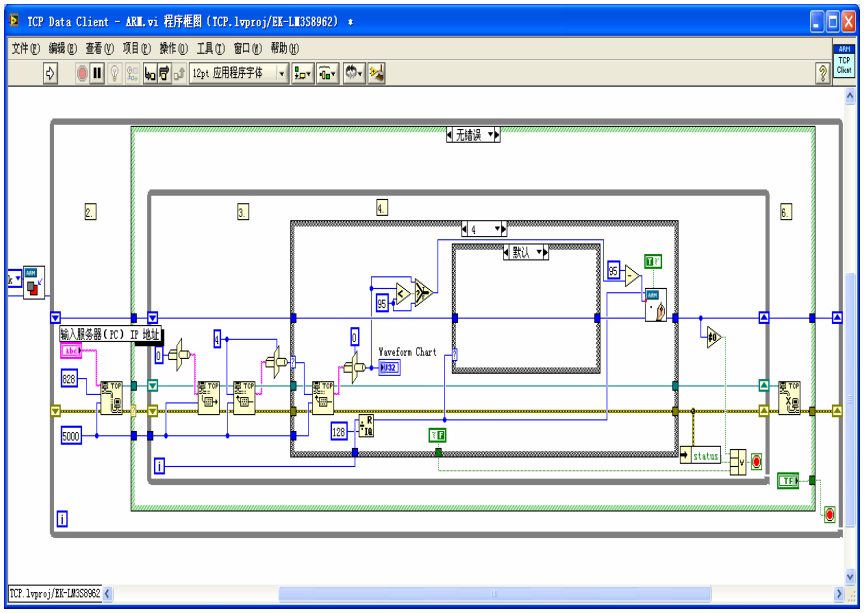


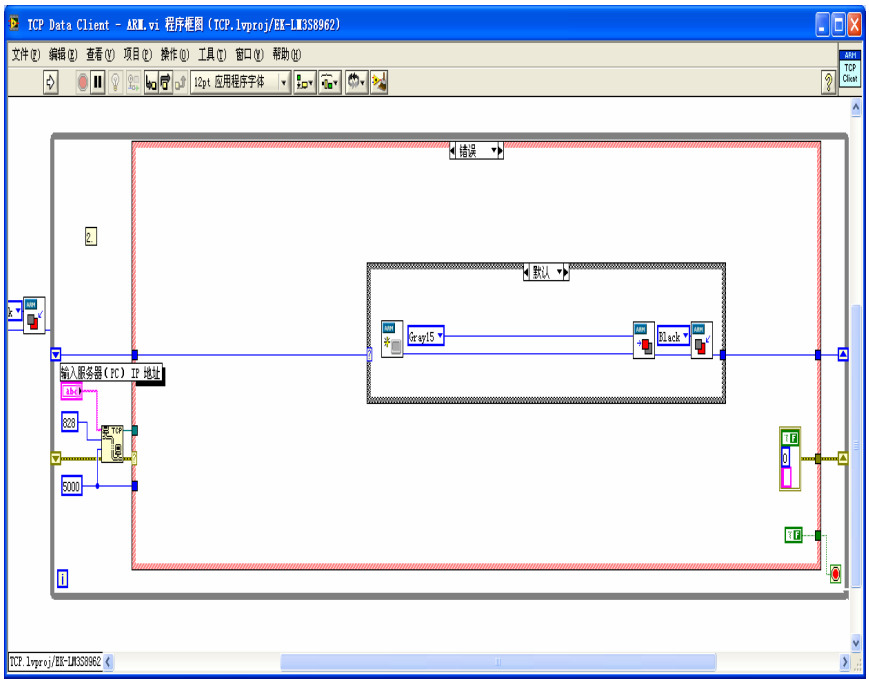
6、 参考下图，建立下位机（ARM）程序 **TCP Data Client - ARM.vi**：


TCP Data Client - ARM.vi 前面板 (TCP.lvproj/EK-...
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
12pt 应用程序字体
ARM
TCP Client
TCP通信实验下位机（ARM）程序
客户端（Client）
服务器端（PC）产生的波形
Plot 0
100
80
60
40
20
0
0
128
输入服务器（PC） IP 地址
192. 168. 2. 114
停止
注释：
1. 分别运行上/下位机程序（Server/Client）
2. 在下位机程序输入服务器（PC）IP地址栏键入上位机（PC）的IP地址
（显示的IP地址是默认设置的IP地址）
TCP.lvproj/EK-LM3S8962

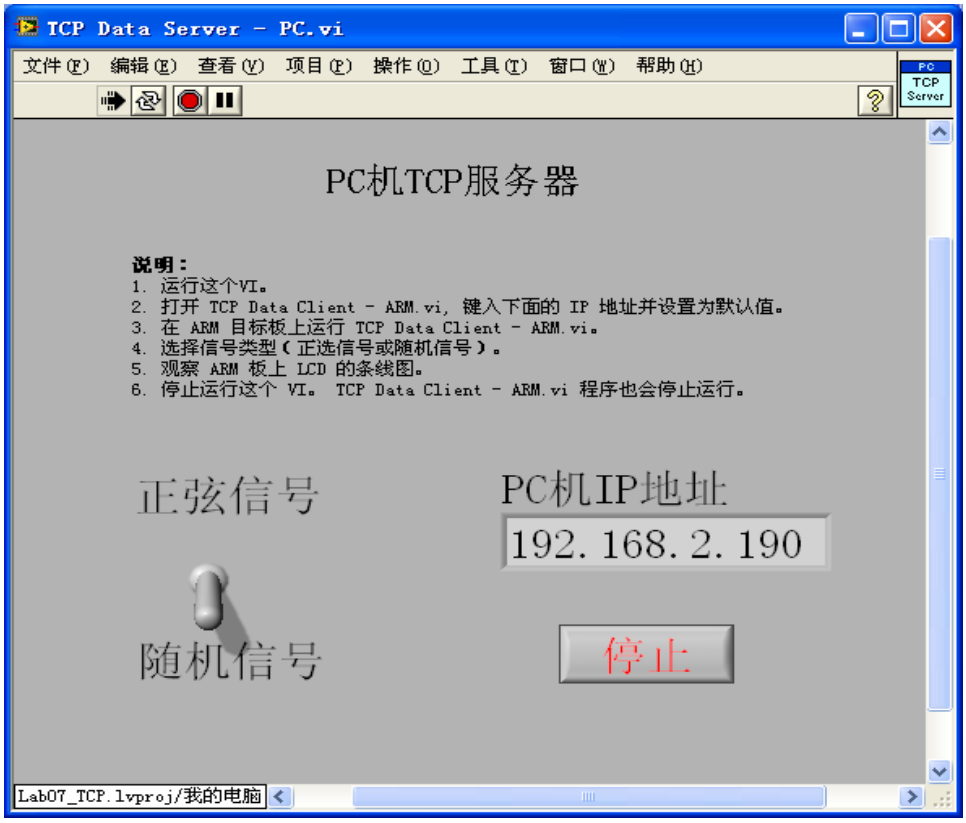

TCP Data Client - ARM.vi 程序框图 (TCP.lvproj/EK-LM3S8962)
无错误
默认
输入服务器(PC) IP 地址
status
TCP.lvproj/EK-LM3S8962


TCP Data Client - ARM.vi 程序框图 (TCP.lvproj/EK-LM3S8962) *
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
12pt 应用程序字体
无错误
输入服务器(PC) IP 地址
Waveform Chart
status
TCP.lvproj/EK-LM3S8962


TCP Data Client - ARM.vi 程序框图 (TCP.lvproj/EK-LM3S8962) *
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
12pt 应用程序字体
无错误
默认
输入服务器(PC) IP 地址
Waveform Chart
status
TCP.lvproj/EK-LM3S8962



7、 在 PC 机上运行 **TCP Data Server - PC.vi**：



8、 在下位机程序 TCP Data Client - ARM.vi 地址栏中输入上位机（PC）IP 地址，然后点击运行按键，下位机程序将自动编译、链接、下载到实验板上并运行。

9、 通过上位机程序选择信号类型，在下位机前面板及实验板 OLED 显示屏上观察波形显示。

实验 7：人机交互游戏实验

一、 实验目的

提高综合编程能力，练习综合使用实验板的各个模块。

二、 实验要求

通过编写一个游戏程序，将 **选择开关** 、**导航开关** 和 **OLED 显示屏** 综合使用。

三、 实验原理

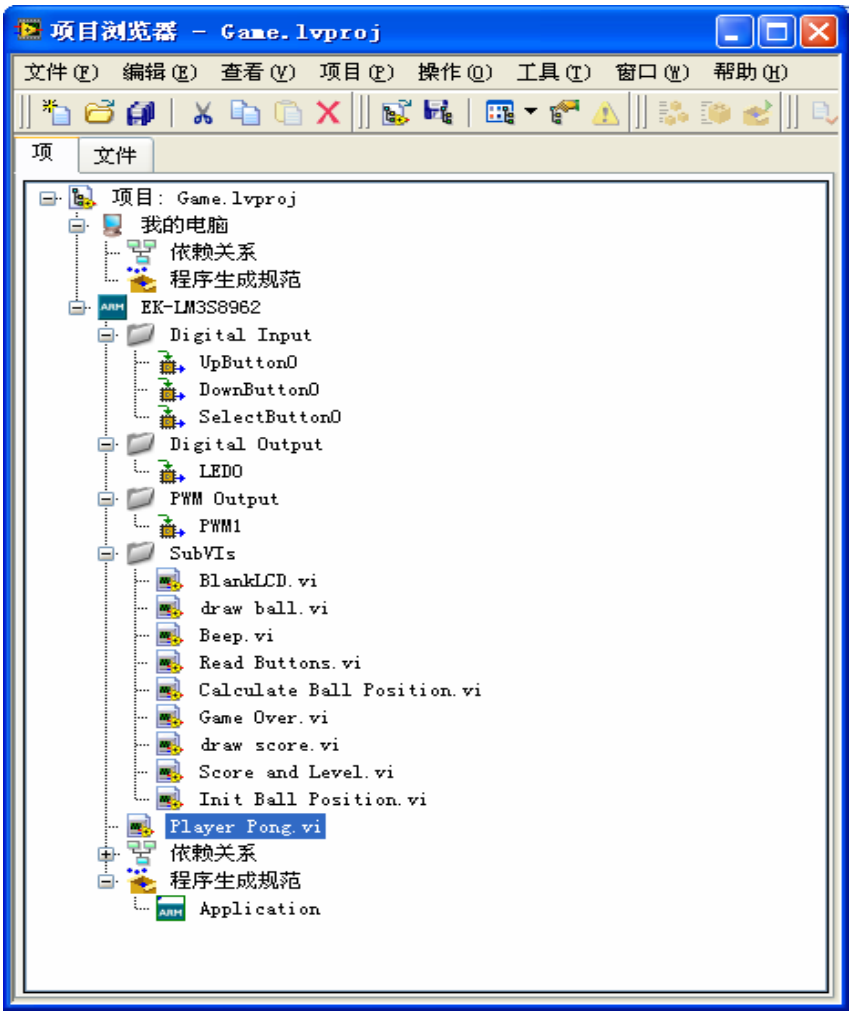
利用 **选择开关** 开始游戏，**导航开关** 的上下按钮移动球拍，在 **OLED 显示屏** 上显示小球的运动。

四、 实验环境

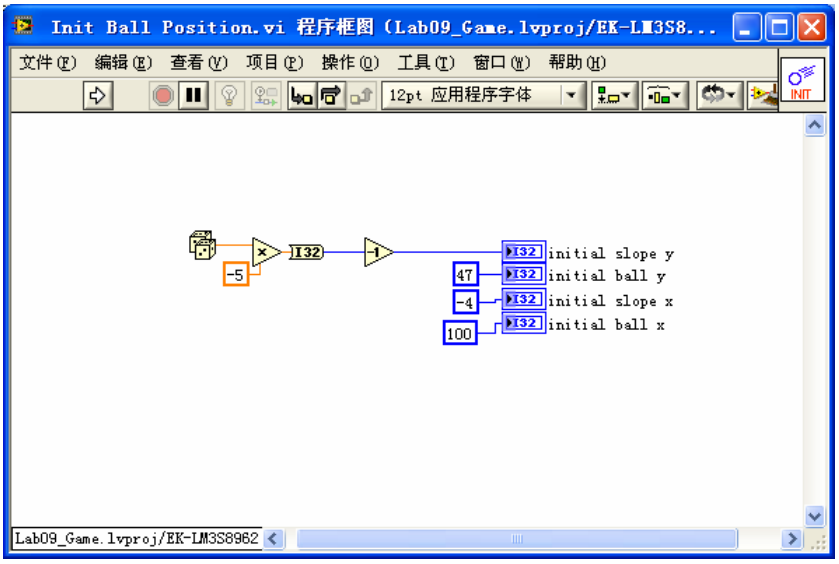
计算机、RealVIEW MDK 集成开发环境、LabVIEW、LabVIEW Embedded Module for ARM Controllers、ULINK2 仿真器、实验板。

五、 实验步骤

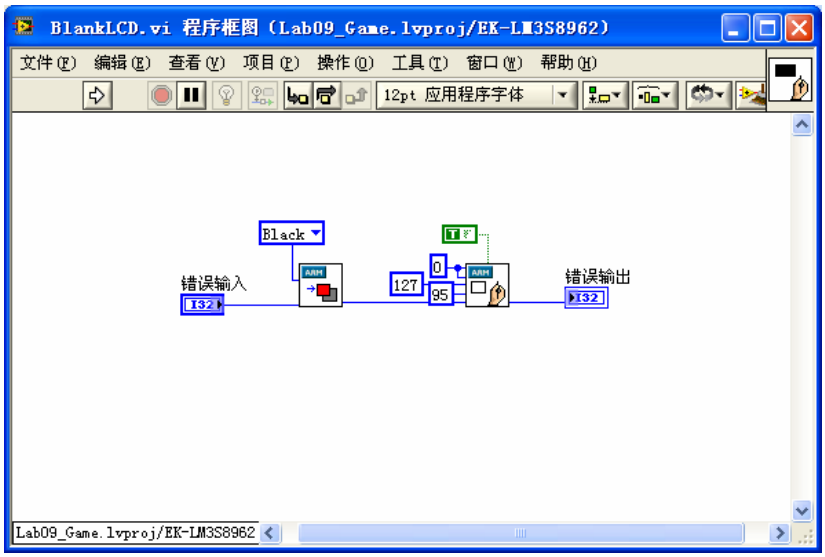
1、 参考下图建立工程：



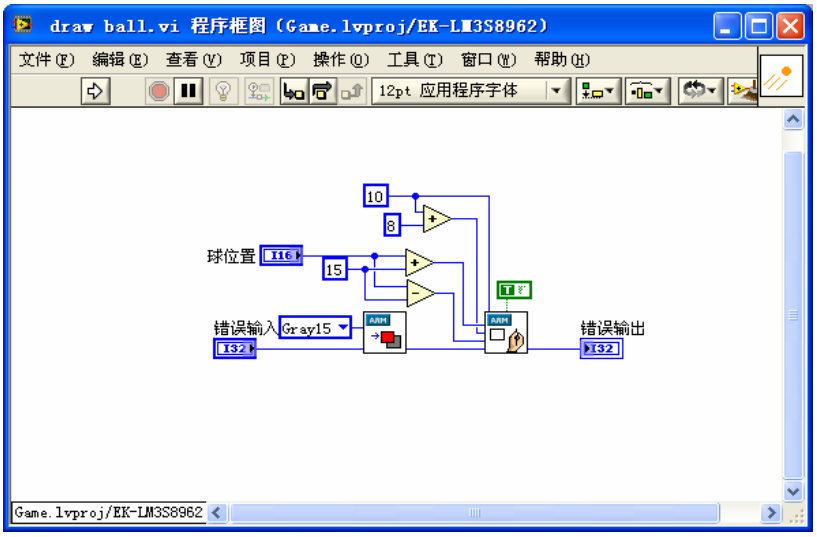
2、 参考下图建立子 VI **Init Ball Position.vi**:



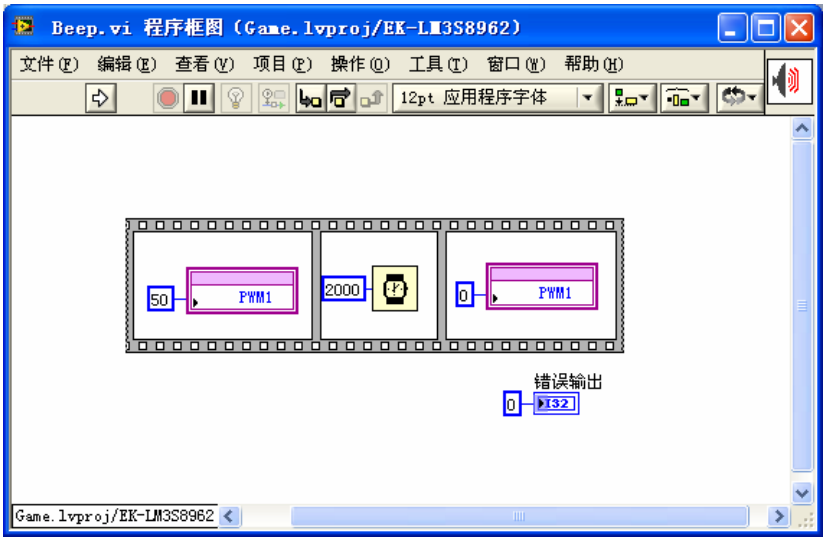
3、 参考下图建立子 VI **BlankOLED.vi**:



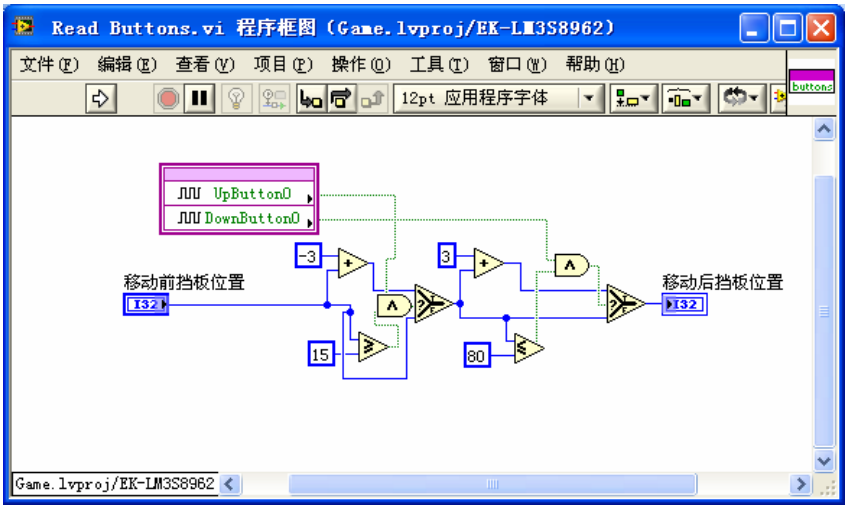
4、参考下图建立子 VI **draw ball.vi**：



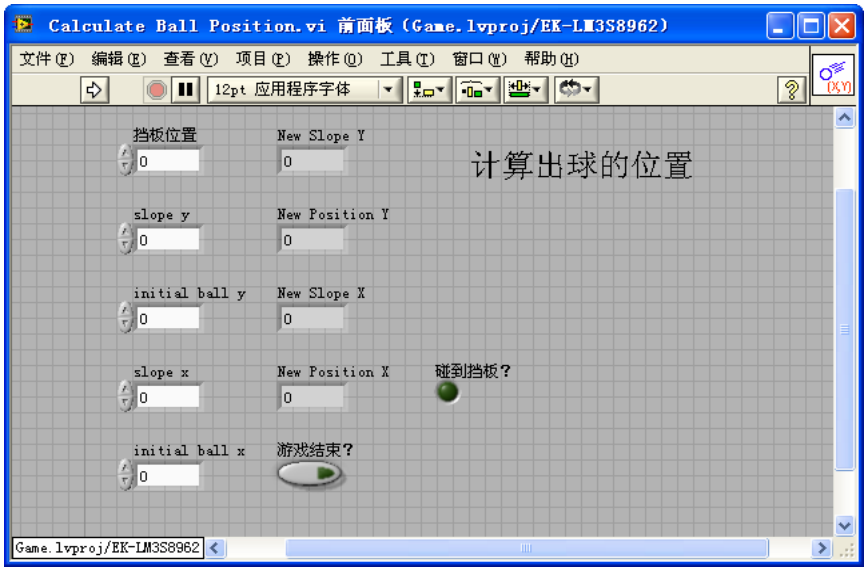
5、参考下图建立子 VI **Beep.vi**：

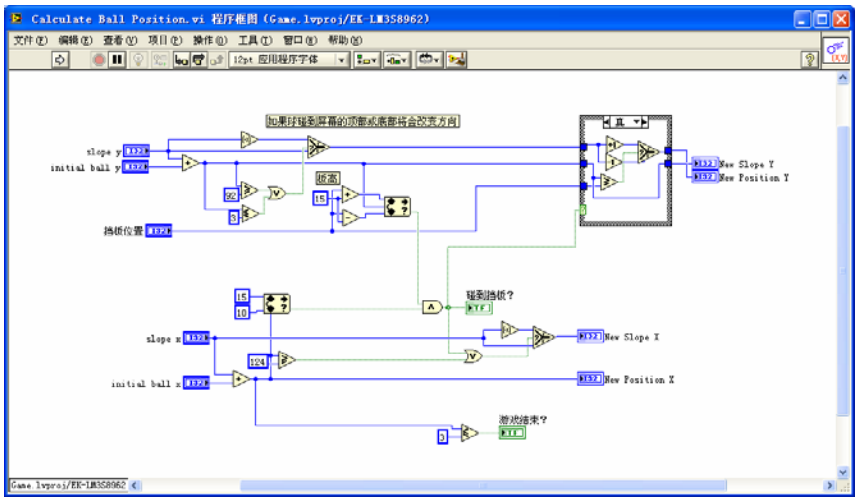


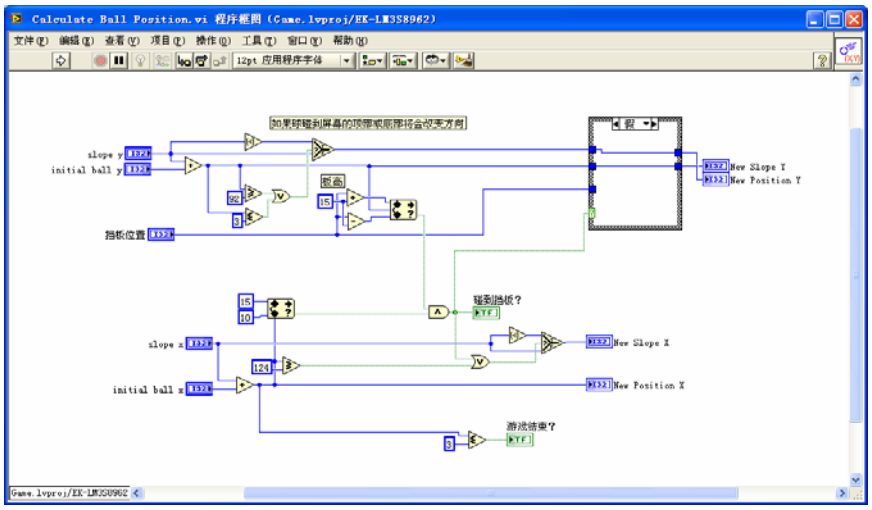
6、 参考下图建立子 VI **Read Buttons.vi**：



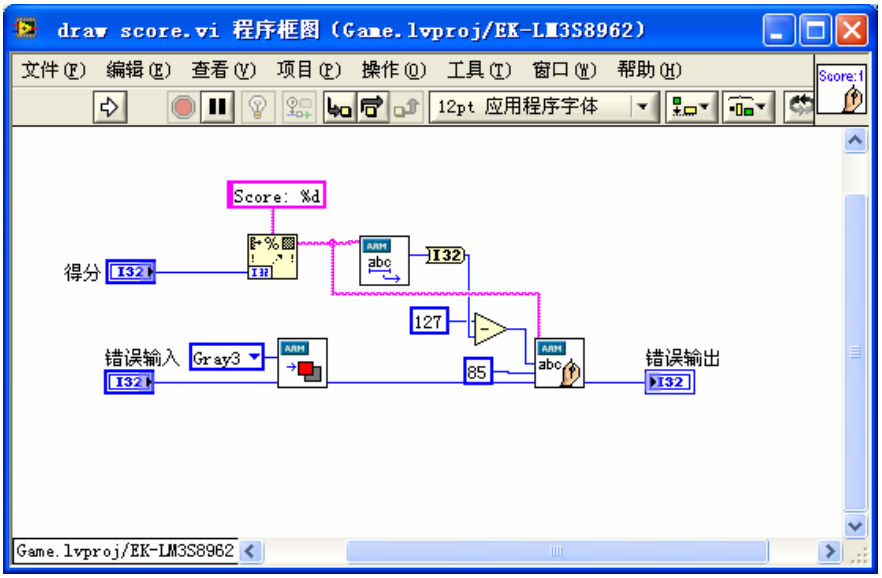
7、 参考下图建立子 VI **Calculate Ball Position.vi**：



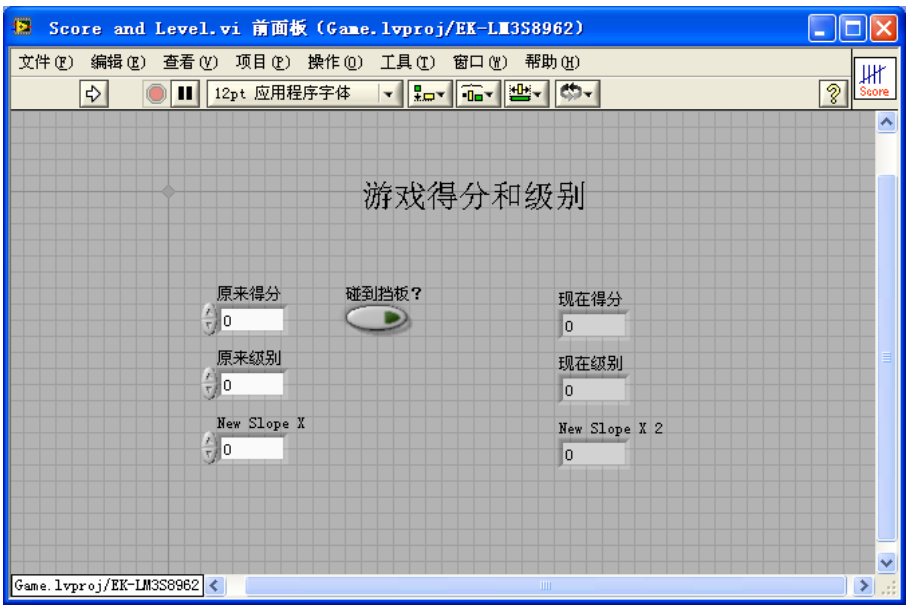


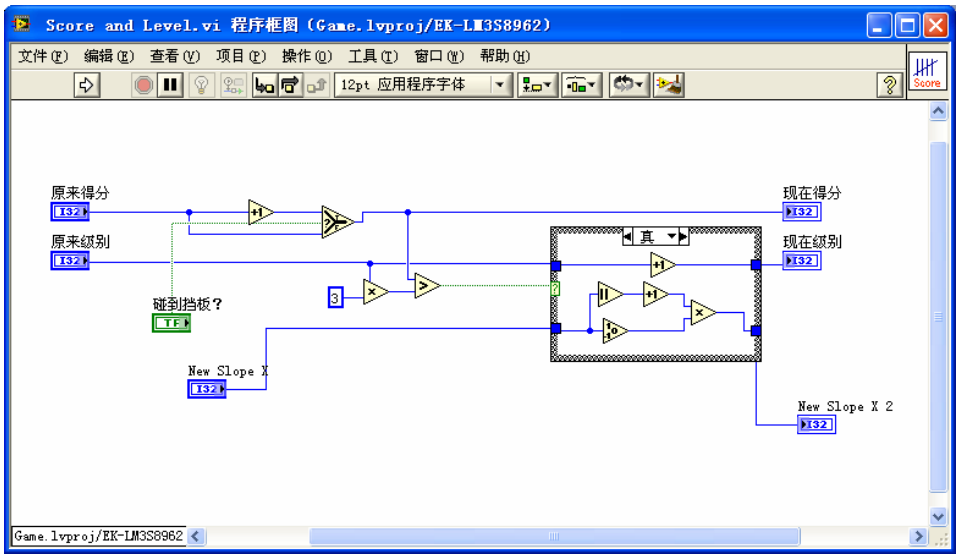


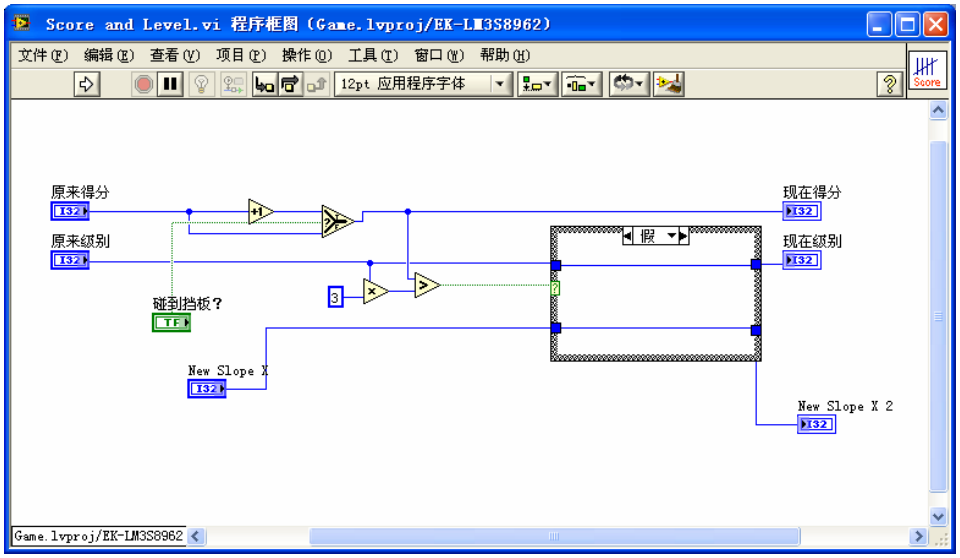
8、参考下图建立子 VI　**draw score.vi**：



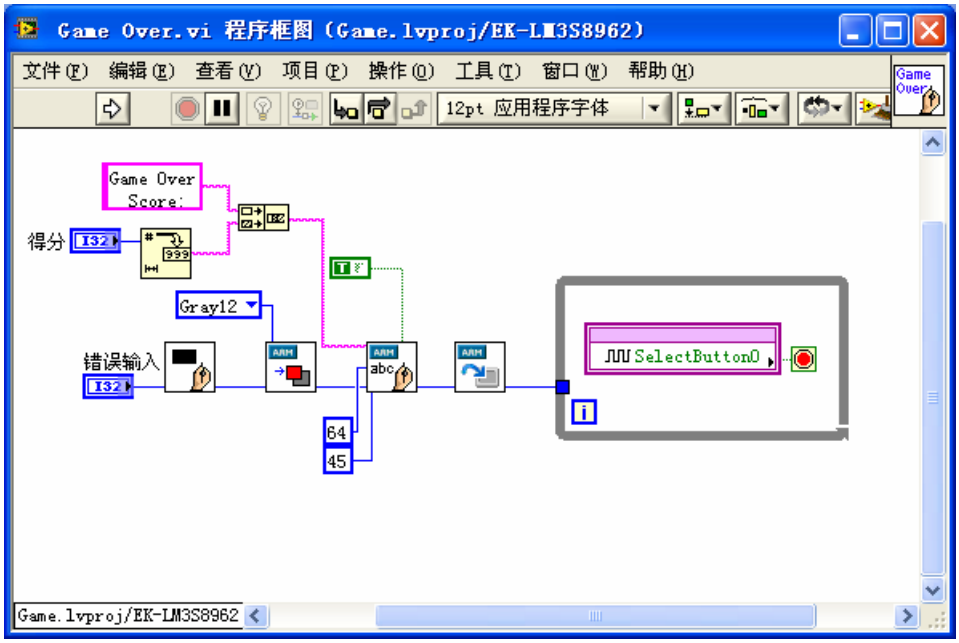
9、参考下图建立子 VI　**Score and Level.vi**：


Score and Level.vi 前面板 (Game.lvproj/EK-LM3S8962)
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
12pt 应用程序字体
Score
游戏得分和级别
原来得分
碰到挡板?
现在得分
原来级别
现在级别
New Slope X
New Slope X 2
Game.lvproj/EK-LM3S8962

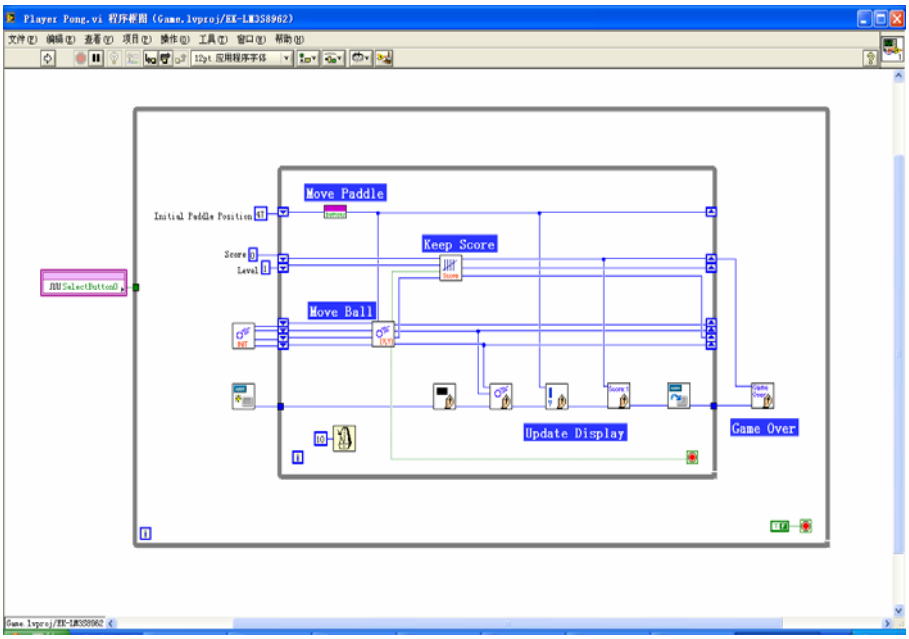

Score and Level.vi 程序框图 (Game.lvproj/EK-LM3S8962)
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
12pt 应用程序字体
Score
原来得分
原来级别
碰到挡板?
New Slope X
现在得分
现在级别
New Slope X 2
真
Game.lvproj/EK-LM3S8962


Score and Level.vi 程序框图 (Game.lvproj/EK-LM3S8962)
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
12pt 应用程序字体
Score
原来得分
原来级别
碰到挡板?
New Slope X
现在得分
现在级别
New Slope X 2
假
Game.lvproj/EK-LM3S8962

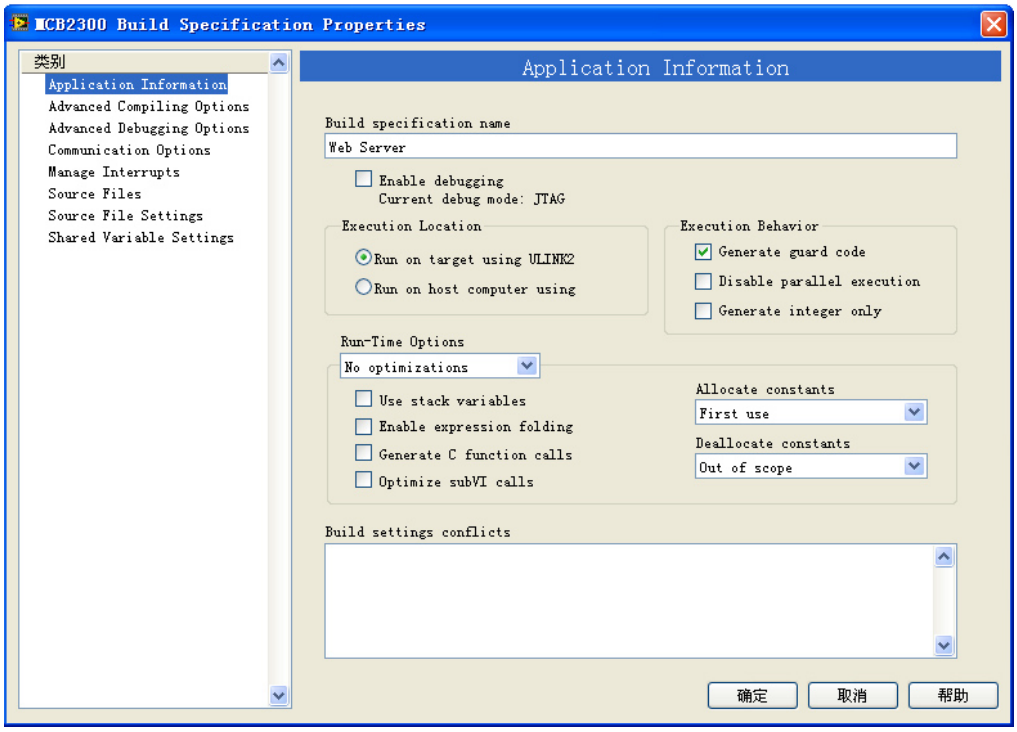
10、 参考下图建立子 VI **Game Over.vi**：



11、 参考下图建立主程序 **Player Pong.vi**：



12、 重要参数设置：鼠标右键点击在项目浏览器中的 **程序生成规范** 下的 **Application**，选择 **属性**，在 **EK-3S8962 Build Specification** 页面中的 **Application Information** 中将 **Enable debugging** 选项去掉（否则程序运行不正确）



13、 运行程序，待程序下载到开发板后，按开发板上的 **选择按钮（SELECT）** 开始游戏，按 **上下方向按键** 上下移动球拍将小球击回；游戏结束后，可以按下 **选择按钮（SELECT）** 重新开始游戏。