TF－VIA

32 位双 CPU 图形化开发嵌入式微控制器实验系统

李东江

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。

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

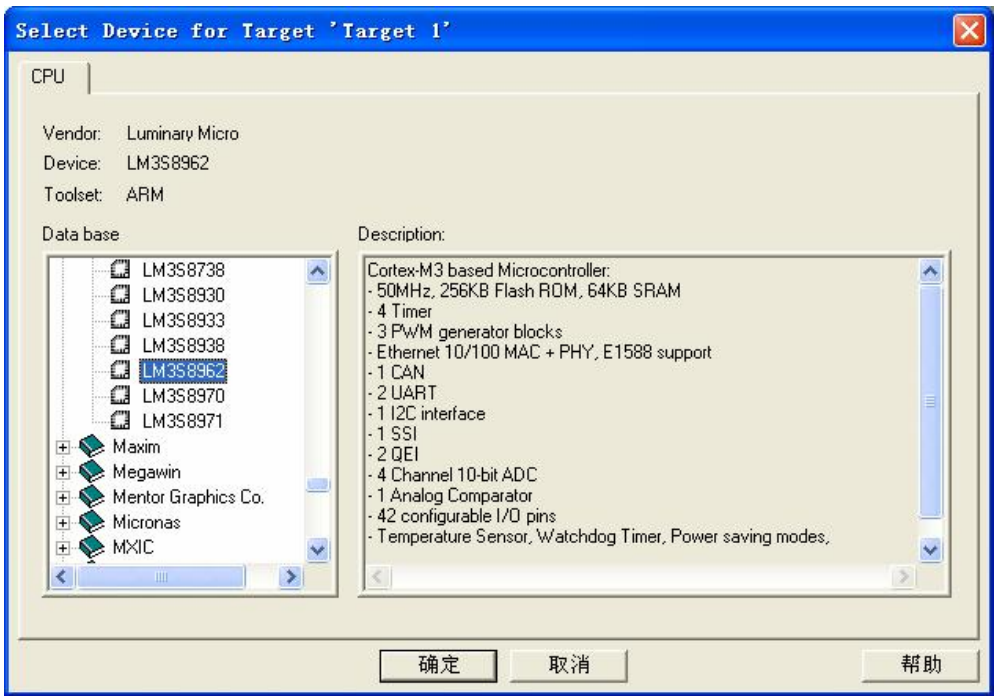
计算机、RealVIEW MDK 集成开发环境、ULINK2适配器、实验箱。

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。

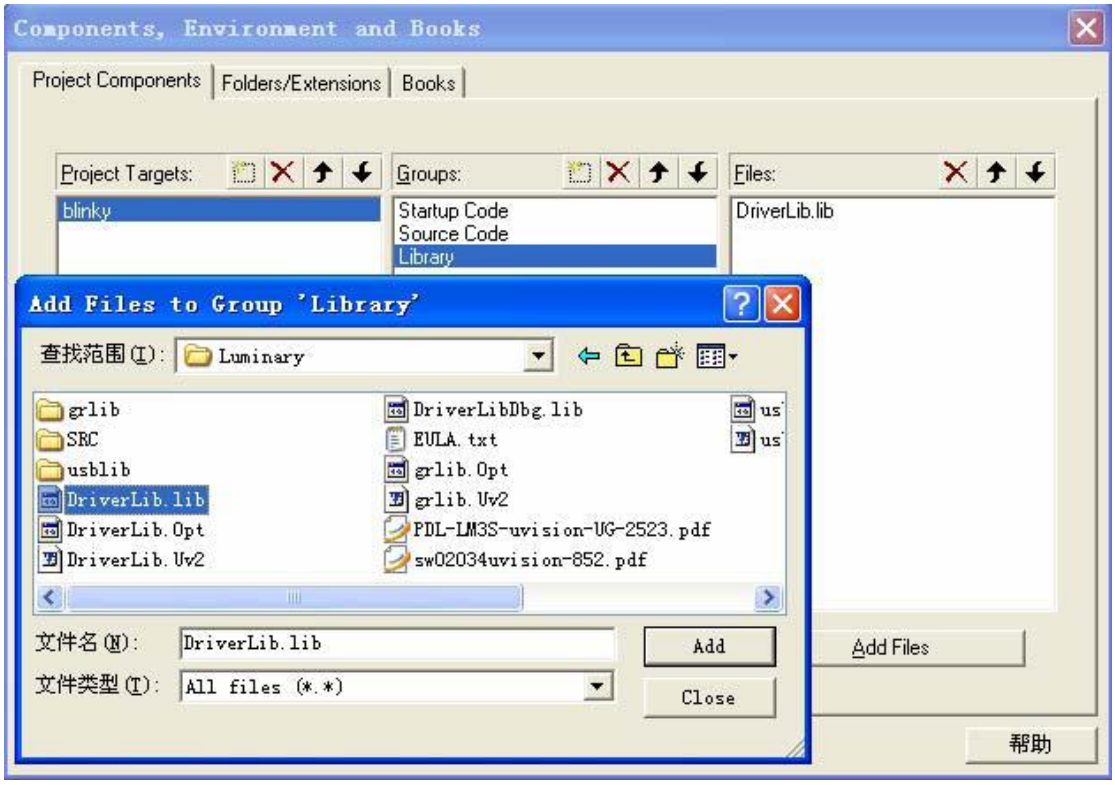
在uVision主界面在选择Project->New Project菜单项，打开一个标准对话框，输入Blinky（建议对每个新建工程使用独立的文件夹，这里建立一个新的文件夹Keil01_Blinky）。单击保存，出现选择处理器对话框：Select Device倫for Target倫‘Target 1’，选择Lunminary公司的LM3S8962控制器，如图所示。单击“确定”后，在弹出的对话框中单击“是”便可将启动代码加入工程。

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

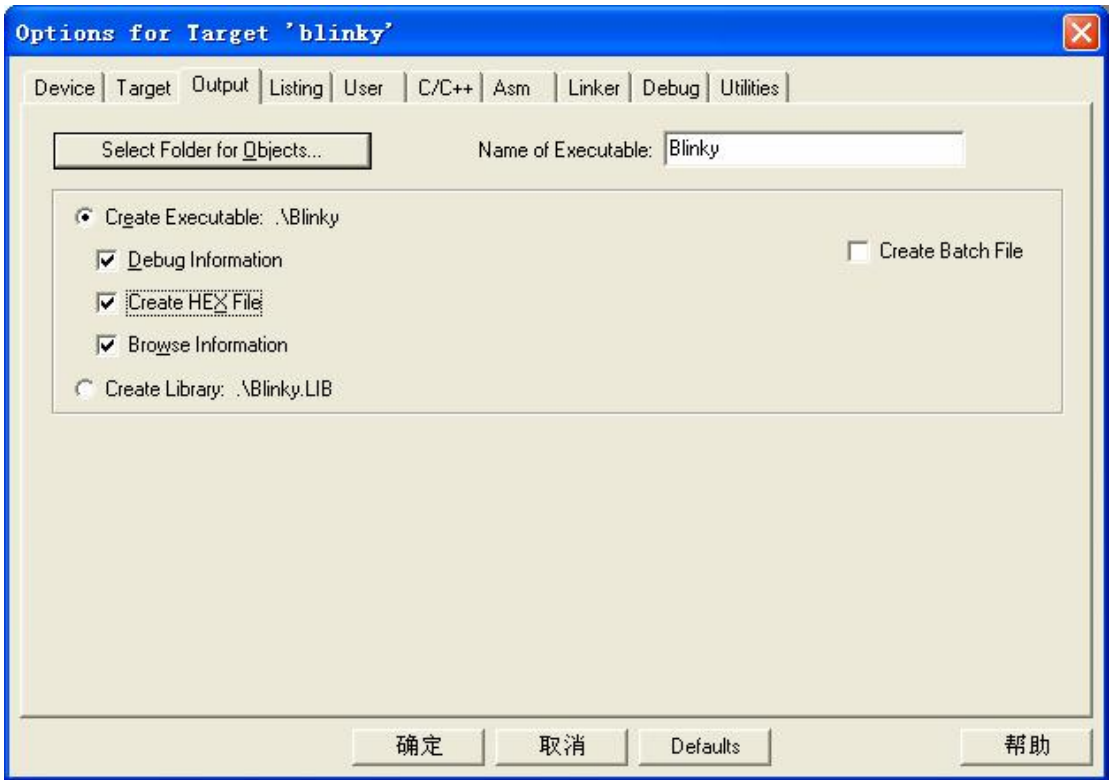
选择菜单项Project->Manage->Components，Environment，Books...在Project Components倫页中，可以创建文件夹，便于管理文件。在Groups倫区域中，点击刚创建的文件夹Library，再在Files区域中点击Add Files，在弹出的对话框中在，指定目录下（如C:\Keil\ARM\RV31\LIB\Luminary）选择DriverLib.lib，点击Add，关闭对话框。如图所示：



4、编译链接工程、调试程序

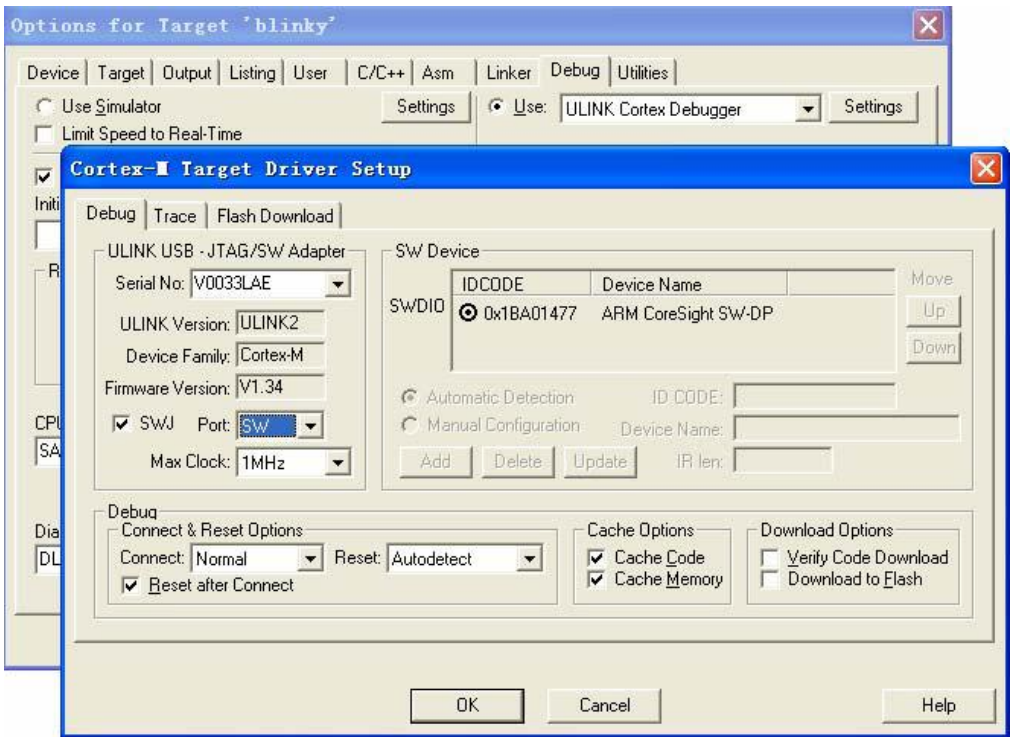
5、建立HEX文件

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倫页选择Create倫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倫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 10：小键盘实验 2

一、实验目的

了解实验箱的硬件环境；

熟悉小键盘的使用。

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。

七、实验步骤

参考实验一。

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 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

了解实验箱的硬件环境；

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

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）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

七、实验步骤

参考实验一。

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 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

了解实验箱的硬件环境；

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。

七、实验步骤

参考实验一。

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 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。

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。

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

i. μCOSII的简介

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

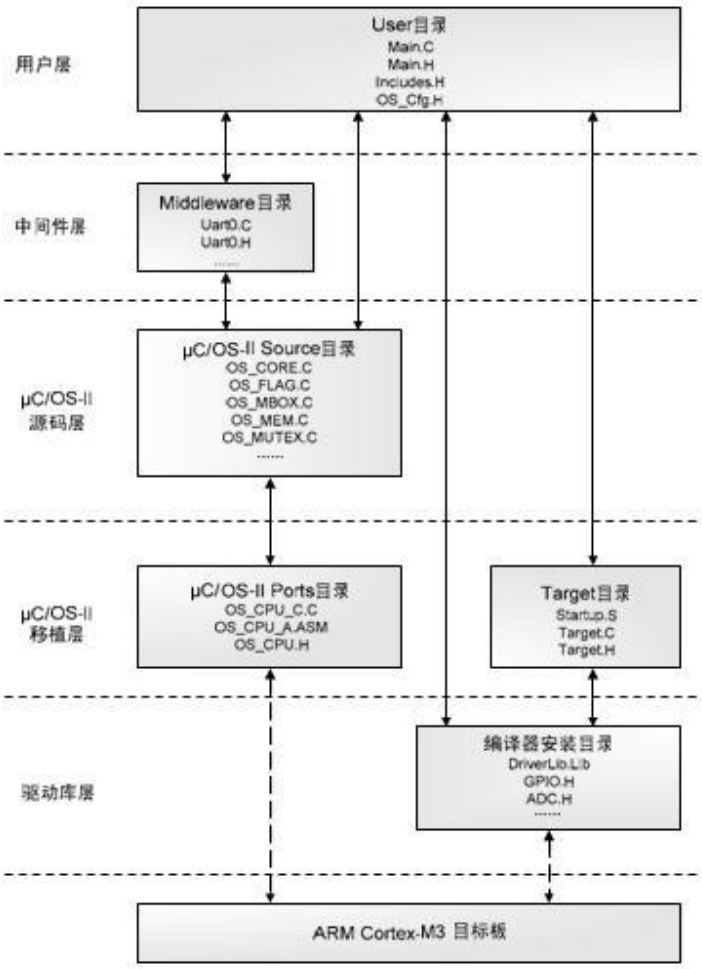
μCOSII采用的是可剥夺型实时多任务内核。可剥夺型的实时内核在任何时候都运行就绪了的最高优先级的任务。μCOSII的任务调度是完全基于任务优先级的抢占式调度，也就是最高优先级的任务一旦处于就绪状态,则立即抢占正在运行的低优先级任务的处理器资源。为了简化系统设计，μCOSII规定所有任务的优先级不同，因为任务的优先级也同时唯一标志了该任务本身。

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。



(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。

本实验原理

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、软件建立工程

首先要在main.c中声明任务，和任务的堆栈。

```
012 /*********************************************************************************************
013   VARIABLES 变量
014 *********************************************************************************************
015 static OS_STK  Task_StartStk[TASK_START_STK_SIZE];                    /*  The stack of s
016                                                                      /*  启动任务的堆栈
017 static OS_STK  Task_LedStk[TASK_LED_STK_SIZE];
018
019 static OS_STK  Task_OLEDbStk[TASK_OLEDb_STK_SIZE];
020
021 /*********************************************************************************************
022   FUNCTION PROTOTYPES 函数声明
023 *********************************************************************************************
024 static void taskStart (void  *parg);                                   /* 启动任务*/
025 static void taskLed (void  *parg);                                     /*  LED任务*/
026 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。

```
09 /*************************************************
10   TASK PRIORITIES 任务优先级
11 *************************************************
12 #define  TASK_START_PRIO                0
13 #define  TASK_LED_PRIO                  1
14 #define  TASK_OLEDb_PRIO                2
15 /*************************************************
16   TASK STACK SIZES  任务堆栈大小
17 *************************************************
18 #define  TASK_START_STK_SIZE            50
19 #define  TASK_LED_STK_SIZE              50
20 #define  TASK_OLEDa_STK_SIZE            50
21 #define  TASK_OLEDb_STK_SIZE            50
22
23 #ifdef __cplusplus
24     }
25 #endif
26
27 #endif
28 /*************************************************
29   END FILE
30 *************************************************
```

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。

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。

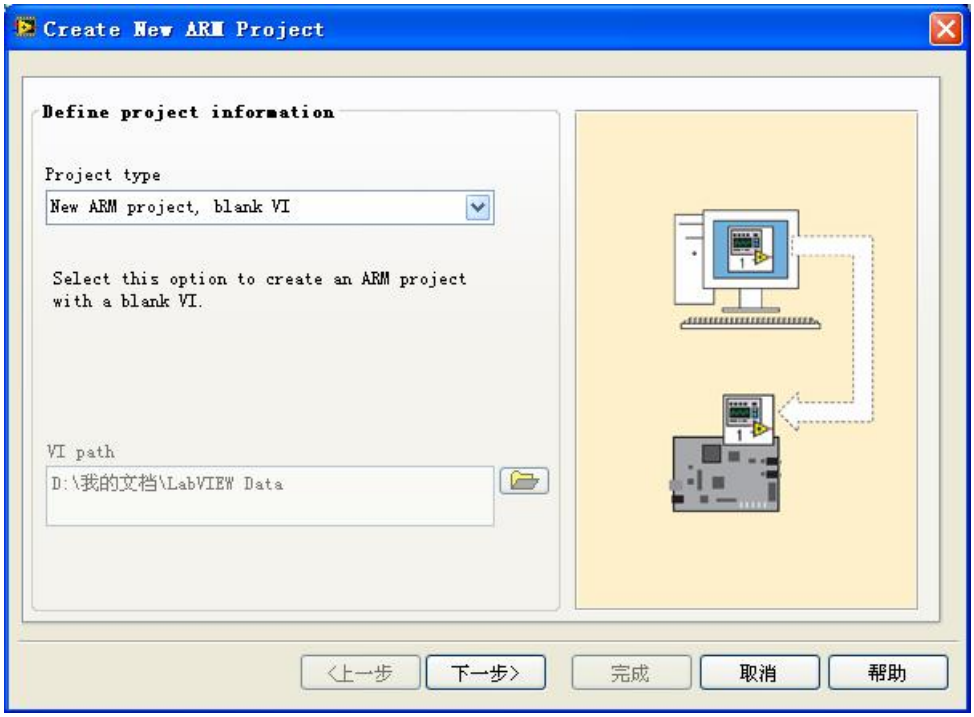
五、 实验步骤

将实验板与计算机连接。

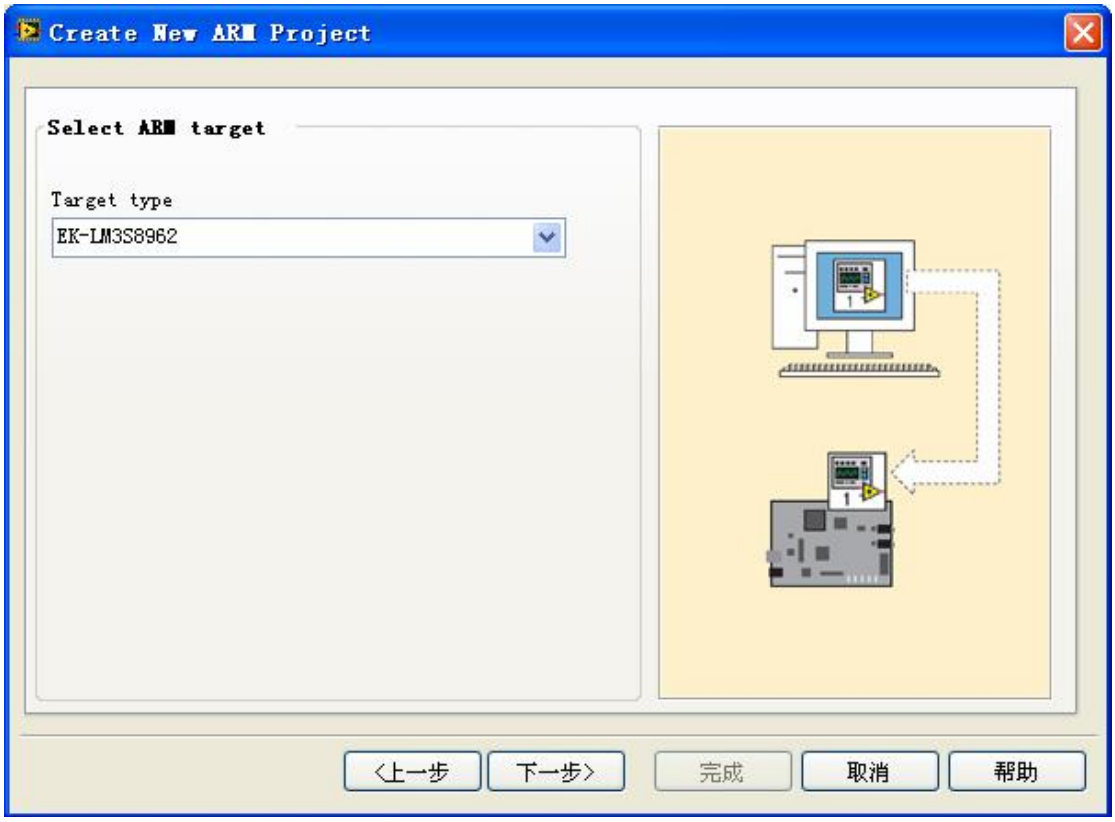
通过USB将实验板与计算机连接提供电源。

启动LABVIEW，选择终端选择**ARM Project**，单击开始新建工程（参见下图）：

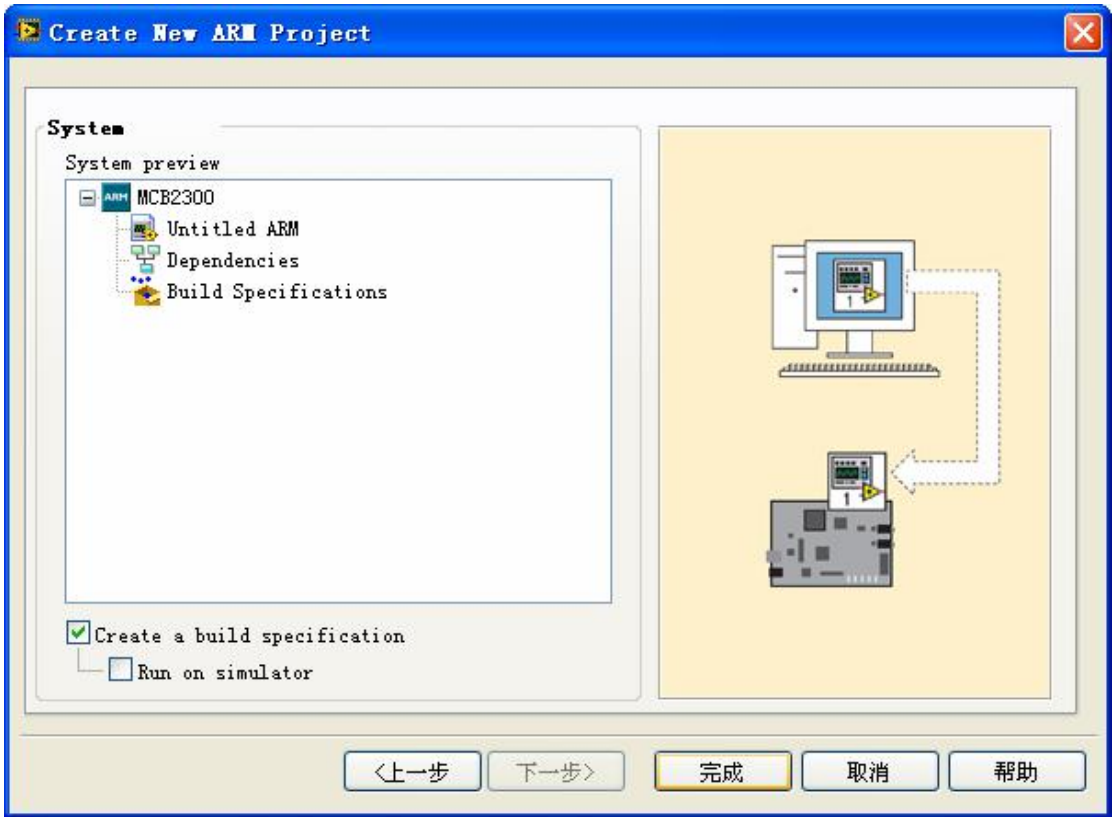
点击开始后，转入新项目创建页面（参见下图），然后点击下一步，进入新页面



在这个页面选择实验板的类型，这里我们选择EK-LM3S8962,然后点击下一步，进入新页面：

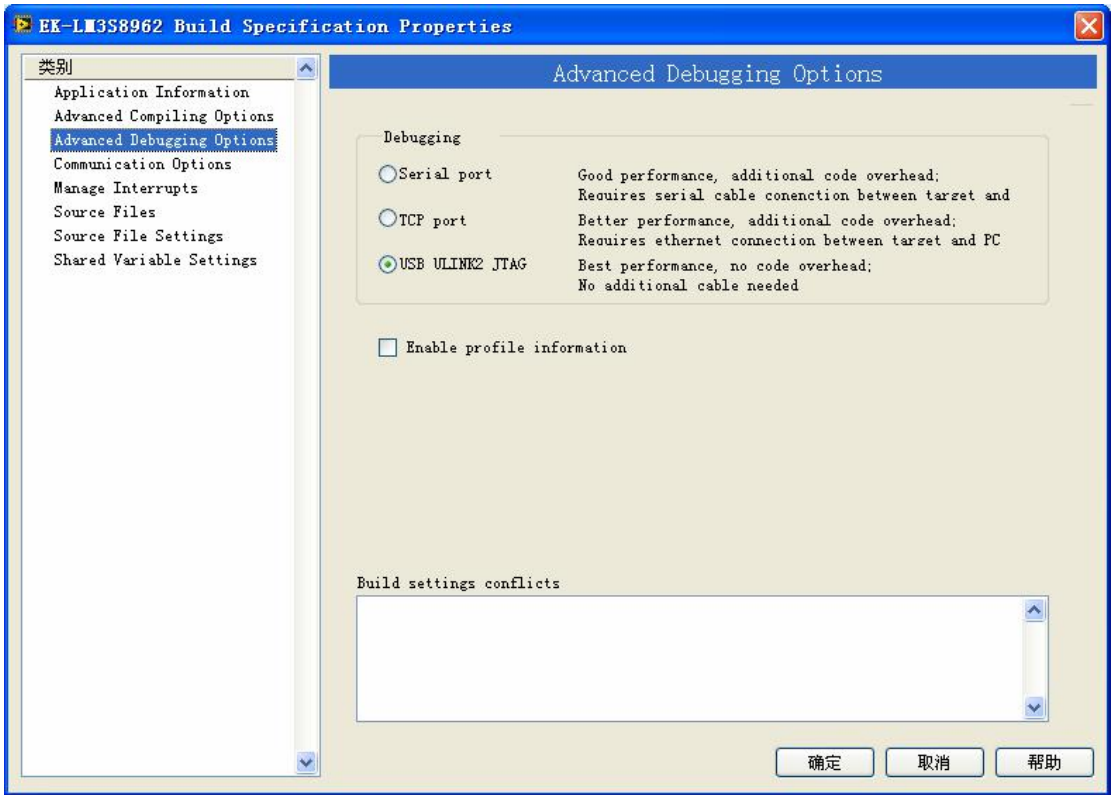


在这一页面中勾选Create a build specification，（Run on Simulator不能勾选）然后点击完成（参见下图）。

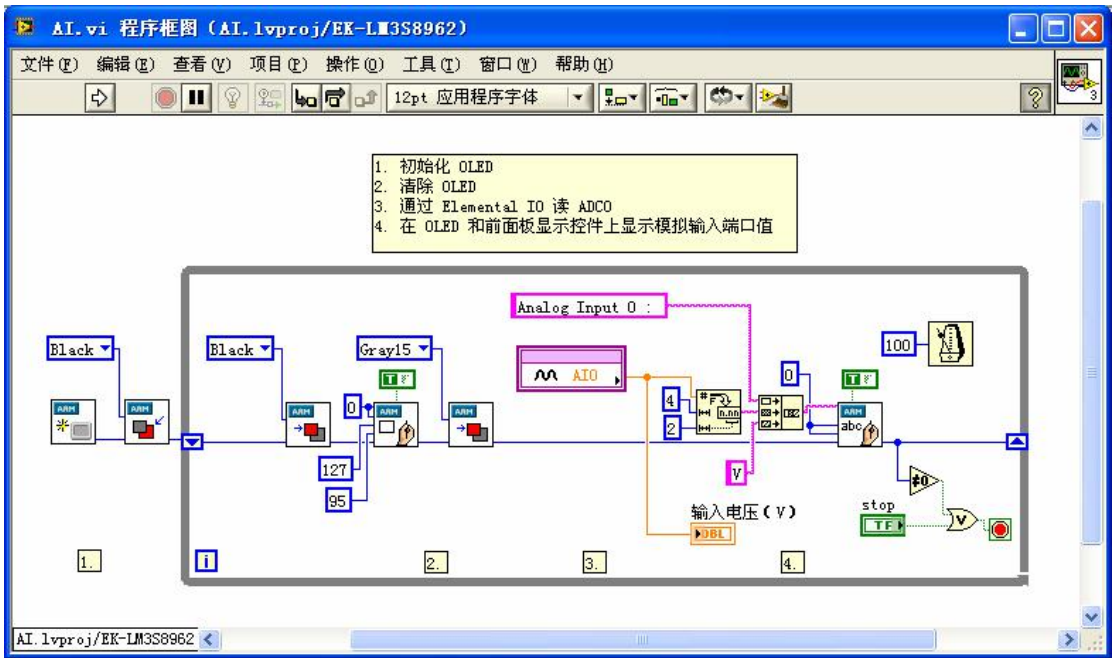


然后保存“项目”与“VI”，项目名称保存为“AI.lvproj”，VI名称保存为“AI.vi”：

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：

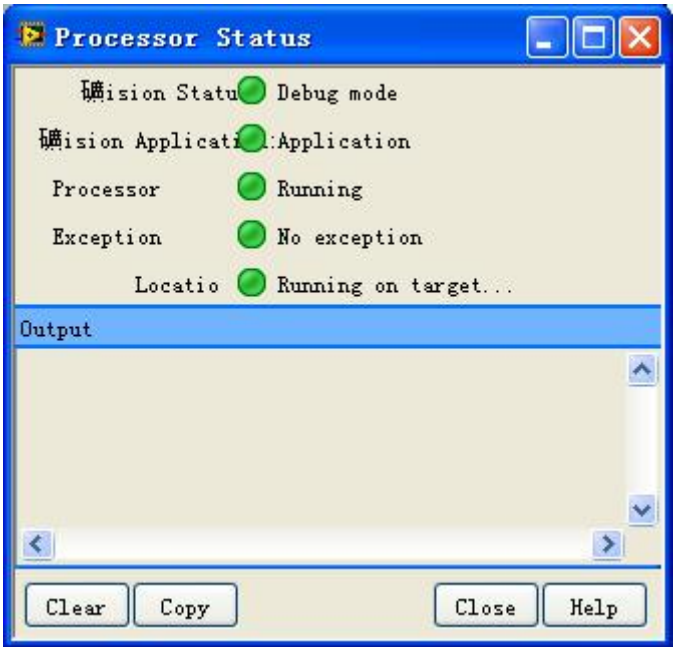


5、在项目浏览器中双击AI.vi,，参考下图建立程序框图：



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。



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。

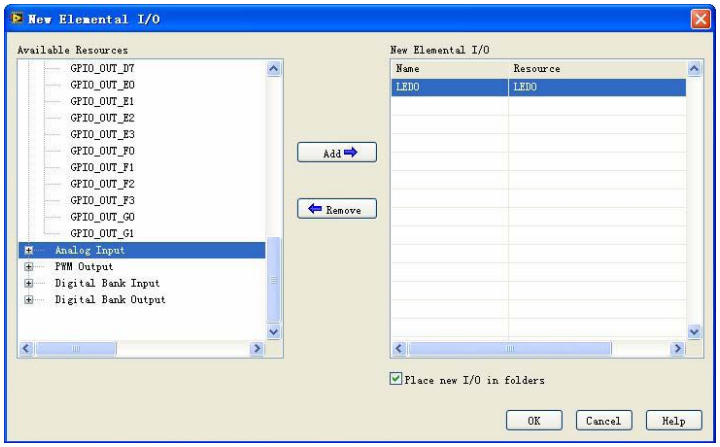
四、 实验环境

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。

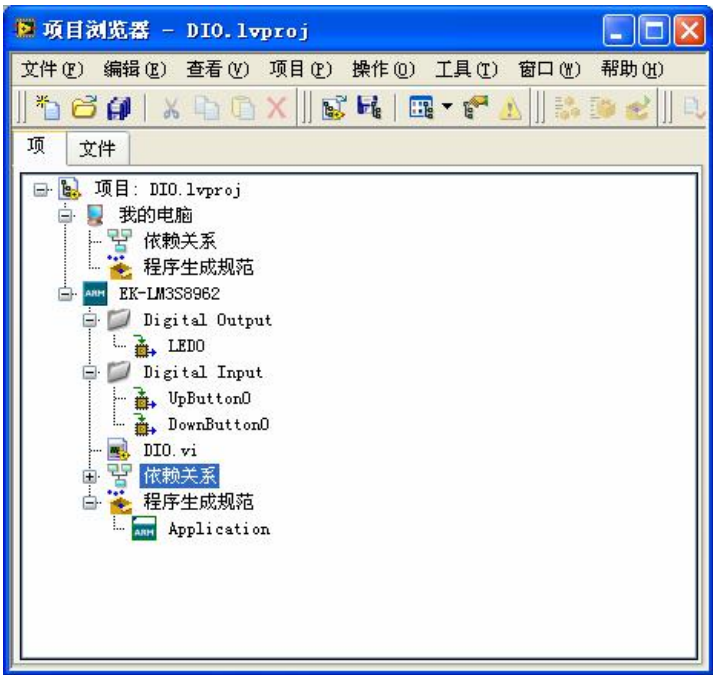
五、 实验步骤

1、建立工程：“**DI0..lvproj**”；

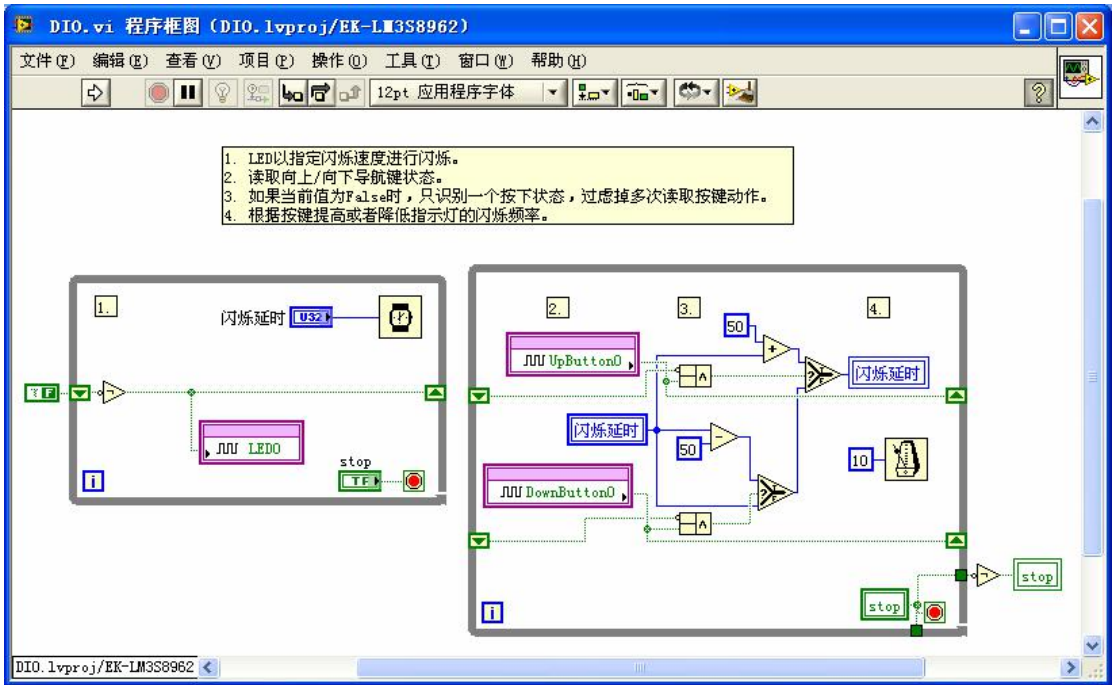
2、在项目浏览器中右键单击**EK-LM3S8982**，选择新建→**Elemental I/O**，添加数字输入端口**LED0**和数字输入端口**UpButton0**和**DownButton0**（参考下图）：



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DIO.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

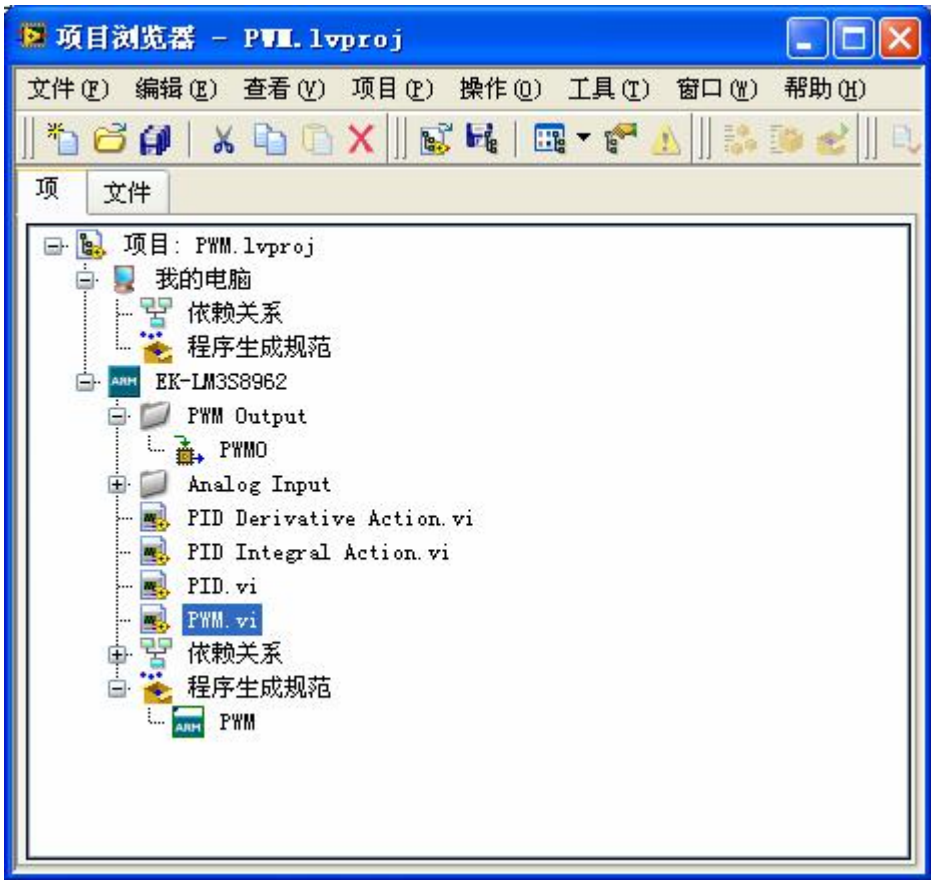
计算机、RealVIEW MDK 集成开发环境、LabVIEW、LabVIEW Embedded Module for ARM Controllers、ULINK2仿真器、实验板。

五、 实验步骤

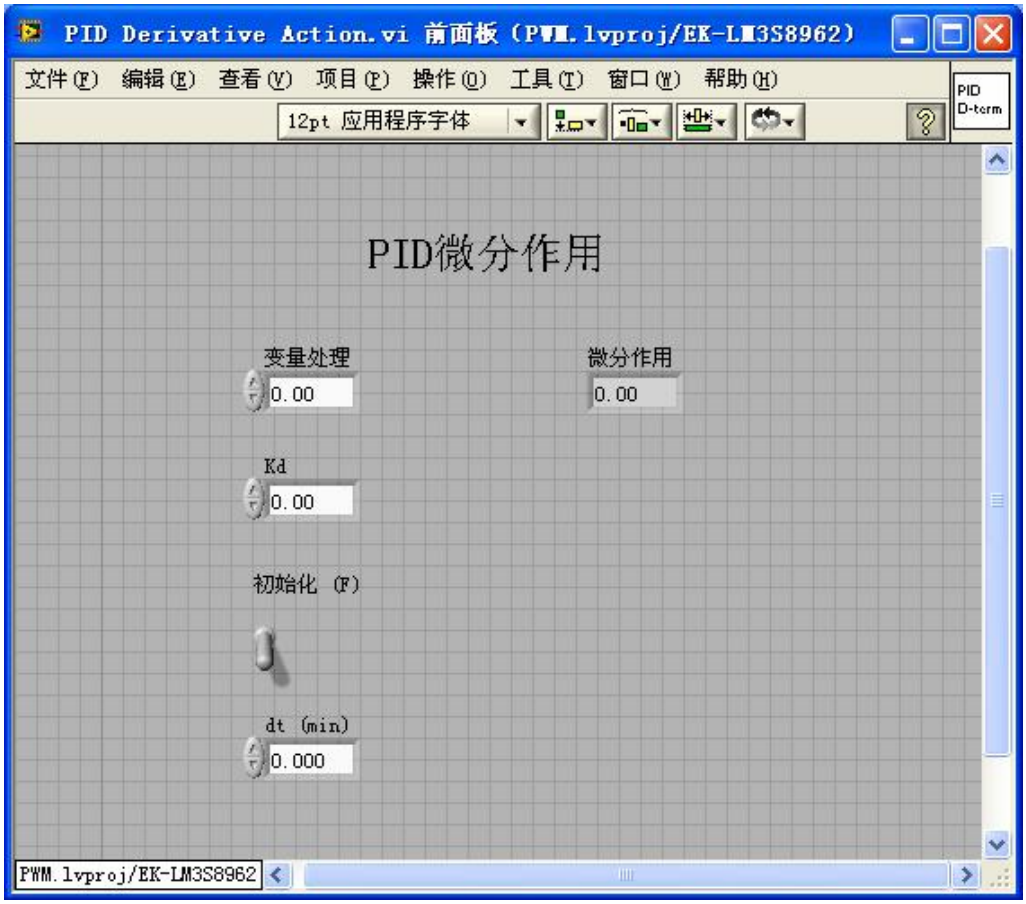
1、建立工程：**“PWM.lvproj”**；

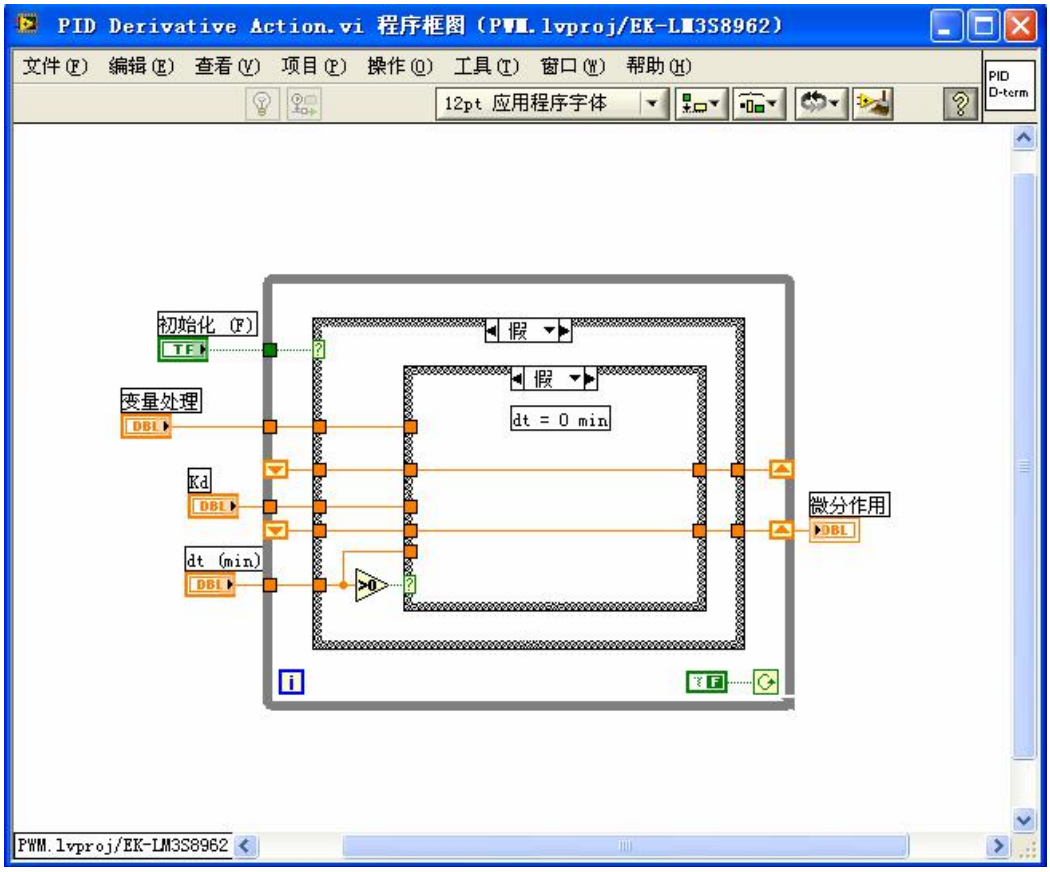
2、添加模拟输入端口**AI0**和脉宽调制端口**PWM1**；

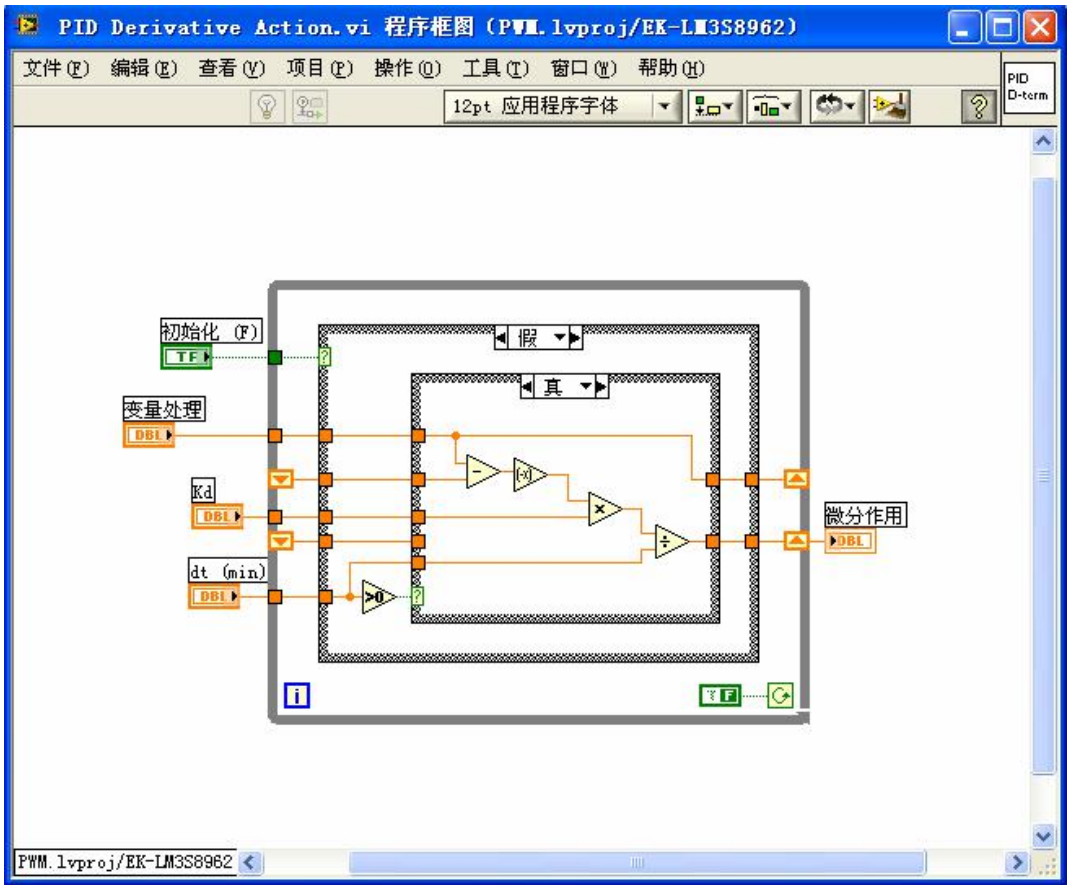
3、项目浏览器中右击**EK-LM3S8962**选择新建→虚拟文件夹并将其重命名为**❽PID SubVIs”**；右击**PID SubVIs**虚拟文件夹，新建三个VI并分别保存为**“PID Derivative Action.vi”**、**“PID Integral Action.vi”**和**“PID.vi”**（参考下图）：

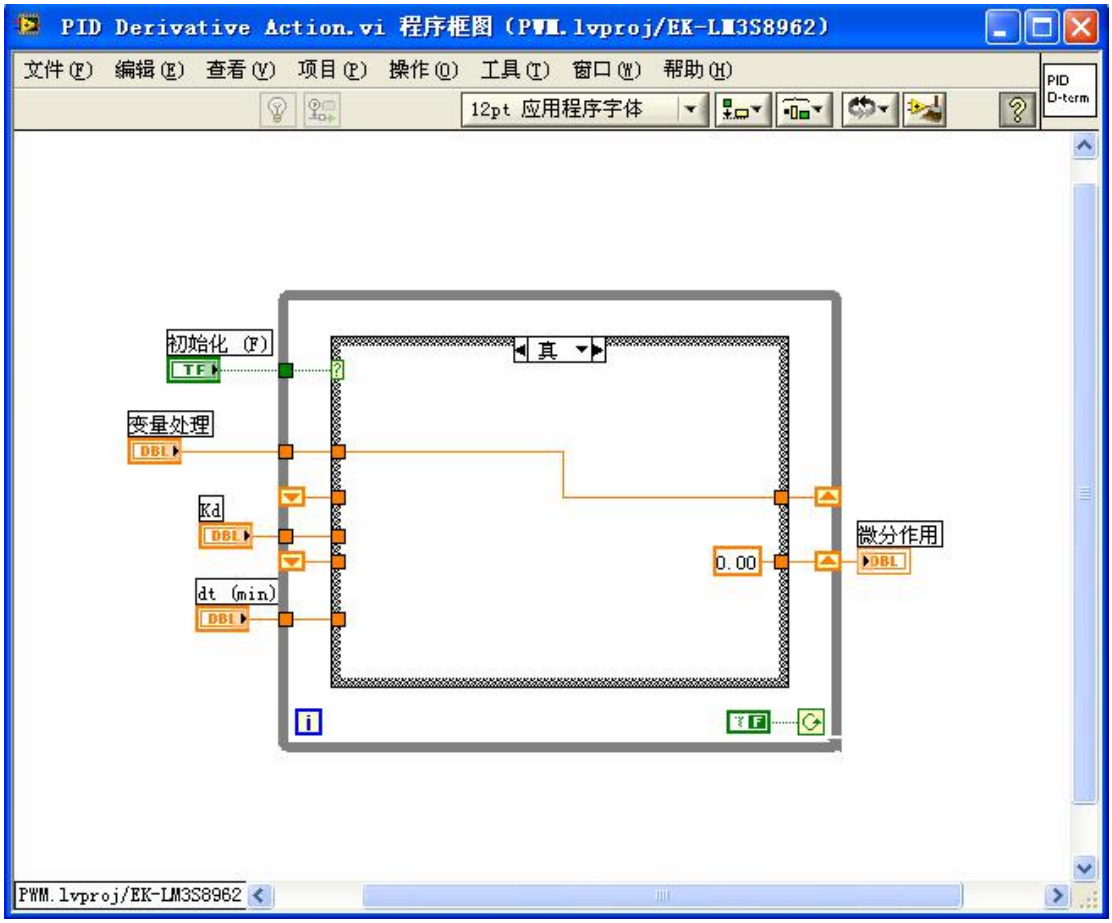


4、参考下图建立子VI **PID Derivative Action.vi**前面板和程序框图：

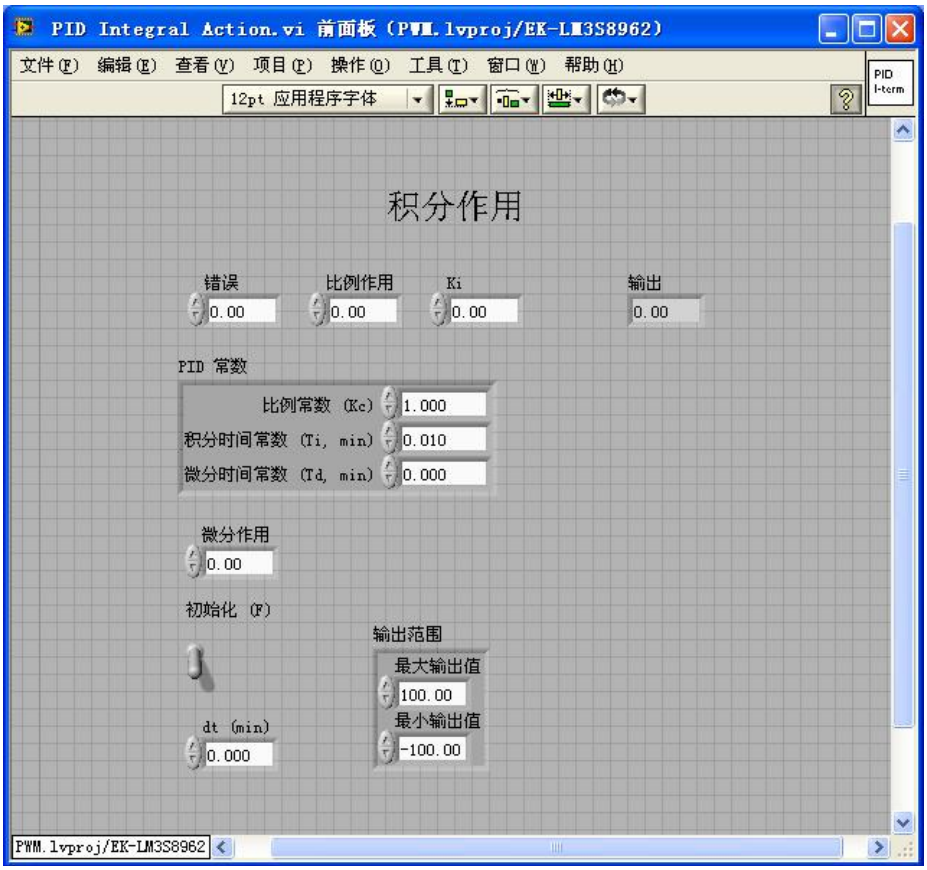

PID Derivative Action.vi 前面板 (PWM.lvproj/EK-LM3S8962)
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
12pt 应用程序字体
PID
D-term
PID微分作用
变量处理
0.00
微分作用
0.00
Kd
0.00
初始化 (F)
dt (min)
0.000
PWM.lvproj/EK-LM3S8962

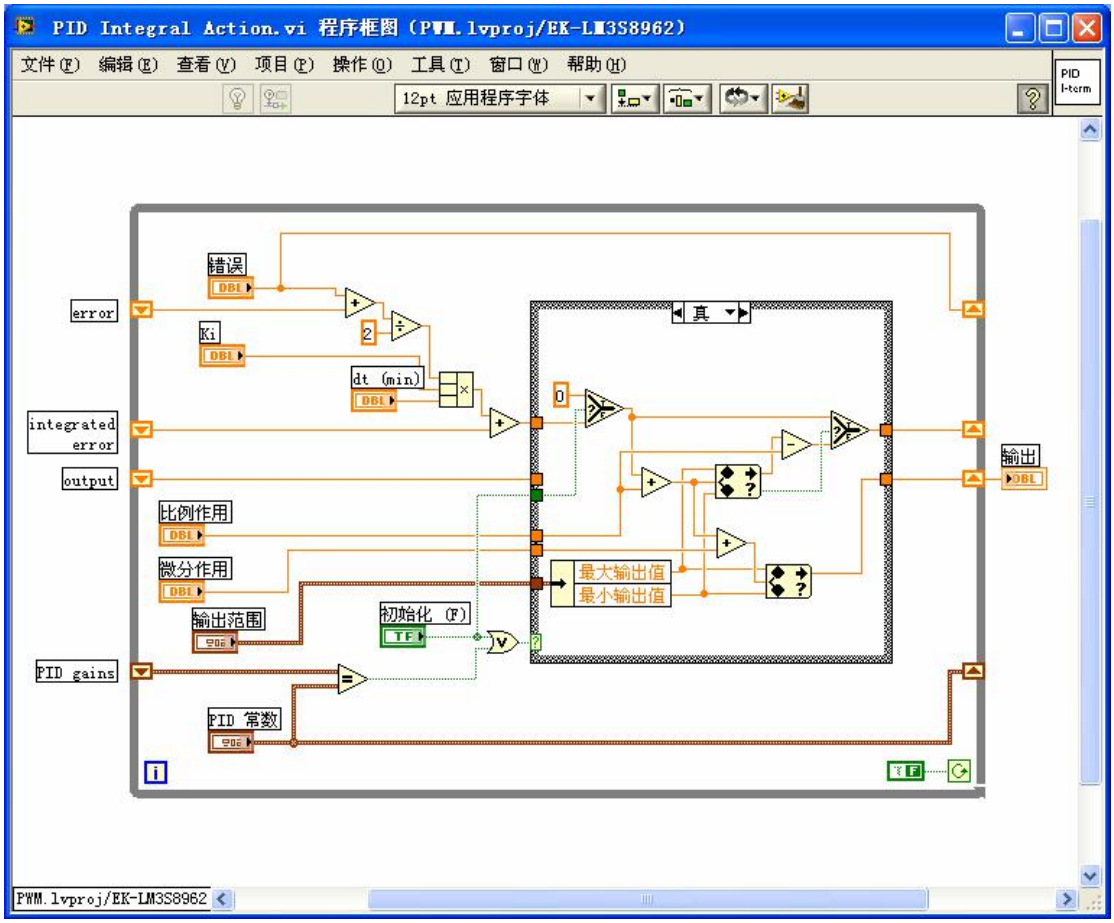

PID Derivative Action.vi 程序框图 (PWM.lvproj/EK-LM3S8962)
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
12pt 应用程序字体
PID
D-term
初始化 (F)
TF
变量处理
DBL
Kd
DBL
dt (min)
DBL
假
假
dt = 0 min
>0
微分作用
DBL
i
TF
PWM.lvproj/EK-LM3S8962

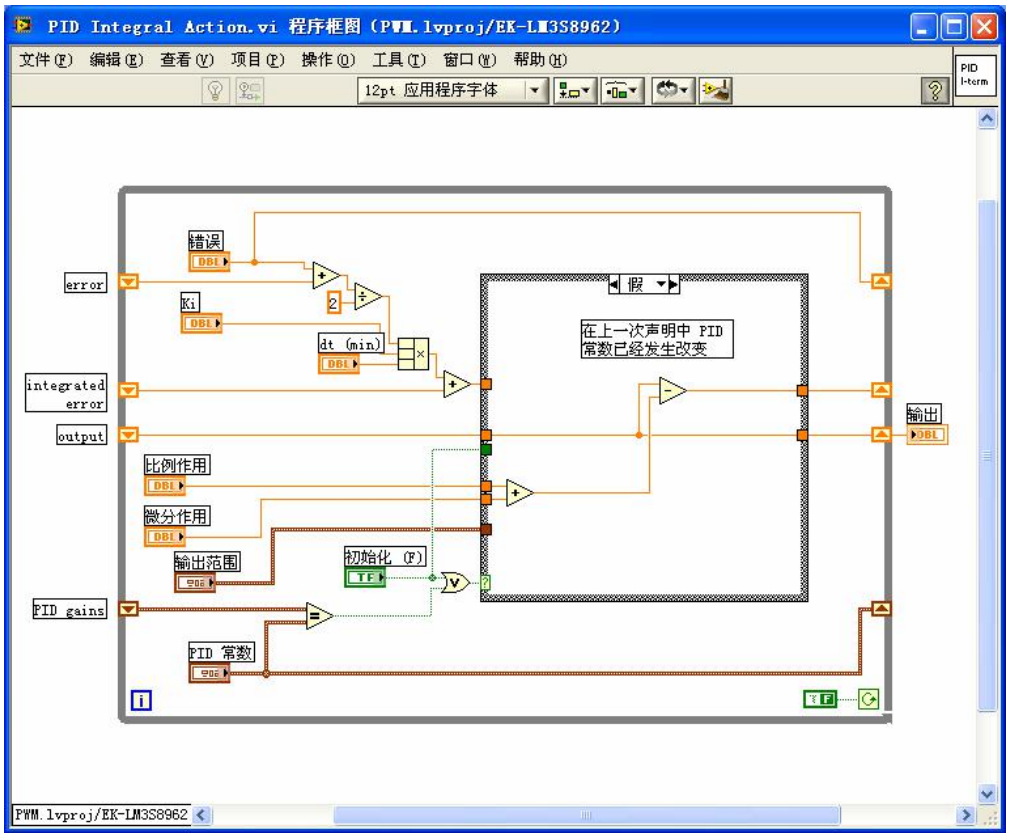

PID Derivative Action.vi 程序框图 (PWM.lvproj/EK-LM3S8962)
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
12pt 应用程序字体
PID
D-term
初始化 (F)
TF
假
真
变量处理
DBL
Kd
dt (min)
>0
微分作用
PWM.lvproj/EK-LM3S8962


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
真
变量处理
DBL
Kd
dt (min)
0.00
微分作用
PWM.lvproj/EK-LM3S8962

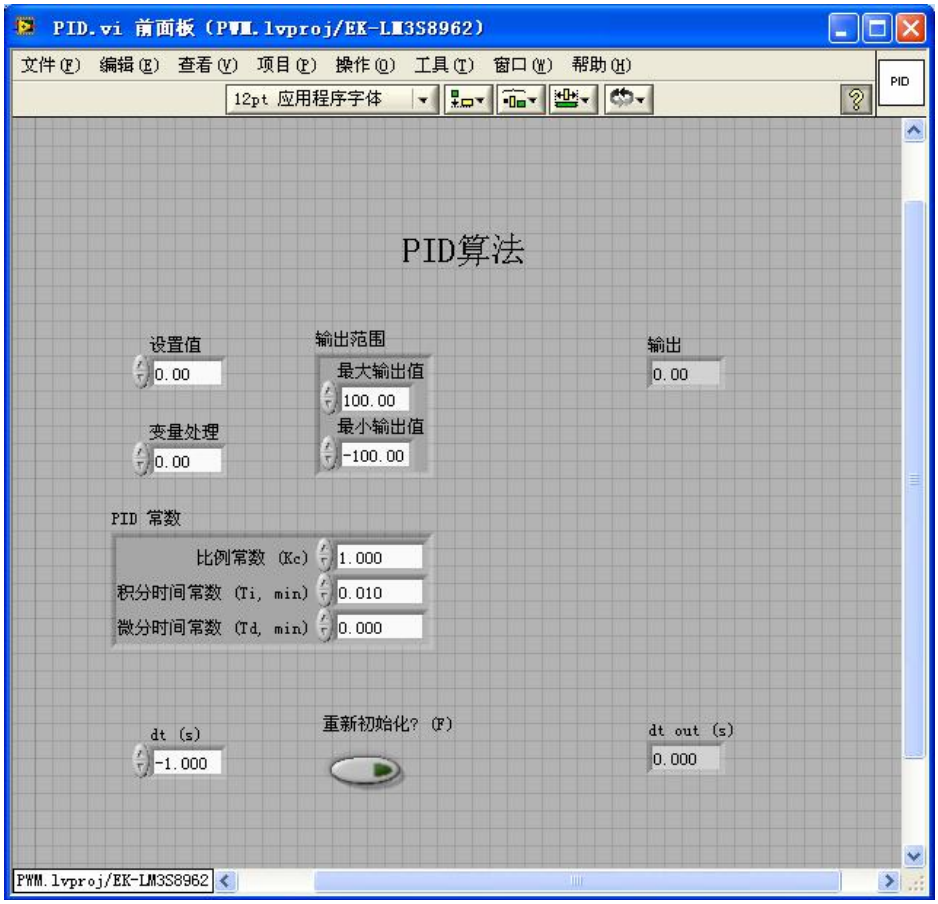
5、参考下图建立子VI PID Integral Action.vi前面板和程序框图；

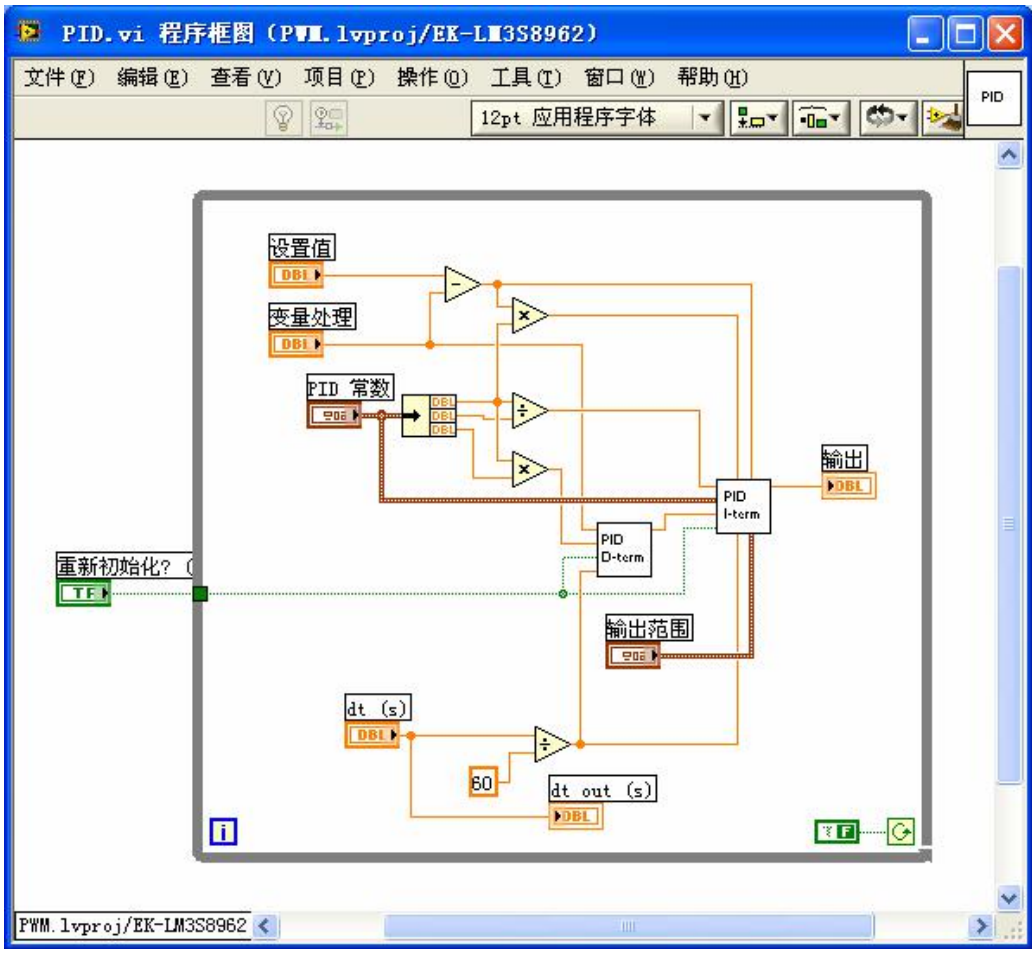




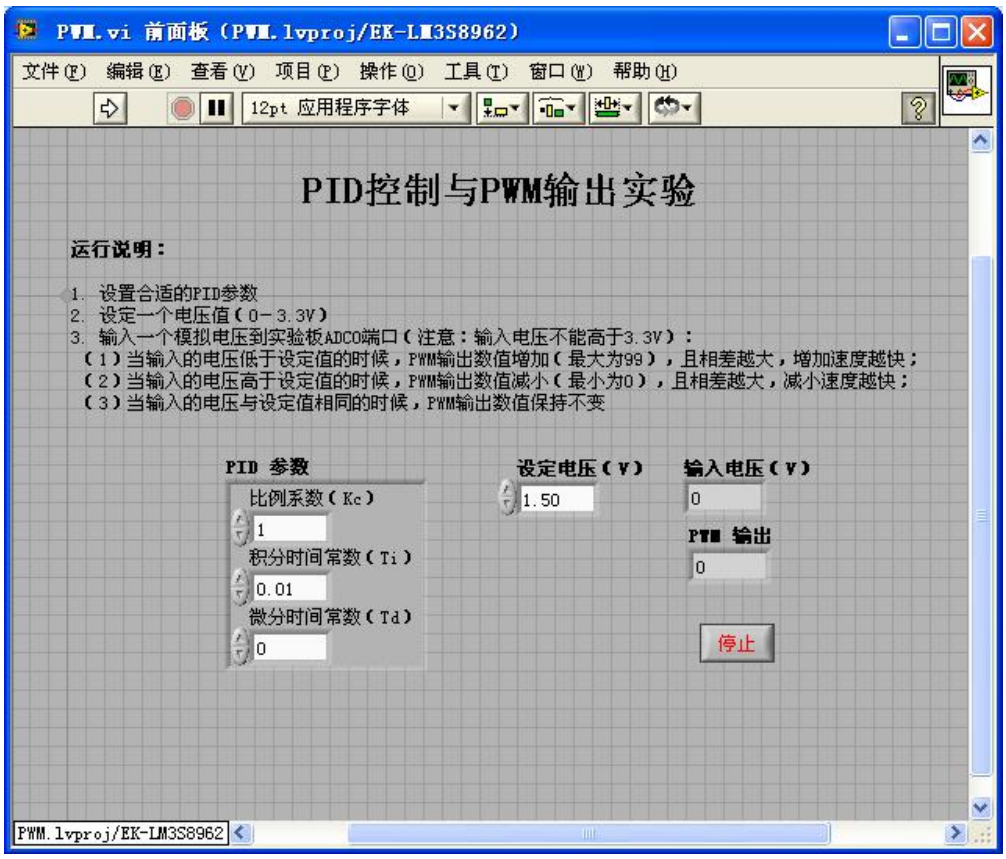


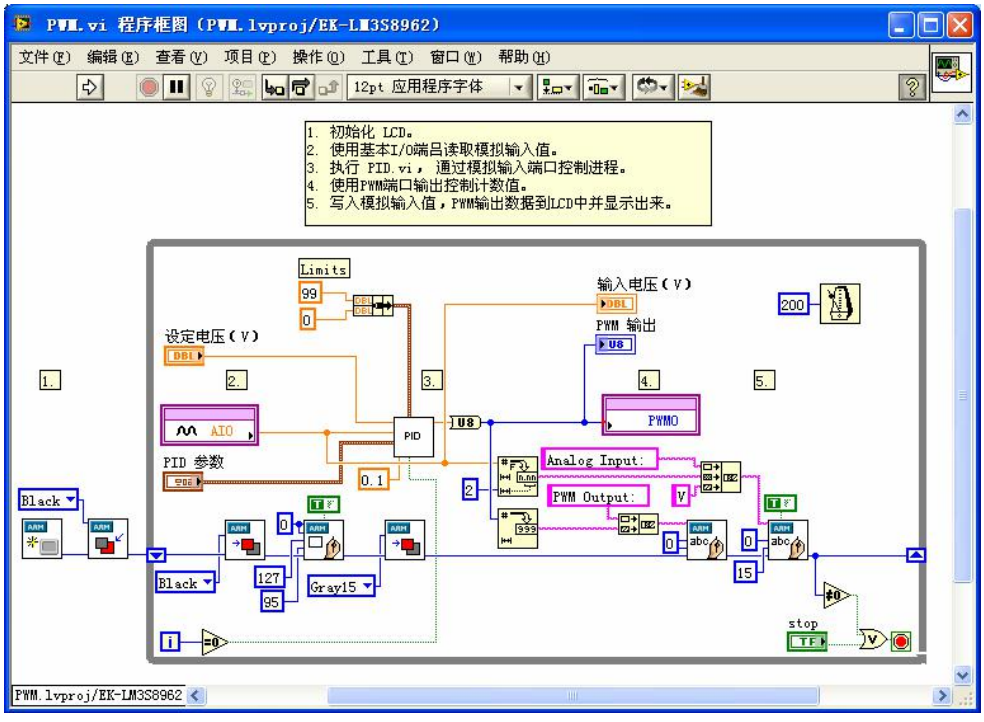
6、参考下图建立子 VI PID.vi 前面板和程序框图；





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。