Z32R 开发板
智林测控技术研究所
www.the0.net

版本 1.0

深入浅出的讲解，带您轻松进入嵌入式开发的大门

Simbba

2008-5-5

前言

智林测控的 Z32R 开发板是为了初学者设计的一款入门级开发板,考虑到趣味和实用,板上带有一块彩色液晶屏,可以更方便地观察试验数据.

在介绍片上接口资源时,不是单纯的理论讲解,而是边做边学,把理论融入实践,使读者了解 NXP 的 ARM 芯片的核心以及各种片上资源.

嵌入式系统是一个软硬件紧密结合的系统,研发嵌入式系统需要广泛的软硬件知识,嵌入式开发是实践性很强的领域,除了理论,必须不断实践.我们愿意和大家分享我们的甘苦.

什么是嵌入式系统?

一般来说,嵌入式系统是“执行专用功能并被内部计算机控制的设备或者系统.嵌入式系统不能使用通用型计算机,而且运行的是固化的软件,用术语表示就是固件(firmware),终端用户很难或者不可能改变固件.”

有技术概念的人,理解上面的定义应该没有问题,但是对非技术领域的人们来说可能存在一些障碍.一个更简单的定义是:“嵌入式系统就是包含了用户所不知晓的计算机的设备.”但是,甚至这个定义也需要某些解释.首先,大多数非 IT 人士对计算机的概念就是一个米色的贴着“Intel-Inside”标签的箱子.嵌入式计算机的发行量远远高于基于 Intel-x86/Pentium 的台式 PC 的发行量.Intel 或许“inside”,但是嵌入式系统却是无处不在.

人们很少会意识到他们往往随身携带了好几个嵌入式系统---手机,手表或者智能卡都嵌有它们,而且他们在与汽车,电梯,厨房设备,电视,录像机以及娱乐系统的嵌入式系统交互时也往往对此毫无觉察.嵌入式系统在工业机器人,医药设备,电话系统,卫星,飞行系统等领域扮演了一个更为重要的角色.正是“看不见”这一个特性嵌入式计算机与通用 PC 计算机相区分.

ARM

究竟什么是 ARM 呢？他是英国一家电子公司的名字，全名的意思是 Advanced RISC Machine。我来多说几句，ARM 敢为天下先，12 年前首创了 chipless 的生产模式，即该公司既不生产芯片，也不设计芯片，而是设计出高效的 IP 内核，授权给半导体公司使用，半导体公司在 ARM 技术的基础上添加自己的设计并推出芯片产品，最后由 OEM 客户采用这些芯片来构建基于 ARM 技术的系 统产品。ARM 的产品是 IP Core，没有任何物理意义上的硬件或者软件实体。目前全球有 103 家巨型 IT 公司在采用 ARM 技术，20 家最大的半导体厂商中有 19 家是 ARM 的用户， 包括德州仪器， Philips, Intel 等。20 大巨头中唯一没有购买 ARM 授权的是 Intel 的老对头 AMD，而 AMD 则收购了 Alchemy 公司与之抗衡，采用的是 MIPS 结构 （有少数宽带路由器使用 MIPS 处理器，思科路由器也有使用这个牌子的）。目前面向市场的有 ARM7, ARM9, ARM9E-S，StrongARM 和 ARM10 系列。所以说 ARM 处理器有许多 OEM 厂家在生产.

ARM(Advanced RISC Machines)，既可以认为是一个公司的名字，也可以认为是对一类微处理器的通称，还可以认为是一种技术的名字。

1991 年 ARM 公司成立于英国剑桥，主要出售芯片设计技术的授权。目前，采用 ARM 技术知识产权（IP）核的微处理器，即我们通常所说的 ARM 微处理器，已遍及工业控制、消费类电子产品、通信系统、网络系统、无线系统等各类产品市场，基于 ARM 技术的微处理器应用约占据了 32 位 RISC 微处理器 75％以上的市场份额，ARM 技术正在逐步渗入到我们生活的各个方面。ARM 公司是专门从事基于 RISC 技术芯片设计开发的公司，作为知识产权供应商，本身不直接从事芯片生产，靠转让设计许可由合作公司生产各具特色的芯片，世界各大半导体生产商从 ARM 公司购买其设计的 ARM 微处理器核，根据各自不同的应用领域，加入适当的外围电路，从而形成自己的 ARM 微处理器芯片进入市场。

目前，全世界有几十家大的半导体公司都使用 ARM 公司的授权，因此既使得 ARM 技术获得更多的第三方工具、制造、软件的支持，又使整个系统成本降低，使产品更容易进入市场被消费者所接受，更具有竞争力。

ARM 公司是知识产权出售公司,ARM 公司只设计内核的不生产具体的芯片.

90 年代初,ARM 率先推出 32 位 RISC 微处理器芯片系统 SoC 知识产权公开授权概念,从此改变了半导体行业. ARM 通过出售芯片技术授权,而非生产或销售芯片,建立起新型的微处理器设计,生产,和销售商业模式.更重要的是,ARM 开创了电子新纪元: 采用 ARM 技术的微处理器遍及各类电子产品,在汽车,消费娱乐,成像,工业控制,网络,存储,安保和无线等市场,ARM 技术无处不在.

ARM 公司自 1991 年正式成立以来，在 32 位 RISC (Reduced Instruction Set Computer CPU 开发领域不断取得突破，其结构已经从 V3 发展到 V6.由于 ARM 公司自成立以来，一直以 IP(Intelligence Property)提供者的身份向各大半导体制造商出售知识产权，而自己从不介入芯片的生产销售，加上其设计的芯核具有功耗低,成本低等显著优点，因此获得众多的半导体厂家和整机厂商的大力支持，在 32 位嵌入式应用领域获得了巨大的成功，目前已经占有 75\% 以上的 32 位 RISC 嵌入式产品市场.在低功耗,低成本的嵌入式应用领域确立了市场领导地位.现在设计,生产 ARM 芯片的国际大公司已经超过 50 多家，国内中兴通讯和华为通讯等公司也已经购买 ARM 公司的芯核用于通讯专用芯片的设计.

目前非常流行的 ARM 芯核有 ARM7TDMI, StrongARM, ARM720T, ARM9TDMI, ARM922T, ARM940T, ARM946T, ARM966T, ARM10TDM1 等.自 V5 以后,ARM 公司提供 PiccoloDSP 的芯核给芯片设计者,用于设计 ARM+DSP 的 SOC(System On Chip)结构的芯片.此外,ARM 芯片还获得了许多实时操作系统(Real Time Operating System)供应商的支持,比较知名的有:Windows CE, Linux, pSOS, VxWorks, Nucleus, EPOC, uCOS, BeOS 等.

随着国内嵌入式应用领域的发展,ARM 芯片必然会获得广泛的重视和应用.ARM 芯片有多达十几种的芯核结构,70 多家芯片生产厂家,以及千变万化的内部功能配置组合.

第一章 STM32 ARM 处理器

STM32 系列 32 位闪存微控制器基于突破性的 ARM Cortex™-M3 内核，这是一款专为嵌入式应用而开发的内核。STM32 系列产品得益于 Cortex-M3 在架构上进行的多项改进，包括提升性能的同时又提高了代码密度的 Thumb-2 指令集，大幅度提高的中断响应，而且所有新功能都同时具有业界最优的功耗水平。目前 ST 是第一个推出基于这个内核的主要微控制器厂商。 STM32 系列产品的目的是为 MCU 用户提供新的自由度。它提供了一个完整的 32 位产品系列，在结合了高性能、低功耗和低电压特性的同时保持了高度的集成 性能和简易的开发特性。

STM32 的优势

- 搭载 ARM 公司最新的、具有先进架构的 Cortex-M3 内核
- 出色的实时性能
- 优越的功效
- 高级的、创新型外设
- 最大的集成性
- 易于开发，加速了面市时间

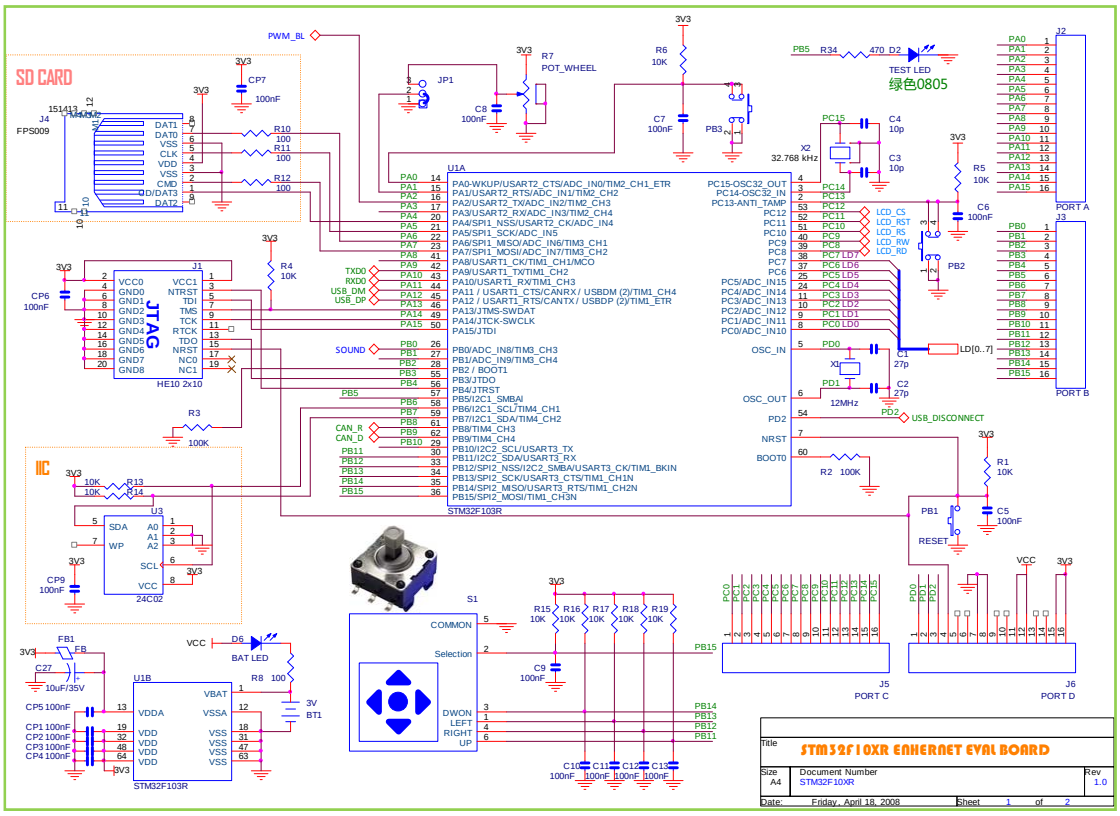
整个产品系列具有脚到脚、外设和软件的高度兼容性，为您提供最大的灵活性。

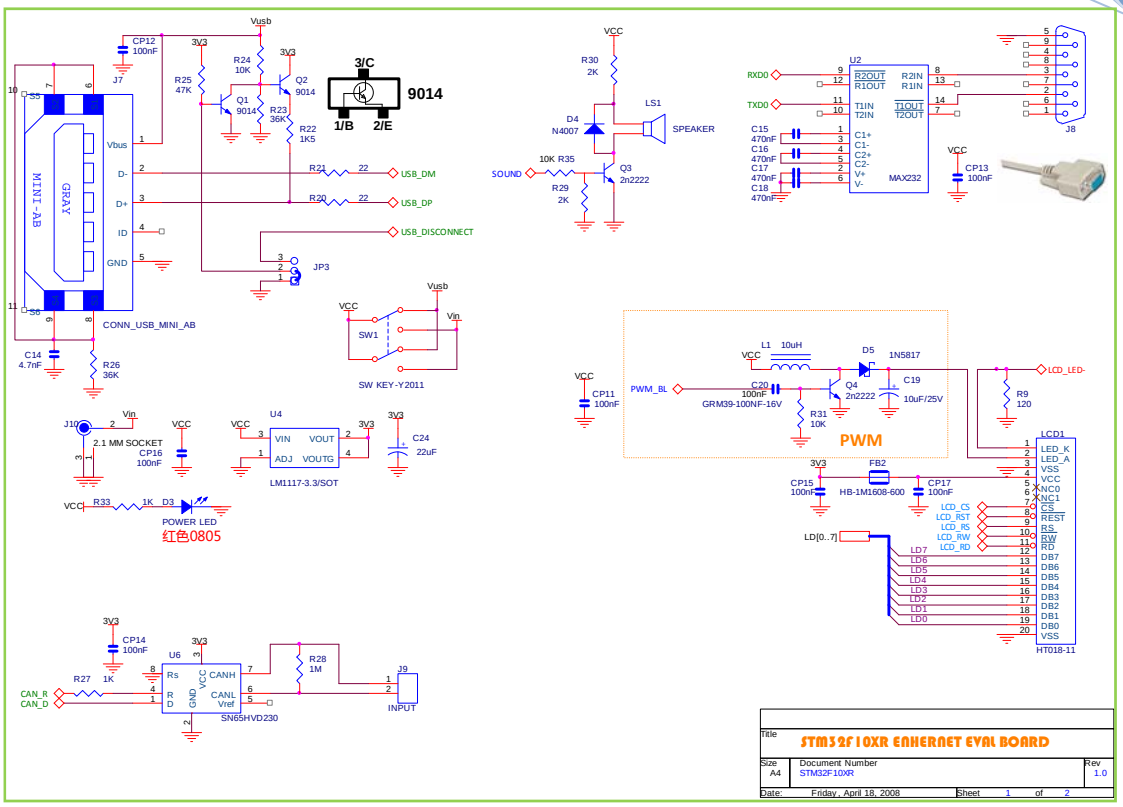
STM32 ，最优的平台 STM32 是您最优的平台，在这一平台上可以支持多项应用。

- 从较少的存储及引脚要求到更大的需求
- 从要求高速运算到使用电池供电
    - 从单一的成本敏感型到复杂的高价值产品 整个产品系列脚到脚、外设和软件的高度兼容性为您提供完全的灵活性。不必改变您的初始设计和软件，您就可以升级到一个较高的存储容量或降级到一个较低的存储容量，也可换用不同的封装形式。

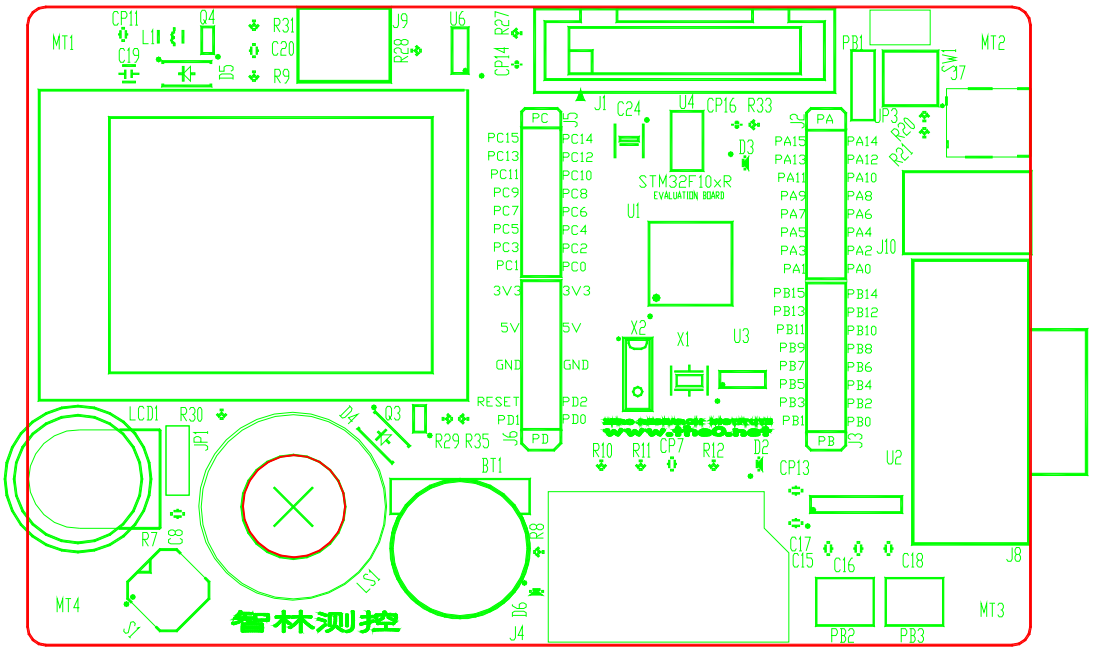
第二章 智林 Z32R 开发板

一、Z32R 开发板原理图



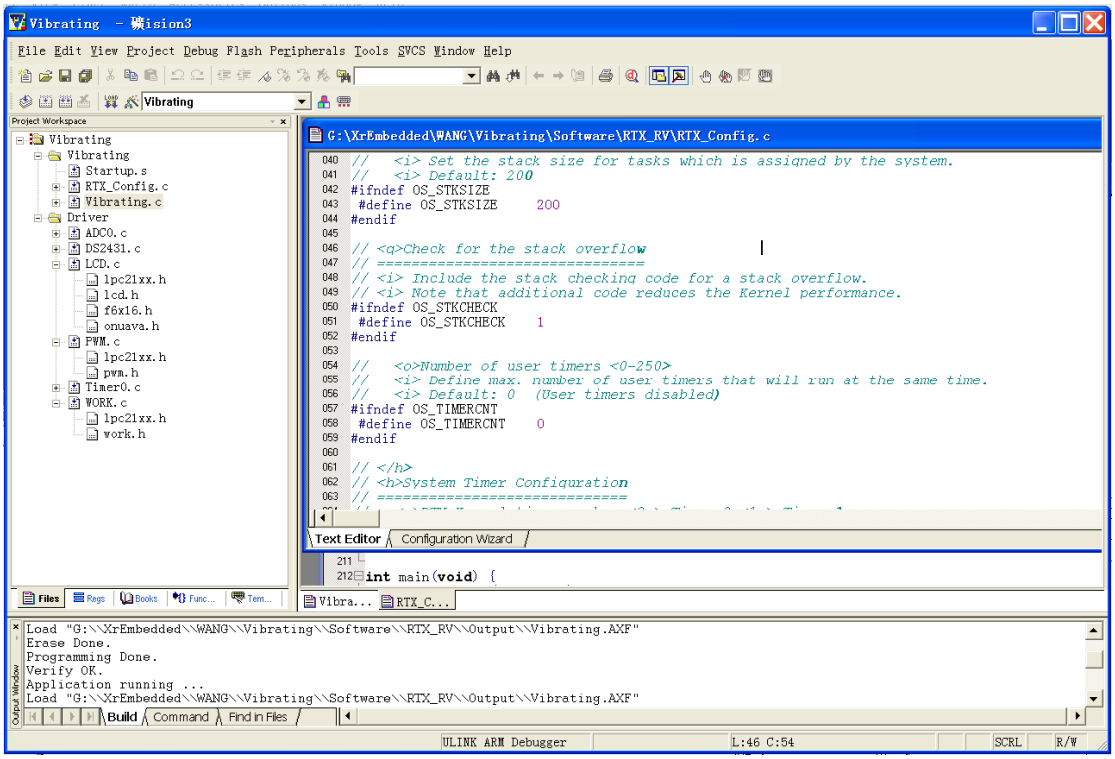

Vusb
CP12 100nF
J7
3V3
R24 10K
R25 47K
Q1 9014
Q2 9014
R23 36K
R22 1K5
3/C
9014
1/B 2/E
Vbus
D-
D+
ID
GND
CONN_USB_MINI_AB
R21 22
R20 22
USB_DM
USB_DP
USB_DISCONNECT
JP3
C14 4.7nF
R26 36K
SW1
SW KEY-Y2011
J10
2.1 MM SOCKET
CP16 100nF
U4
LM1117-3.3/SOT
C24 22uF
R33 1K
D3
POWER LED
红色0805
CP14 100nF
U6
R27 1K
CAN_R
CAN_D
SN65HVD230
R28 1M
J9
INPUT
R30 2K
D4 N4007
LS1
SPEAKER
SOUND
10K R35
R29 2K
Q3 2n2222
U2
MAX232
RXD0
TXD0
C15 470nF
C16 470nF
C17 470nF
C18 470nF
CP13 100nF
J8
L1 10uH
D5 1N5817
PWM_BL
CP11 100nF
C20 100nF GRM39-100NF-16V
Q4 2n2222
C19 10uF/25V
R31 10K
PWM
R9 120
LCD_LED-
FB2
HB-1M1608-600
CP15 100nF
CP17 100nF
LD[0..7]
LCD_CS
LCD_RST
LCD_RS
LCD_RW
LCD_RD
LCD1
HT018-11
Title
STM32F10XR ENHERNET EVAL BOARD
Size A4
Document Number STM32F-10XR
Rev 1.0
Date: Friday, April 18, 2008
Sheet 1 of 2

二、Z32R 开发板板图布局



第三章 好用的 Keil for ARM 开发环境

很多电子工程师都是从 51 单片机转过来的，很熟悉 Keil 的 C51 开发环境，当看到 Keil for ARM 的开发环境时，你会发现一切是那么熟悉，我们几乎不用关心汇编语言，可以直接进入 C 语言编程。我们所要学习和了解的主要内容是熟悉片上设备的用法。



Keil uVision 调试器可以帮助用户准确地调试 ARM 器件的片内外围功能(I2C、CAN、UART、SPI、中断、I/O 口、A/D 转换器、D/A 转换器 和 PWM 模块等功能)。ULINK USB-JTAG 转换器将 PC 机的 USB 端口与用户的目标硬件相连(通过 JTAG 或 OCD)，使用户可在目标硬件上调试代码。通过使用 Keil uVision IDE/调试器和 ULINK USB-JTAG 转换器，用户可以很方便地编辑、下载和在实际的目标硬件上测试嵌入的程序。

支持 Philips、Samsung、 Atmel、 Analog Devices、 Sharp、 ST 等众多厂商 ARM7 内核的 ARM 微控制器。

高效工程管理的 uVision3 集成开发环境

- Project/Target/Group/File 的重叠管理模式，并可逐级设置；
- 高度智能彩色语法显示；
- 支持编辑状态的断点设置,并在仿真状态下有效。

高速 ARM 指令/外设仿真器

- 高效模拟算法缩短大型软件的模拟时间；
- 软件模拟进程中允许建立外部输入信号；
- 独特的工具窗口，可快速查看寄存器和方便配置外设；
- 支持 C 调试描述语言，可建立与实际硬件高度吻合的仿真平台；

- 支持简单/条件/逻辑表达式/存储区读写/地址范围等断点。

多种流行编译工具选择

- Keil 高效率 C 编译器；
- ARM 公司的 RealView 编译器；
- GNU GCC 编译器；
- 后续厂商的编译器。

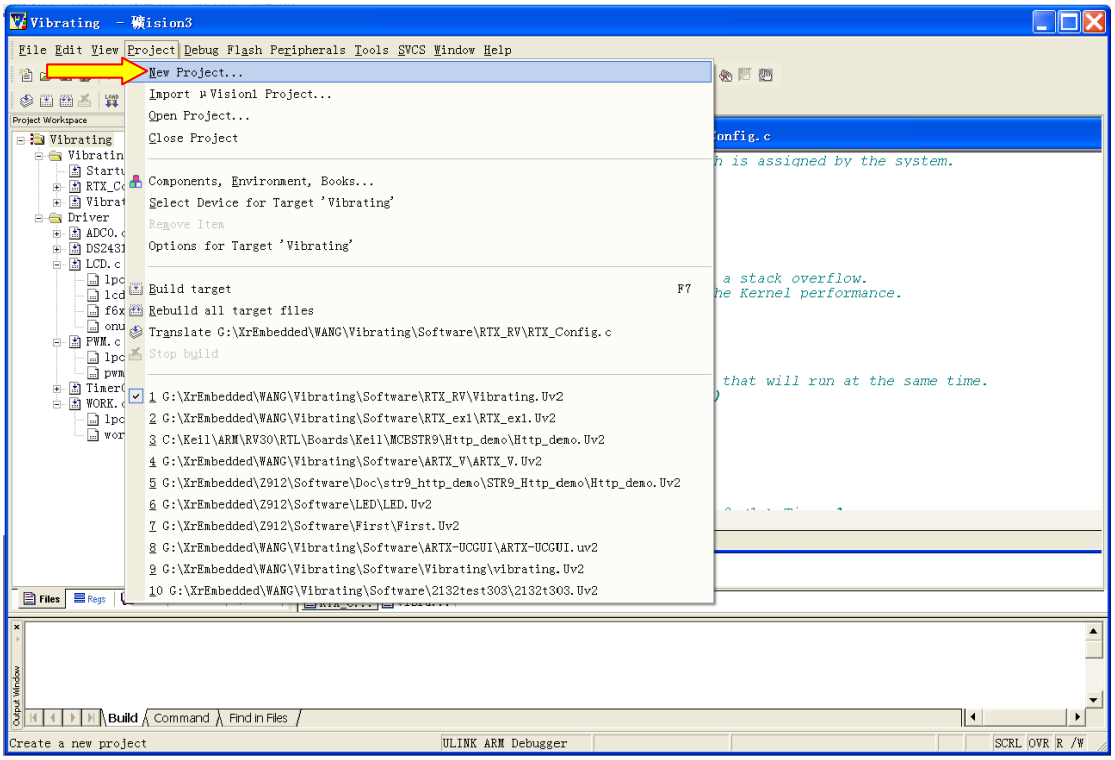
第四章 开始行动---第一个实验

在本章中我们会通过实际操作，熟悉 Keil 开发环境。所以这章会很详细的解说具体操作。

一、启动 Keil

出现启动画面后

进入 Keil 开发环境，Keil 会自动加载上次开发的项目。

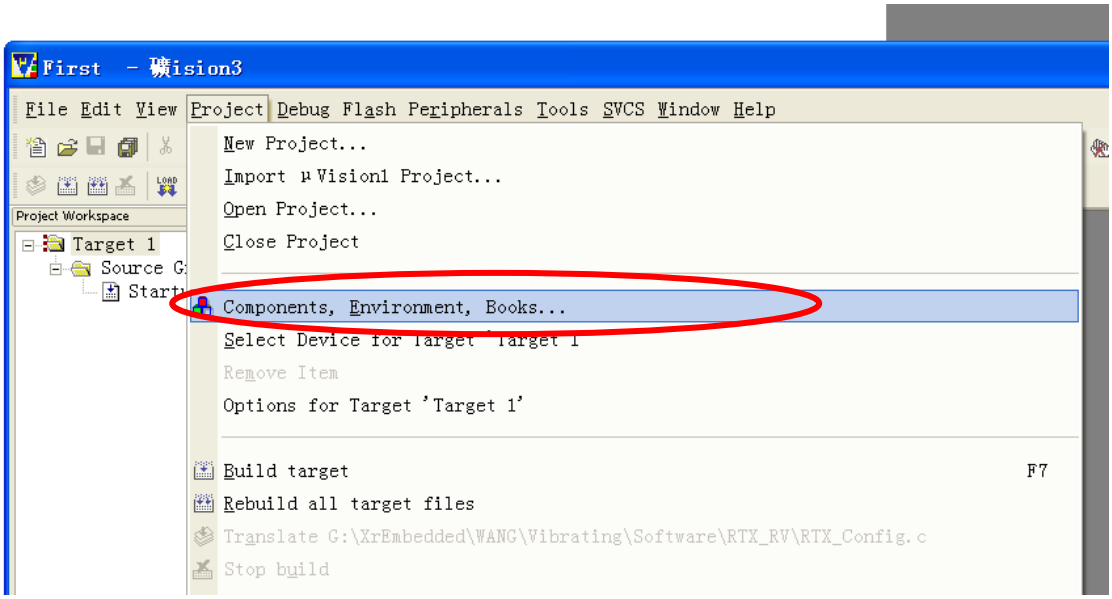


二、建立新项目

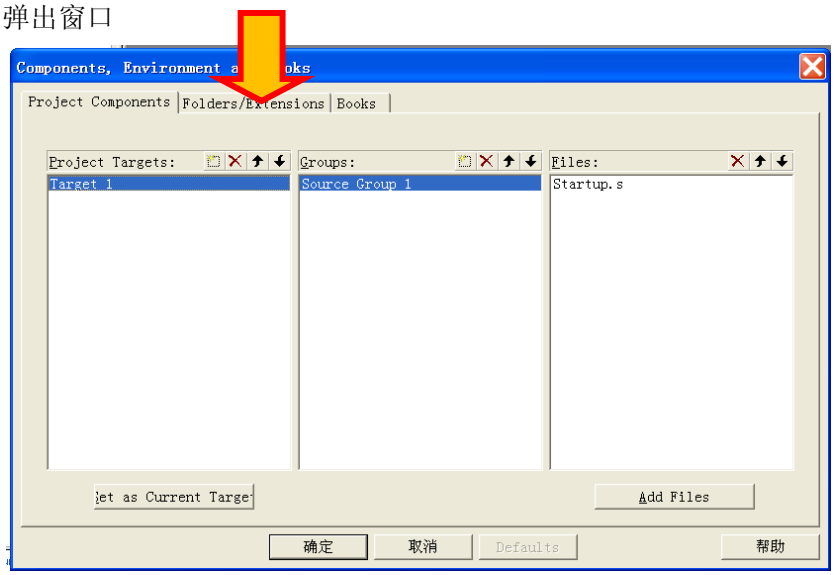
设置编译器

因为 Keil 公司已经被 ARM 公司买去了，现在我们可以在 Keil 环境中使用 ARM 的编译器了，

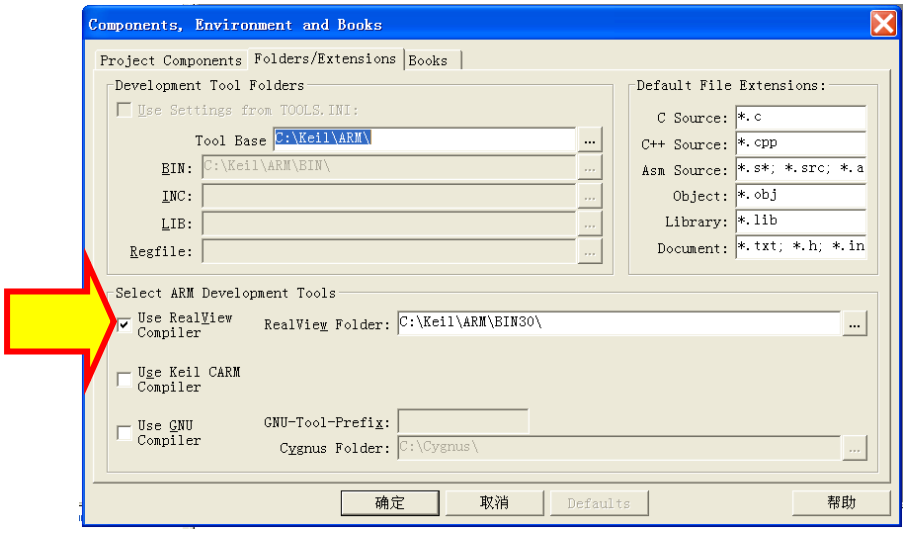
所以在建来新项目前，我们要注意一下当前编译器是什么，点击 Project 项的



弹出窗口



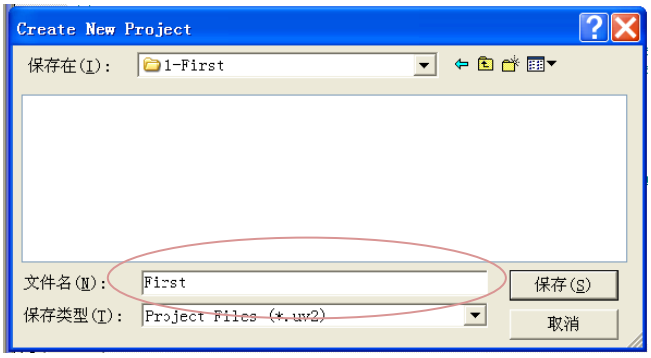
点击“Folders/Extensions”



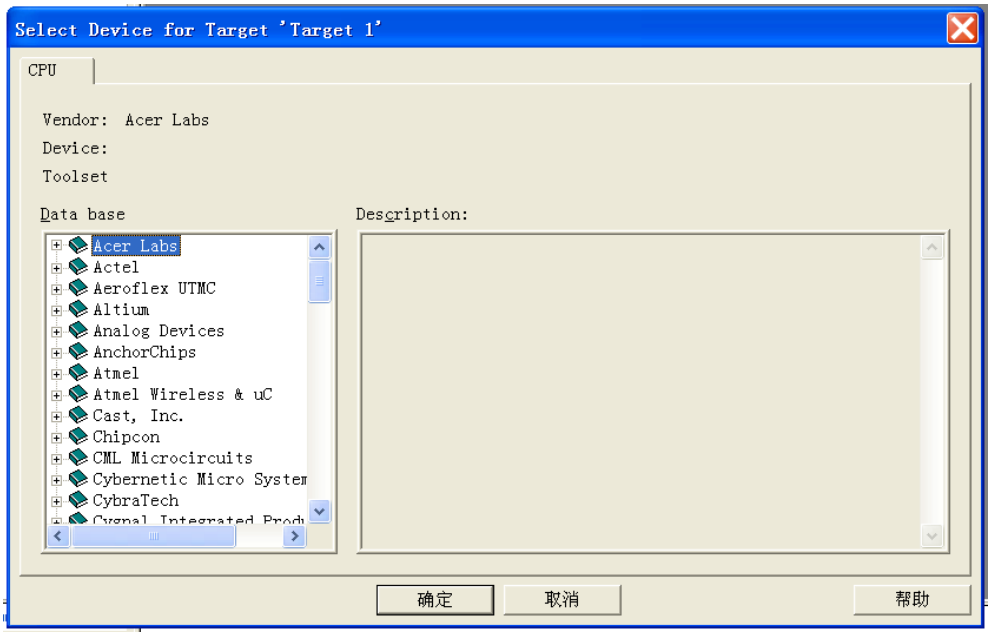
勾选 Use RealView Compiler，我们使用 ARM 公司的最新发行的编译器 RealVfiew。

建立新项目

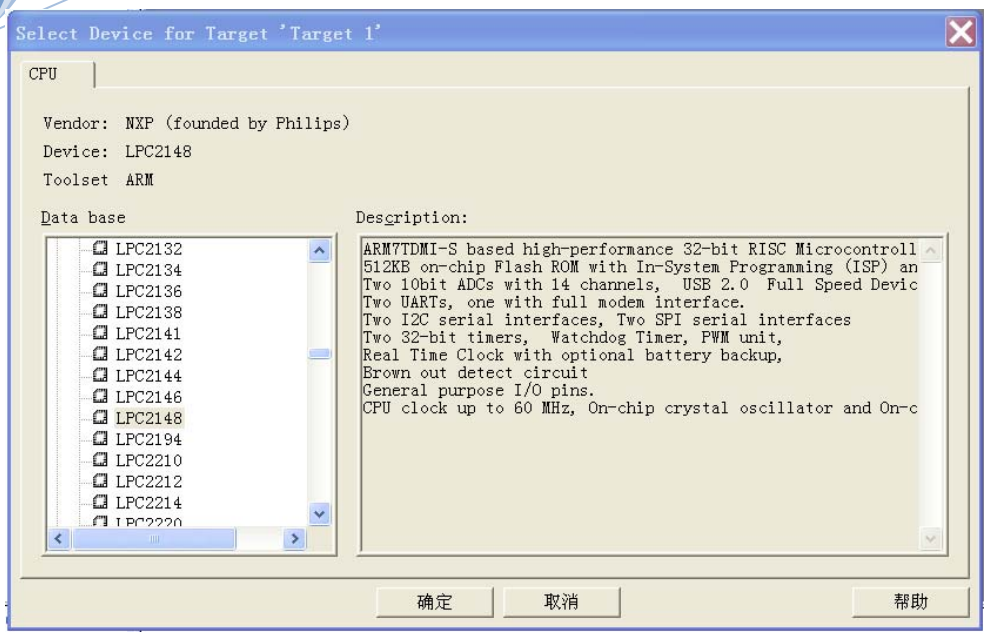
点击主菜单的 Projcet 项的 New Project。弹出文件对话窗：



选择新项目的文件夹，输入新项目名称，点击保存

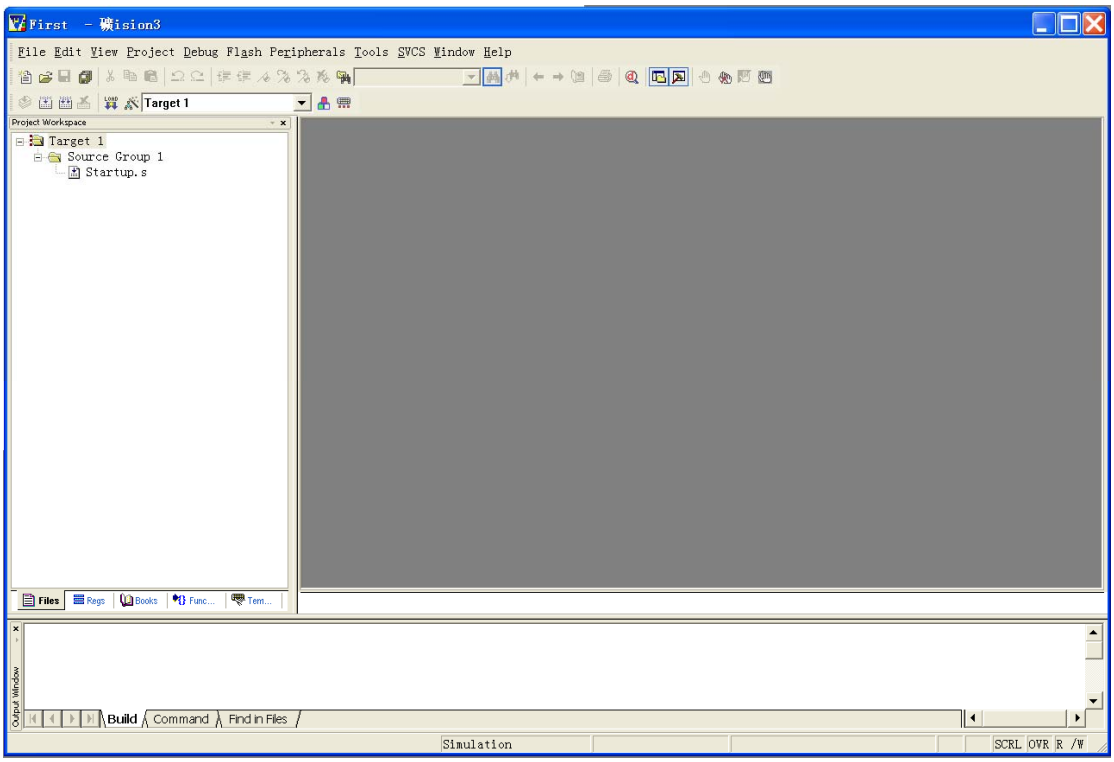


出现设备选择窗口，找到 STMicroelectronics 分类，点击前面的加号“+”展开，选择我们开发板对应的 CPU 型号，例如 STM32F103RB



点击“确定”按钮，弹出如下对话框

这是在询问是否需要把启动代码拷贝到目标文件夹并添加到项目中，我们选择“是”，这时候，开发环境已经为我们建立了一个只包含启动代码的空项目



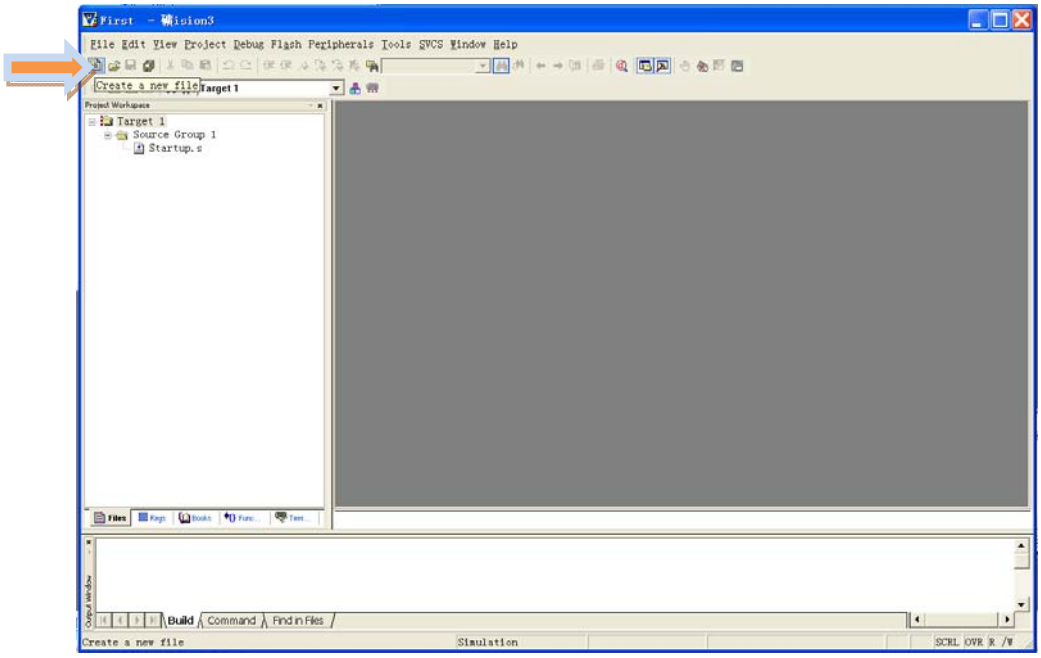
我们看到这个项目目前只包含一个汇编文件 Startup.s 是启动代码，除非非常必要，否则我们不必修改这个文件，我们只要写 C 语言就可以了，这是 Keil 环境做的方便之处。现在我们

开始编写第一个程序。

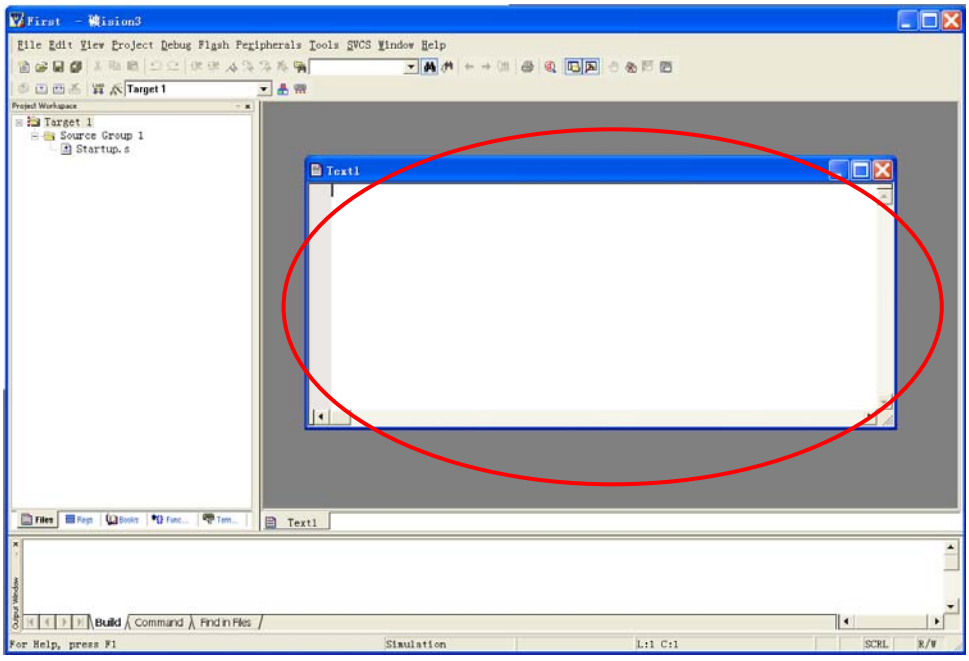
编写第一个程序

建立空白文档

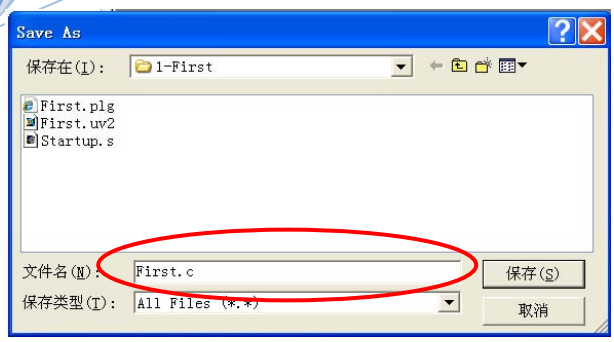
点击第一个快捷按钮“Create a new file”



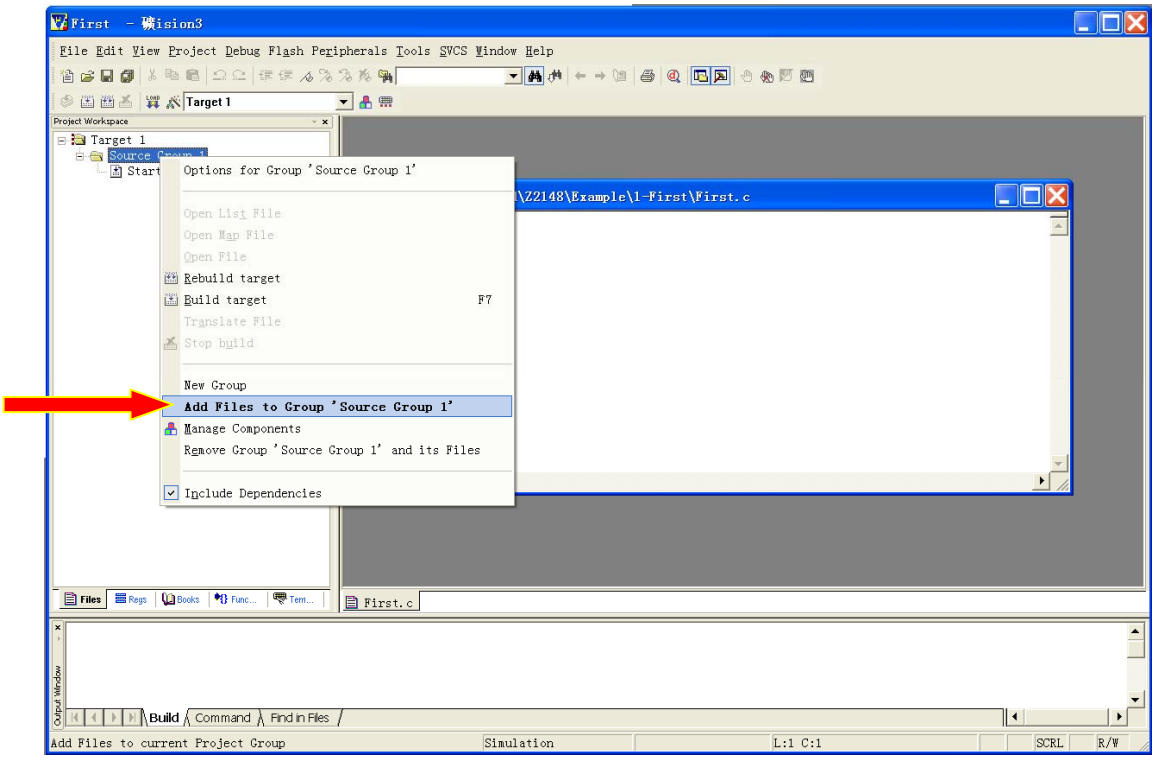
出现一个文件编辑窗口



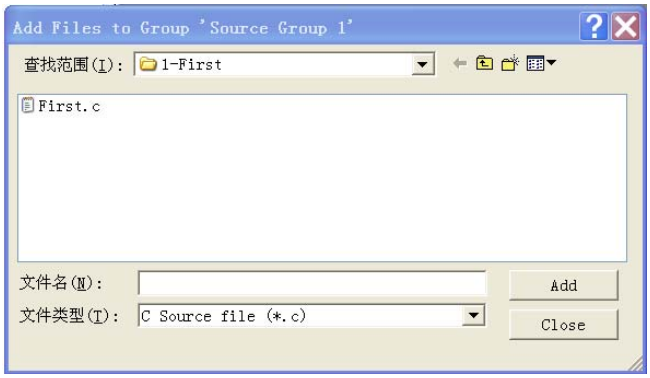
我们先存储这个文件，点击“存储”快捷按钮，弹出对话框



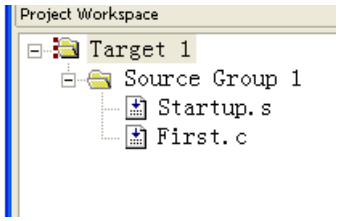
在“文件名”栏输入我们想要的名字，比如 First.c，储存。这时候这个文件并没有加入到当前项目中，我们需要手工加入。在项目窗口右击“Souce Group 1”



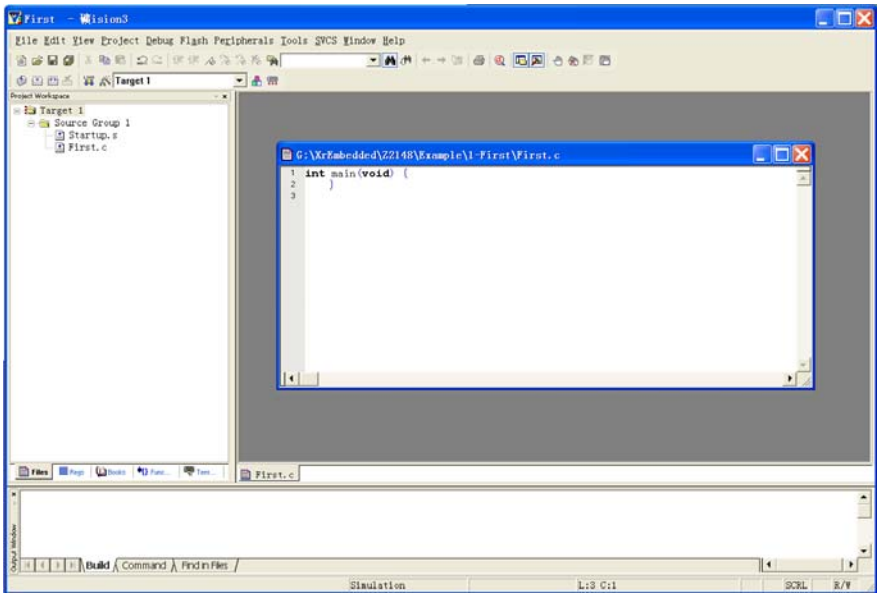
再点击“Add Files to Group “Source Group 1””



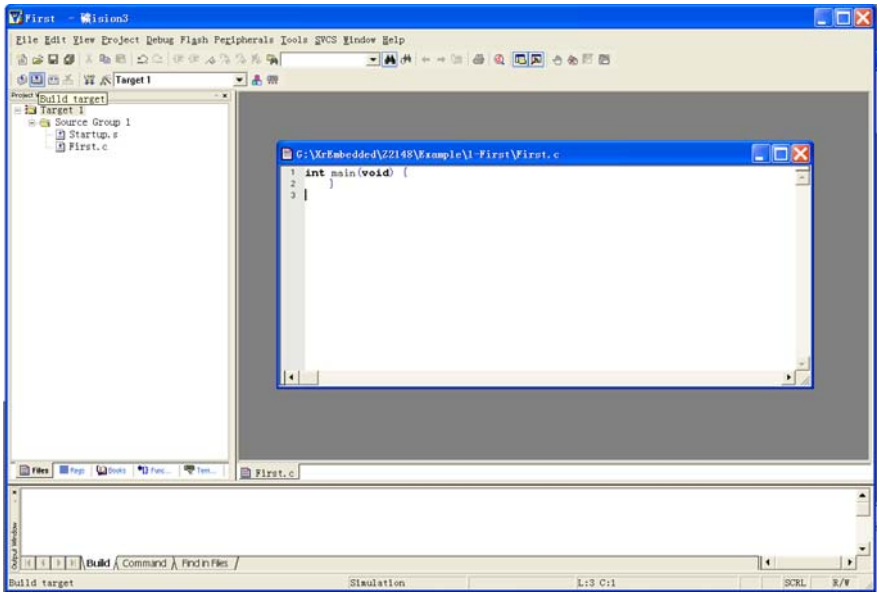
在弹出的对话框中选择我们刚才新建的文件。点击“Add”，再点击“Close”，这时，我们可以看到项目窗口里已经多了一个我们刚才建立的文件。



现在我们在编辑窗口里输入一个最简单的 C 程序



按 Build target 快捷按钮，编译



我们看到输出窗口显示如下信息

```
Build target 'Target 1'
assembling Startup.s...
compiling First.c...
linking...
```

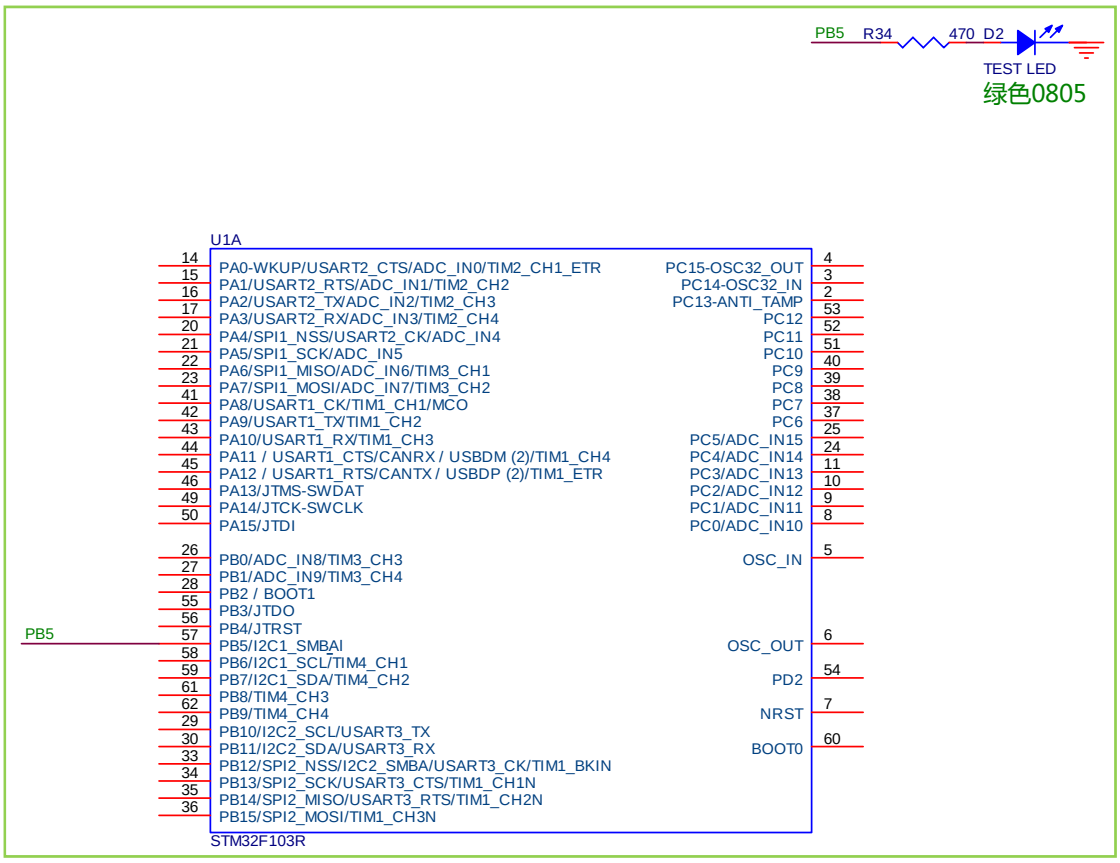
```
"First.axf" - 0 Error(s), 0 Warning(s).
```

这说明我们已经成功编译了这个最简单的 C 程序，虽然它什么也不做。接下来我们会做一个很简单但是有点作用的程序了，并且介绍如何调试。

第五章 GPIO 试验一：输出

硬件

在智林 Z32R 开发板 CPU 的 57 脚，即 PB5 上接着一个发光管 D2



在这个试验中，我们会写一个程序点亮它，目的是向大家介绍 GPIO（通用输入输出）的操作方法。

软件

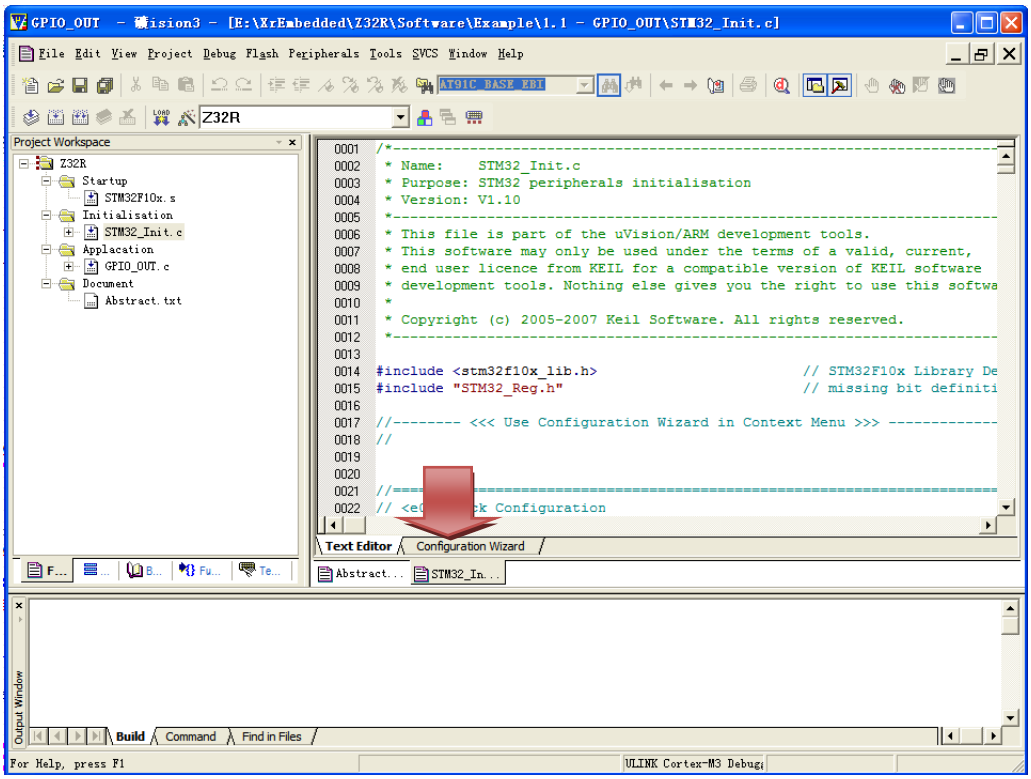
我们建立一个新项目叫 GPIO-OUT， 建立主程序 GPIO-OUT.c。输入源程序如下

```
/*-------------------------------------------------------*\
 |  引入相关芯片的头文件                                 |
\*-------------------------------------------------------*/
#include <stm32f10x_lib.h>                          // STM32F10x Library
Definitions
#include "STM32_Init.h"                            // STM32 Initialization
/*-------------------------------------------------------*\
 | HARDWARE DEFINE                                       |
\*-------------------------------------------------------*/
```

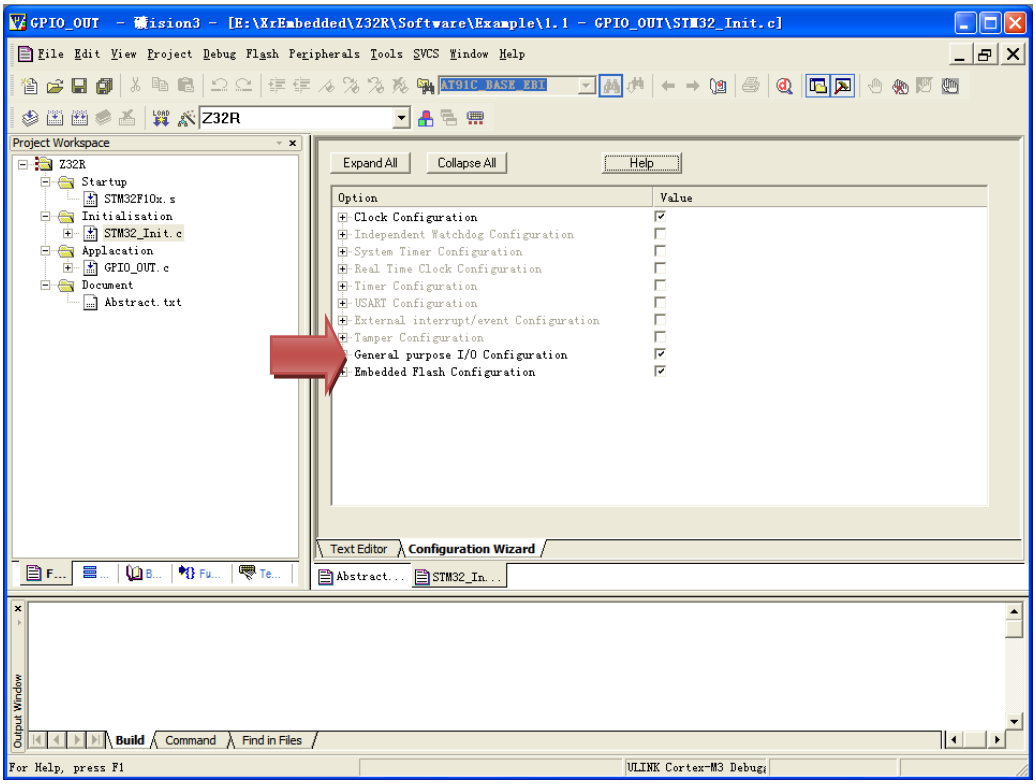
```
#define LED             ( 1 << 5 )              // PB5: LED D2
/*-------------------------------------------------------------*\
 |  Delay                                                      |
 |  延时 Inserts a delay time.                                  |
 |  nCount: 延时时间                                           |
 |  nCount: specifies the delay time length.                    |
\*-------------------------------------------------------------*/
void Delay(vu32 nCount) {
  for(; nCount != 0; nCount--);
  }
/*-------------------------------------------------------------*\
 | MIAN ENTRY                                                  |
\*-------------------------------------------------------------*/
int main (void) {
  stm32_Init ();                                // STM32 setup
  for(;;) {
    GPIOB->ODR &= ~LED;                          // switch on LED
    Delay(2000000);
    GPIOB->ODR |=  LED;                          // switch off LED
    Delay(2000000);
    }
  }
/*-------------------------------------------------------------*\
 | END OF FILE                                                 |
\*-------------------------------------------------------------*/
```

配置管脚

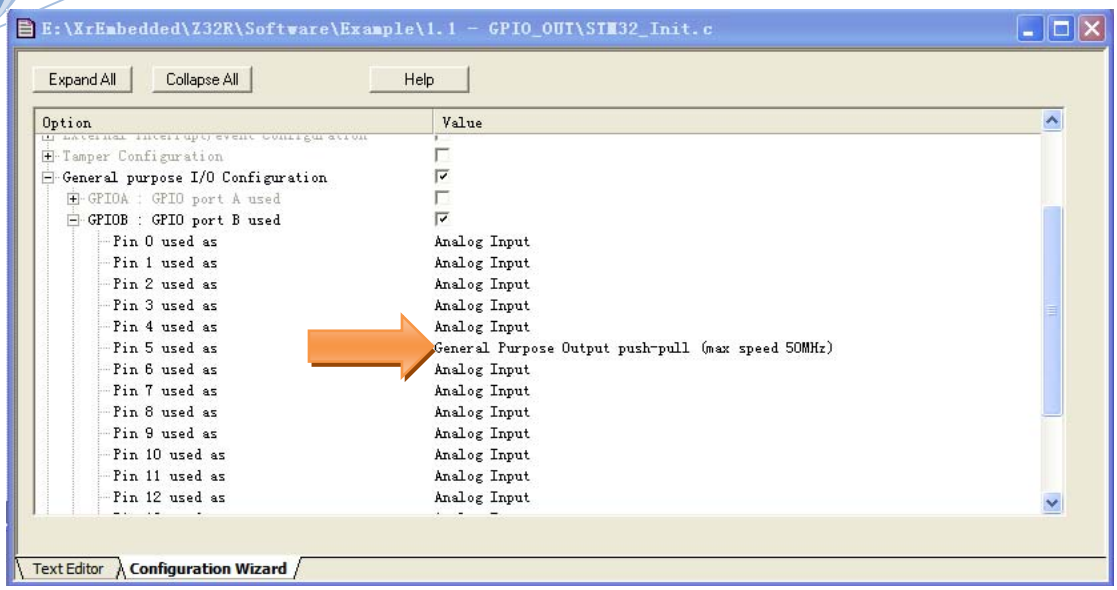
复制我们例子中的 STM32_Init.c 文件到工程目录中，并加入到工程。打开这个文件：



点击 Configuration Wizard 标签页。出现：



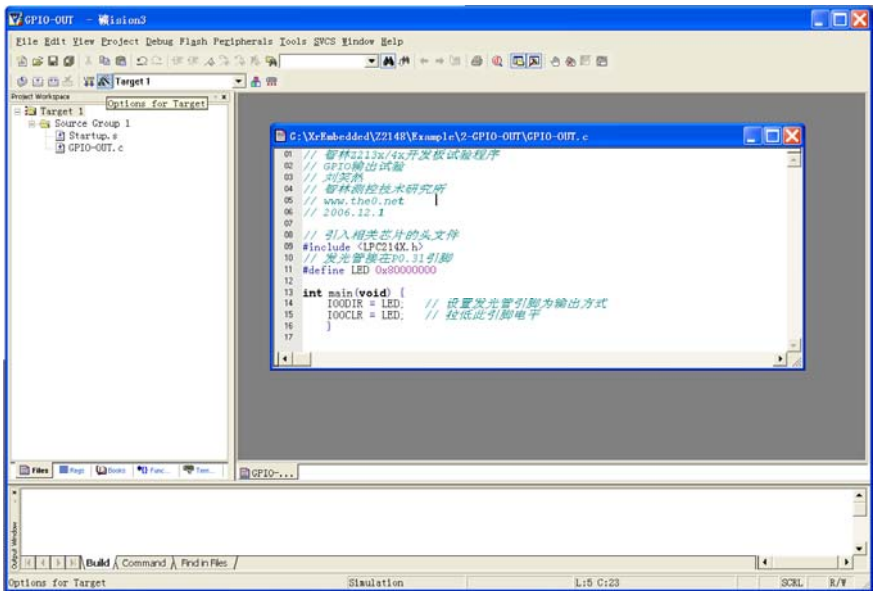
展开 General purpose I/O Configuration 后，继续展开 GPIOB:GPIO port B used



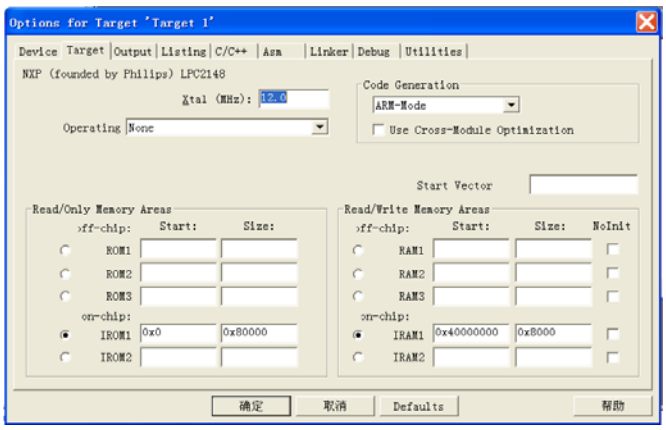
在 Pin 5 used as 后选择 General Purpose Output push-pull (max speed 50MHz)

输出路径

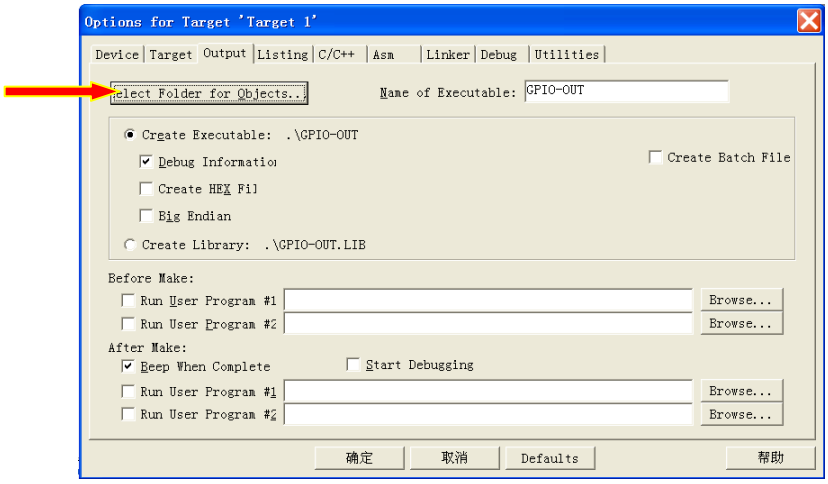
因为在编译过程中，编译器会产生很多中间文件，为了使我们的程序文件夹的文件显得不凌乱，我们可以设置输出文件的路径，点击“目标选项”快捷按钮



弹出窗口



选择 Output



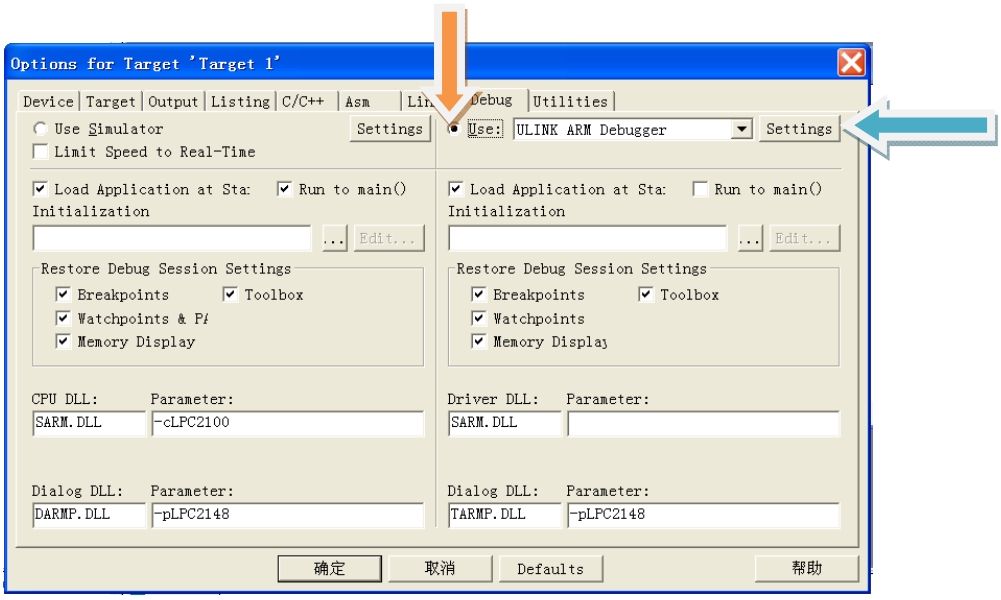
点击 Select Folder for Objects...，弹出对话框，可以新建一个 output 路径，然后选择它。同样对于 Listing 也做同样设置。然后，我们可以编译程序了。

连接仿真器

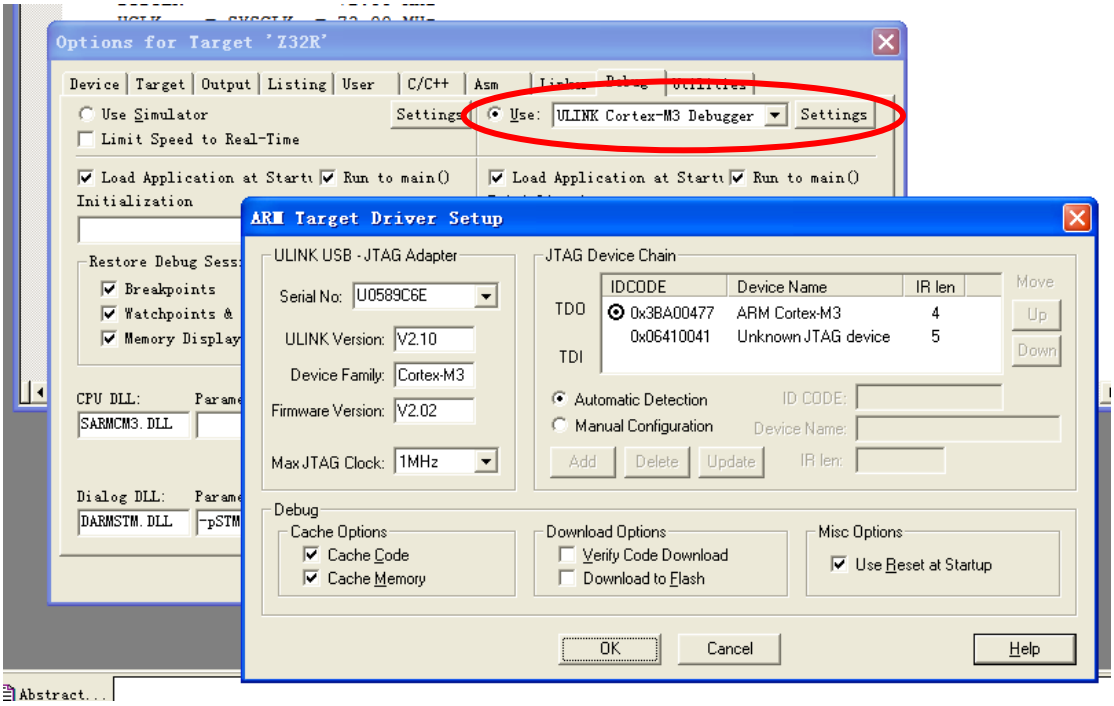
编译通过后，就要进入调试阶段了，连接仿真器和目标板的 JTAG 接口，打开目标板电源开关，把 ULink 的 USB 线插入到 PC 机。

调试准备

为了调试程序需要做一些必要的设置，点击“目标选项”快捷按钮，点选 Debug 选项卡

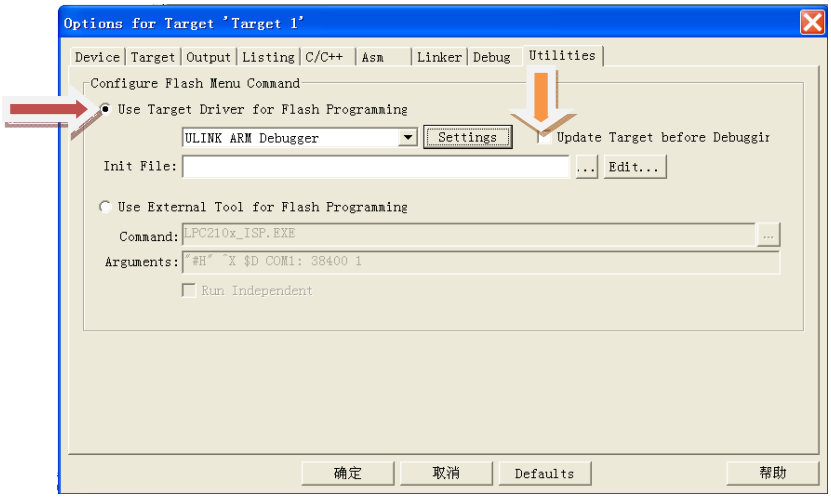


选择 Use:ULINK Cortex-M3 Debugger，然后点击 Settings，弹出窗口

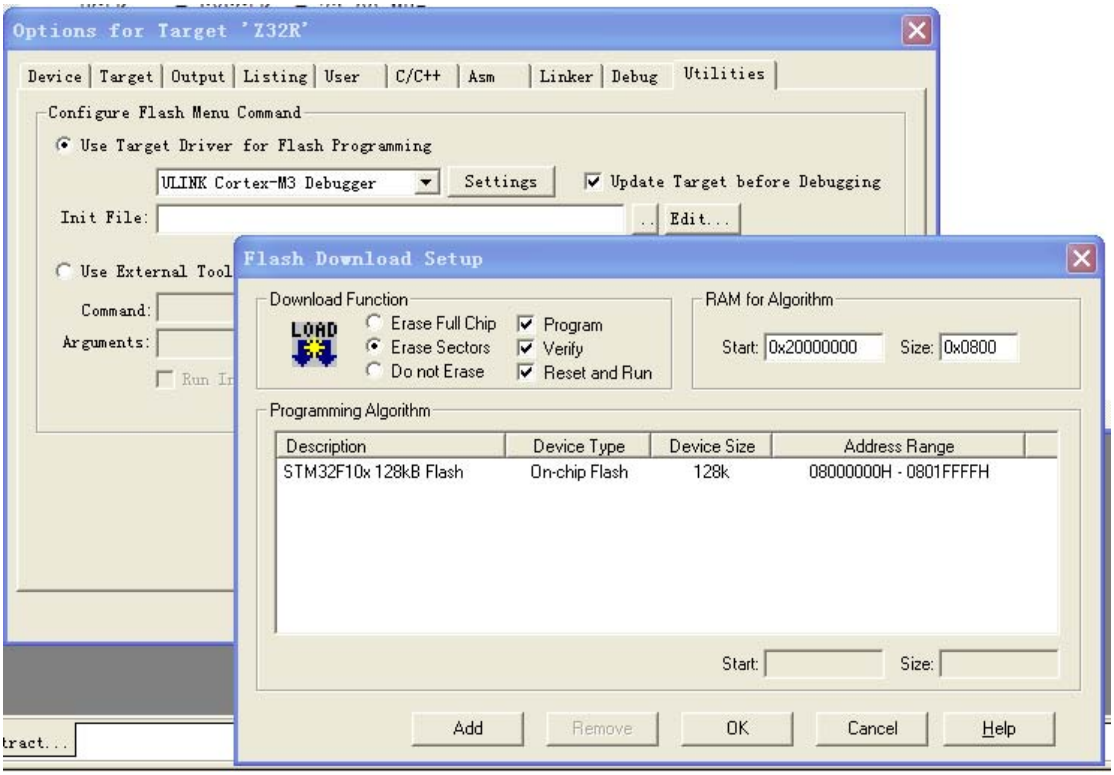


我们可以看到，在 JATG Device Chain 窗口里已经发现了 ARM Cortex-M3 的 CPU 核心，不要勾选 Verify Code Download 和 Download to Flash

点击 OK 退出，回到“目标选项”，点选 Utilities



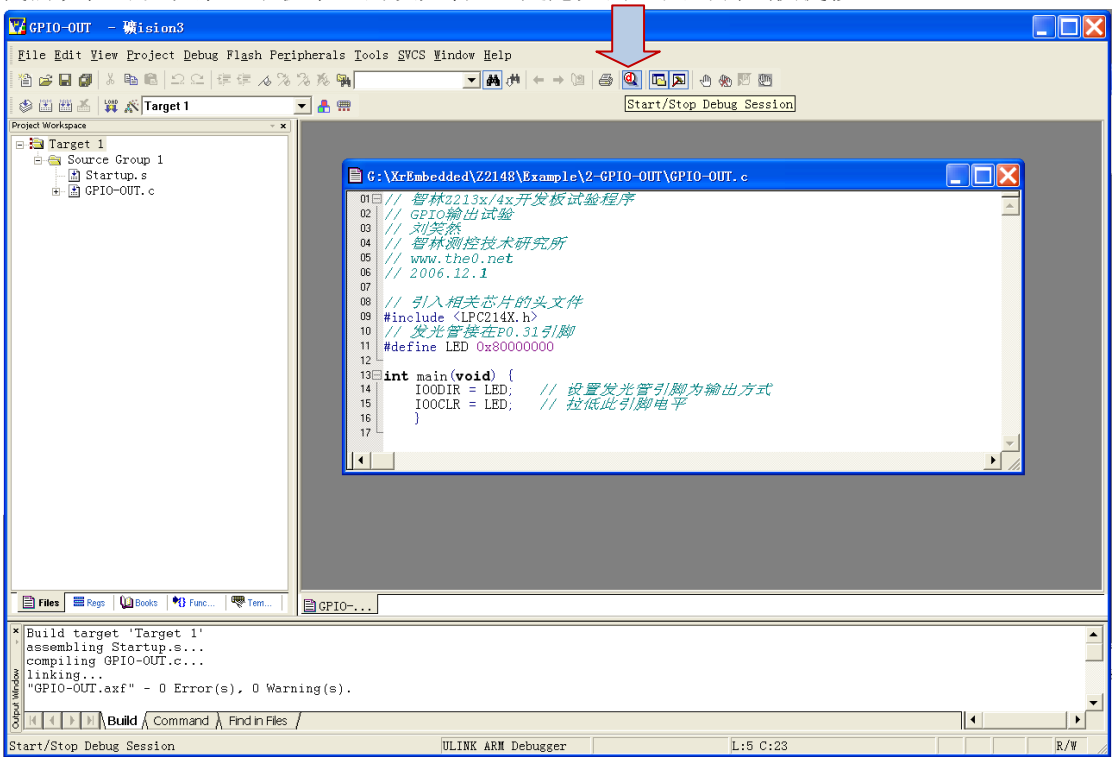
点选 Use Target Driver for Flash Programming，确定是 ULINK Cortex-M3 Debugger，勾选 Update Target before Debugging，点击 Settings 按钮：



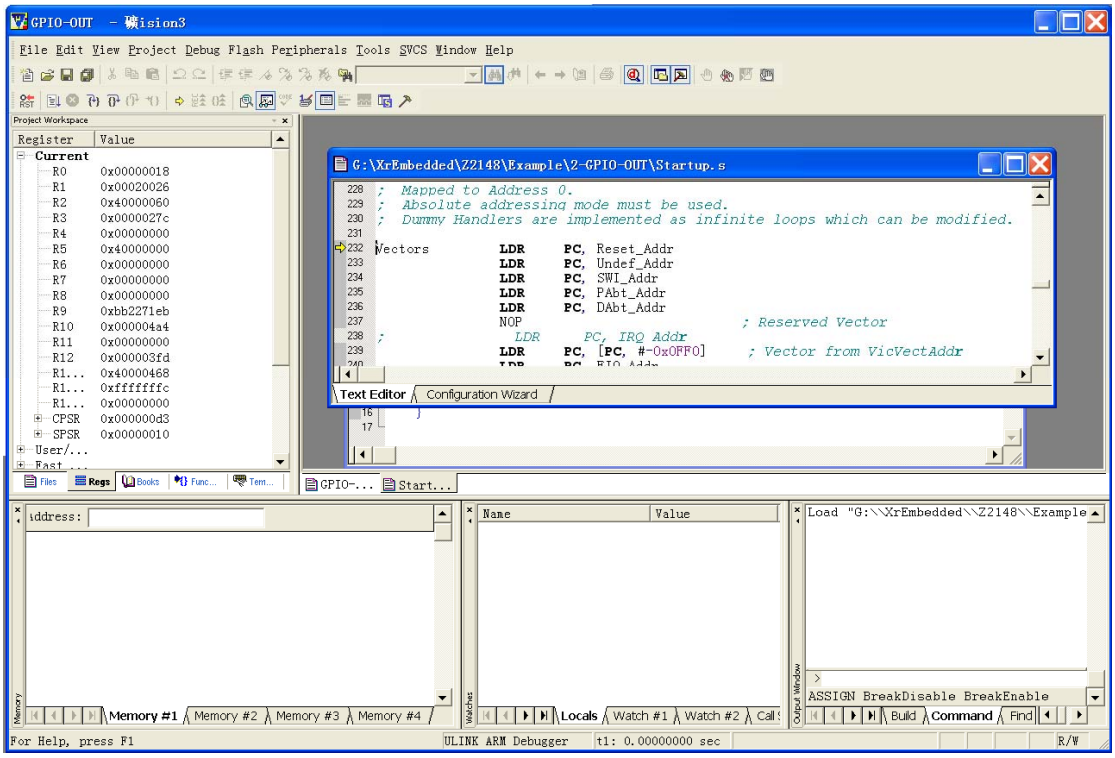
勾选 Reset and Run，这个选项可以让我们下载程序后立刻运行，方便调试。

开始调试

我们现在可以观察一下板子上的发光管 D2 是熄灭的，点击调试快捷按钮



开发环境进入了调试状态

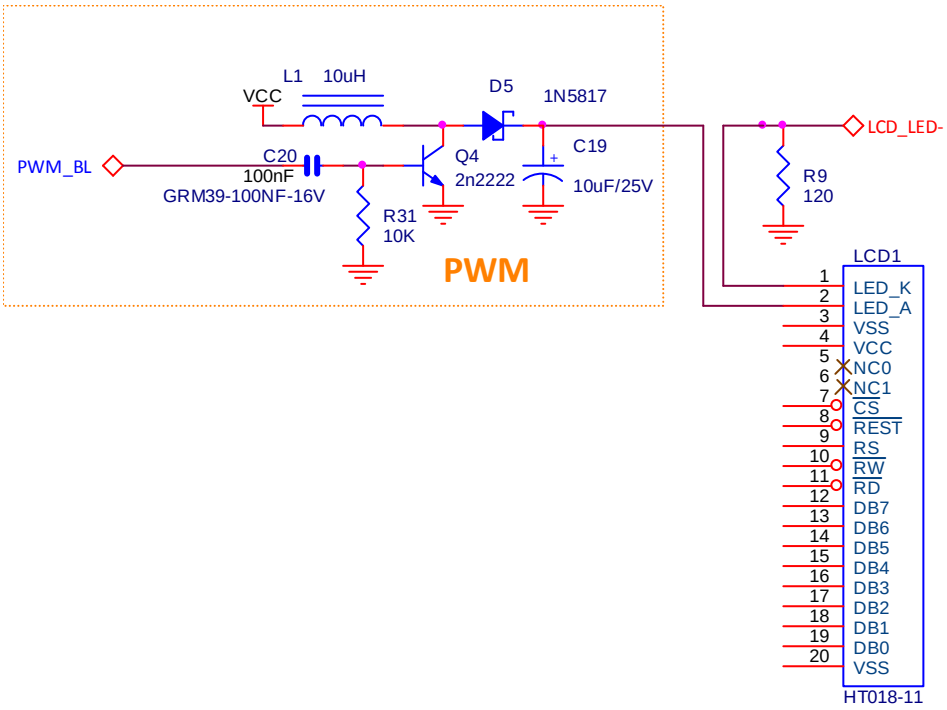


我们可以执行让程序单步、执行到光标位置等等调试动作，非常方便。现在我们把光标停在主程序开始，让程序执行到光标处，然后单步执行，可以观察发光管，可以变亮，或熄灭。

第六章 点亮液晶背光：PWM 试验

有了液晶屏，我们就可以很方便的观察显示数据，为了进行许多后续试验提供诸多方便。点亮液晶的第一步是产生液晶背光。这款液晶屏的背光是 LED 构成，需要 6-7V 的电压驱动，而开发板是 5V 供电，所以我们需要一个 DC/DC 升压电路。

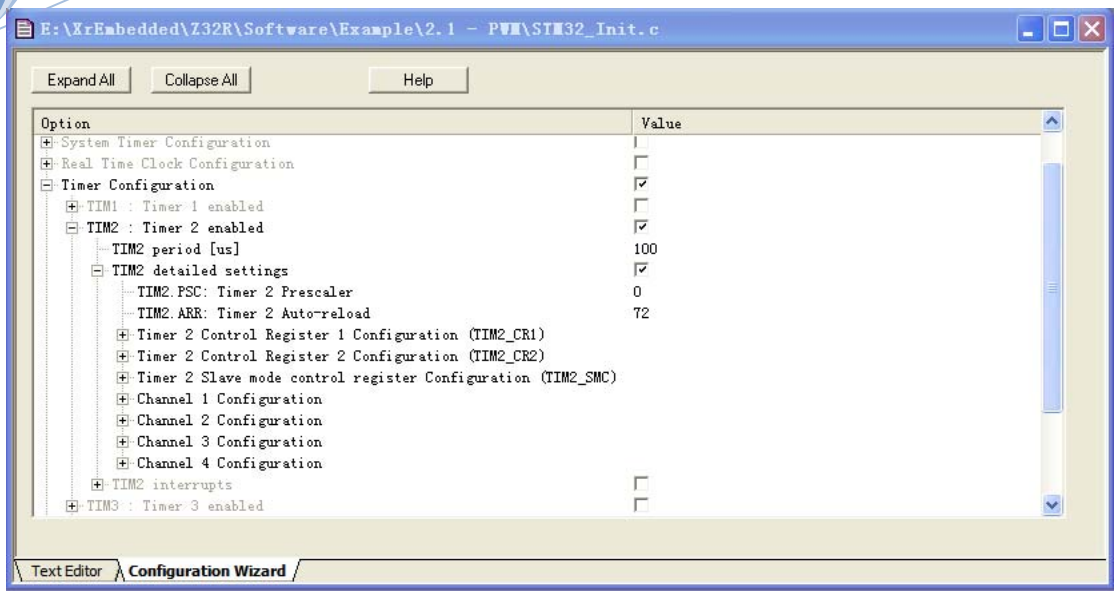
硬件



这是典型的 BOOST DC-DC 变换电路，由 CPU 产生的 PWM 脉冲控制三极管实现升压，会把输入的 5 伏电压提升到 6-7 伏，用来点亮液晶背光。

软件

在这个实验里，不需要我们写代码，只要配置一下就有了，和上个实验一样，我们配置 STM32_Init.c 文件，使用 Configuration Wizard，我们配置 TIM2 如下图：

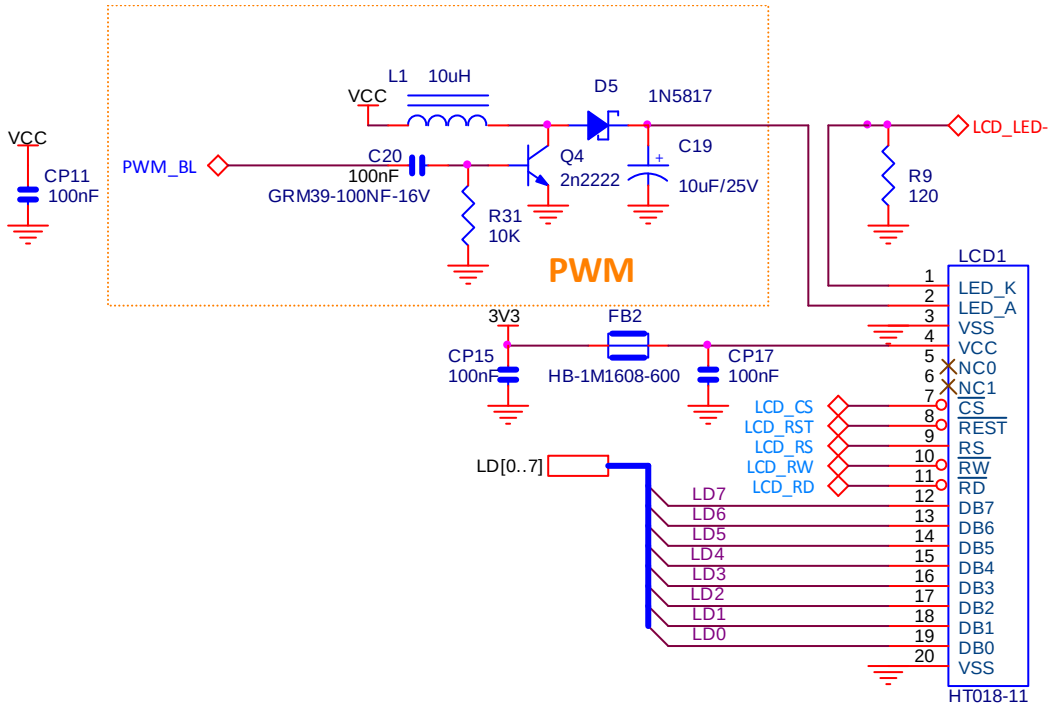


软件运行后可以看到液晶的白色背光发光管点亮了。

第七章 点亮液晶屏

智林 Z32R 开发板配置了一块彩色液晶屏，驱动芯片是 SPFD54124B.

硬件



软件 1：显示英文字符串

我们首先定义了底层硬件操作：

```
/*-------------------------------------------------------------*\
 | HARDWARE DEFINITIONS                                        |
\*-------------------------------------------------------------*/
/* PINS:
   - DB0 = PC0
   - DB1 = PC1
   - DB2 = PC2
   - DB3 = PC3
   - DB4 = PC4
   - DB5 = PC5
   - DB6 = PC6
   - DB7 = PC7
   - RD  = PC8
   - RW  = PC9
```

```
   - RS  = PC10
   - RST = PC11
   - CS  = PC12                                          */

#define PIN_CS              (   1 << 12)
#define PIN_RST             (   1 << 11)
#define PIN_RS              (   1 << 10)
#define PIN_RW              (   1 <<  9)
#define PIN_RD              (   1 <<  8)
#define PINS_CTRL           (0x1F <<  8)
#define PINS_DATA           (0xFF <<  0)
#define PINS_ALL            (PINS_CTRL | PINS_DATA)

/* Enable Clock for peripheral driving LCD pins                          */
#define LCD_CLOCK_EN        (RCC->APB2ENR |= (1 << 4)); // enable clock for GPIOC

/* pin CS  setting to 0 or 1                                             */
#define LCD_CS(x)           GPIOC->ODR = (GPIOC->ODR & ~PIN_CS) | (x ? PIN_CS : 0);
/* pin RST setting to 0 or 1                                             */
#define LCD_RST(x)          GPIOC->ODR = (GPIOC->ODR & ~PIN_RST) | (x ? PIN_RST : 0);
/* pin RS setting to 0 or 1                                              */
#define LCD_RS(x)           GPIOC->ODR = (GPIOC->ODR & ~PIN_RS) | (x ? PIN_RS : 0);
/* pin RW setting to 0 or 1                                              */
#define LCD_RW(x)           GPIOC->ODR = (GPIOC->ODR & ~PIN_RW) | (x ? PIN_RW : 0);
/* pin RD setting to 0 or 1                                              */
#define LCD_RD(x)           GPIOC->ODR = (GPIOC->ODR & ~PIN_RD) | (x ? PIN_RD : 0);

/* Reading DATA pins                                                    */
#define LCD_DATA_IN         (((GPIOC->IDR & PINS_DATA) >> 0) & 0xFF)
/* Writing value to DATA pins                                           */
#define LCD_DATA_OUT(x)     GPIOC->ODR = (GPIOC->ODR & ~PINS_DATA) | (x << 0);

/* Setting all pins to output mode                                      */
#define LCD_ALL_DIR_OUT     GPIOC->CRL = (GPIOC->CRL & 0x00000000) | 0x33333333; \
                           GPIOC->CRH = (GPIOC->CRH & 0xFFF00000) | 0x00033333;

/* Setting DATA pins to input mode                                      */
#define LCD_DATA_DIR_IN     GPIOC->CRL = (GPIOC->CRL & 0x00000000) | 0x44444444;

/* Setting DATA pins to output mode                                     */
#define LCD_DATA_DIR_OUT    GPIOC->CRL = (GPIOC->CRL & 0x00000000) | 0x33333333;
```

我们用 GPIO 模拟驱动液晶的总线时序。

```
void LCD_Data(unsigned int Data) {
```

```
  LCD_CS(0);
  LCD_RD(1);
  LCD_RS(1);
  LCD_Bus_Write_Byte( Data );
  LCD_CS(1);
  }
void LCD_Reg_Set(unsigned int Cmd,unsigned int Data) {
  LCD_CS(0);
  LCD_RD(1);
  LCD_RS(0);
  LCD_Bus_Write_Byte( Cmd );
  LCD_RS(1);
  LCD_Bus_Write_Byte( Data );
  LCD_CS(1);
  }
```

这两个函数用来写指令和数据。

剩下的工作，就是按照 SPFD54124B 的手册进行操作，大家直接读源码就可以理解了，这里就不罗嗦了。

本节的例子程序的运行结果是在液晶上显示几行英文文字。

```
int main (void) {
  stm32_Init ();                                  // STM32 setup
  LCD_Init();
  LCD_Clear_Screen(Blue);
  for(;;) {
    GPIOB->ODR &= ~LED;                           // switch on LED
    Delay(2000000);
    GPIOB->ODR |=  LED;                           // switch off LED
    Delay(2000000);

    Font = 0;
    LCD_PutString(10,30,"STM32F EVAL BOARD",Cyan,Blue);
    Font = 1;
    LCD_PutString(38,46,"Version 1.0",Green,Blue);
    Font = 0;
    LCD_PutString(30,65,"ZERO Research",Yellow,Blue);
    LCD_PutString(36,81,"www.the0.net",Magenta,Blue);
    }
  }
```

软件 2：显示汉字

我们可以定义一个所用汉字的小字库，文件为 GB1616.h，其结构如下：

```
// ------------------  汉字字模的数据结构定义 ------------------------ //
struct typFNT_GB16                  // 汉字字模数据结构
```

```
{
       unsigned char Index[3];                  // 汉字内码索引
       char Msk[32];                           // 点阵码数据
};
```

然后定义字模数据：

```
const struct typFNT_GB16 codeGB_16[] =         // 数据表
{
"智", 0x00,0x00,0x18,0x00,0x11,0x22,0x1F,0xBF,0x64,0x22,0x05,0x22,0x7F,0xA2,0x0A,0x3E,
     0x09,0xA2,0x10,0x80,0x6F,0xF8,0x08,0x08,0x0F,0xF8,0x08,0x08,0x0F,0xF8,0x08,0x08,
"林", 0x00,0x00,0x0C,0x18,0x08,0x10,0x0A,0x12,0x7F,0x7F,0x08,0x10,0x18,0x30,0x18,0x58,
     0x2C,0x54,0x2A,0x54,0x4A,0x92,0x08,0x91,0x09,0x10,0x08,0x10,0x08,0x10,0x08,0x10,
"测", 0x00,0x00,0x20,0x02,0x1B,0xE2,0x0A,0x2A,0x02,0xAA,0x42,0xAA,0x2A,0xAA,0x2A,0xAA,
     0x0A,0xAA,0x12,0xAA,0x12,0xAA,0x72,0xAA,0x10,0x82,0x11,0x42,0x12,0x2A,0x14,0x24,
"控", 0x00,0x00,0x18,0x30,0x11,0x22,0x11,0xFF,0x7D,0x52,0x12,0x54,0x10,0x91,0x14,0x8F,
     0x19,0x00,0x70,0x04,0x51,0xFE,0x10,0x20,0x10,0x20,0x10,0x22,0x73,0xFF,0x20,0x00,
… …
};
```

接着我们来做一个显示汉字的函数：

```
#include "GB1616.h"   //16*16 汉字字模

void PutGB1616(unsigned short x, unsigned short  y, unsigned char c[2],
unsigned int f,unsigned int b){
    unsigned int i,j,k;

    LCD_SetArea(x, y, x+16-1, y+16-1);
    LCD_Inst(0x2C);

    IO1DIR |= LCD_BUS_DAT;

    IO0CLR =  LCD_BUS_CS;
    IO0SET =  LCD_BUS_RD;
    IO1SET =  LCD_BUS_RS;

    for (k=0;k<49;k++) {
      if ((codeGB_16[k].Index[0]==c[0])&&(codeGB_16[k].Index[1]==c[1])){
        for(i=0;i<32;i++) {
          unsigned short m=codeGB_16[k].Msk[i];
          for(j=0;j<8;j++) {
            if((m&0x80)==0x80) {
                LCD_Bus_Write_Byte(f>>8);
                LCD_Bus_Write_Byte(f);
                }
            else {
                LCD_Bus_Write_Byte(b>>8);
```

```
                LCD_Bus_Write_Byte(b);
                }
            m<<=1;
            }
          }
        }
      }

    IO0SET =  LCD_BUS_CS;
    IO1DIR &= ~LCD_BUS_DAT;
    }
```

我们可以调用这个函数显示汉字了：

```
#include "TFT018.h"

int main() {
    BacklightOn();
    LCD_Init();
    LCD_Clear_Screen(Blue);
    for(;;) {
        Font = 0;
        LCD_PutString(10,0,"STM32 开发板",Cyan,Blue);
        LCD_PutString(0,16,"STM32 Dev Board",Cyan,Blue);
        Font = 1;
        LCD_PutString(38,46,"Version 1.0",Green,Blue);
        Font = 0;
        LCD_PutString(10,60,"智林测控技术研究所",Yellow,Blue);
        LCD_PutString(30,80,"ZERO Research",Yellow,Blue);
        LCD_PutString(36,100,"www.the0.net",Magenta,Blue);
        }
    }
```

软件 3：显示位图 BMP

这里我们使用了 ARM 汇编器的一个功能，就是可以直接引入二进制文件，我们在 BMP.s 汇编文件中引入：

```
                PRESERVE8
; ------------------------ Load bmp picture ------------------------
    AREA    |subr|,DATA,READONLY
    EXPORT      bmp
bmp
;   incbin seaside.BMP
;   incbin tree.BMP
;   incbin beauty.BMP
```

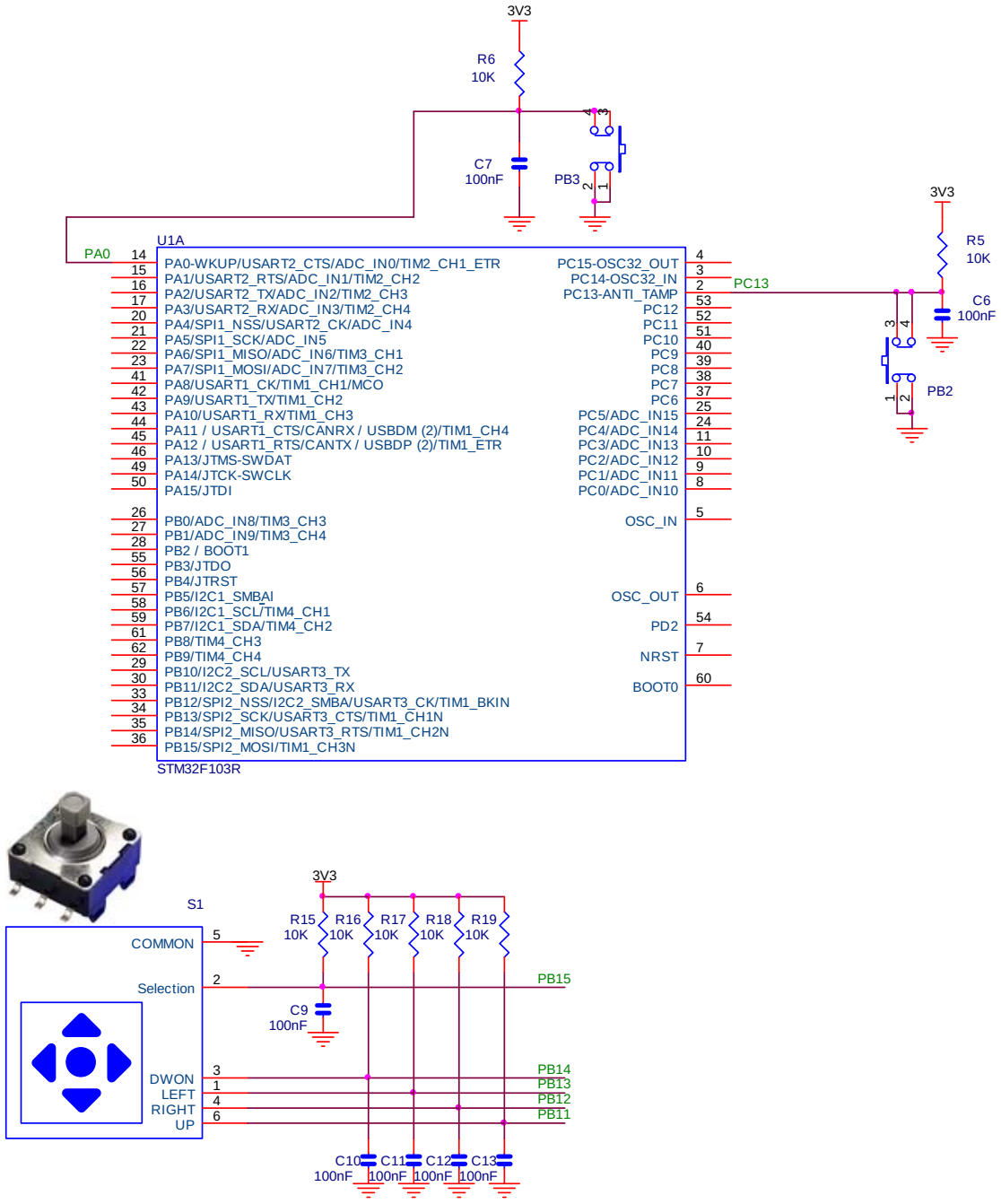
```
;   incbin girl.BMP
;   incbin sunflower.BMP
    incbin boat.BMP
;   incbin lake.BMP
;   incbin sweetdays.BMP


                END
```

我们准备好了几个位图，大家可以把分号去掉，看看别的图片，但是每次只能有一个图片打开，其他的前面都要加上分号注释掉。

第八章 GPIO 试验二：输入

智林 Z32R 开发板带有一个游戏杆和 2 个按钮开关作为开关量输入试验用。

硬件



软件 1：输入实验

当然，我们需要在 STM32_Init.c 的 Configuration Wizard 中配置相应的端口为输入。

```
int main (void) {
  stm32_Init ();                                    // STM32 setup
  LCD_Init();
  LCD_Clear_Screen(Blue);
  for(;;) {
    Font = 0;
    LCD_PutString(30,0,"STM32F 开发板",Cyan,Blue);
    LCD_PutString(30,16,"GPIO 输入演示",Red,Yellow);
    Font = 1;
    LCD_PutString(38,32,"Version 1.0",Green,Blue);
    Font = 0;
    LCD_PutString(10,94,"智林测控技术研究所",Yellow,Blue);
    LCD_PutString(36,111,"www.the0.net",Magenta,Blue);

    if (((GPIOB->IDR & JOYSTICK) == JOYSTICK ))  {   // Check if JOYSTICK key is pressed
      GPIOB->ODR &= ~LED;                              // switch off LED
      }
    else  {                                           // JOYSTICK key is pressed
      GPIOB->ODR |=  LED;                              // switch on LED
      }

    if (((GPIOB->IDR & UP) == 0 ))                  // Check if UP is pressed
      LCD_PutString(48,40,"上",Yellow,Red);
    else
      LCD_PutString(48,40,"上",Red,Yellow);
    if (((GPIOB->IDR & DOWN) == 0 ))                // Check if DOWN is pressed
      LCD_PutString(48,76,"下",Yellow,Red);
    else
      LCD_PutString(48,76,"下",Red,Yellow);
    if (((GPIOB->IDR & LEFT) == 0 ))                // Check if LEFT is pressed
      LCD_PutString(30,58,"左",Yellow,Red);
    else
      LCD_PutString(30,58,"左",Red,Yellow);
    if (((GPIOB->IDR & RIGHT) == 0 ))               // Check if RIGHT is pressed
      LCD_PutString(66,58,"右",Yellow,Red);
    else
      LCD_PutString(66,58,"右",Red,Yellow);
    if (((GPIOB->IDR & OK) == 0 ))                  // Check if OK is pressed
      LCD_PutString(48,58,"OK",Yellow,Red);
    else
```

```
        LCD_PutString(48,58,"OK",Red,Yellow);


    if (((GPIOC->IDR & BP2) == 0 ))                    // Check if BP2 is not pressed
        LCD_PutString(100,40,"BP2",Yellow,Red);
    else
        LCD_PutString(100,40,"BP2",Red,Yellow);
    if (((GPIOA->IDR & BP3) == 0 ))                    // Check if BP3 is not pressed
        LCD_PutString(100,76,"BP3",Yellow,Red);
    else
        LCD_PutString(100,76,"BP3",Red,Yellow);
    }
}
```

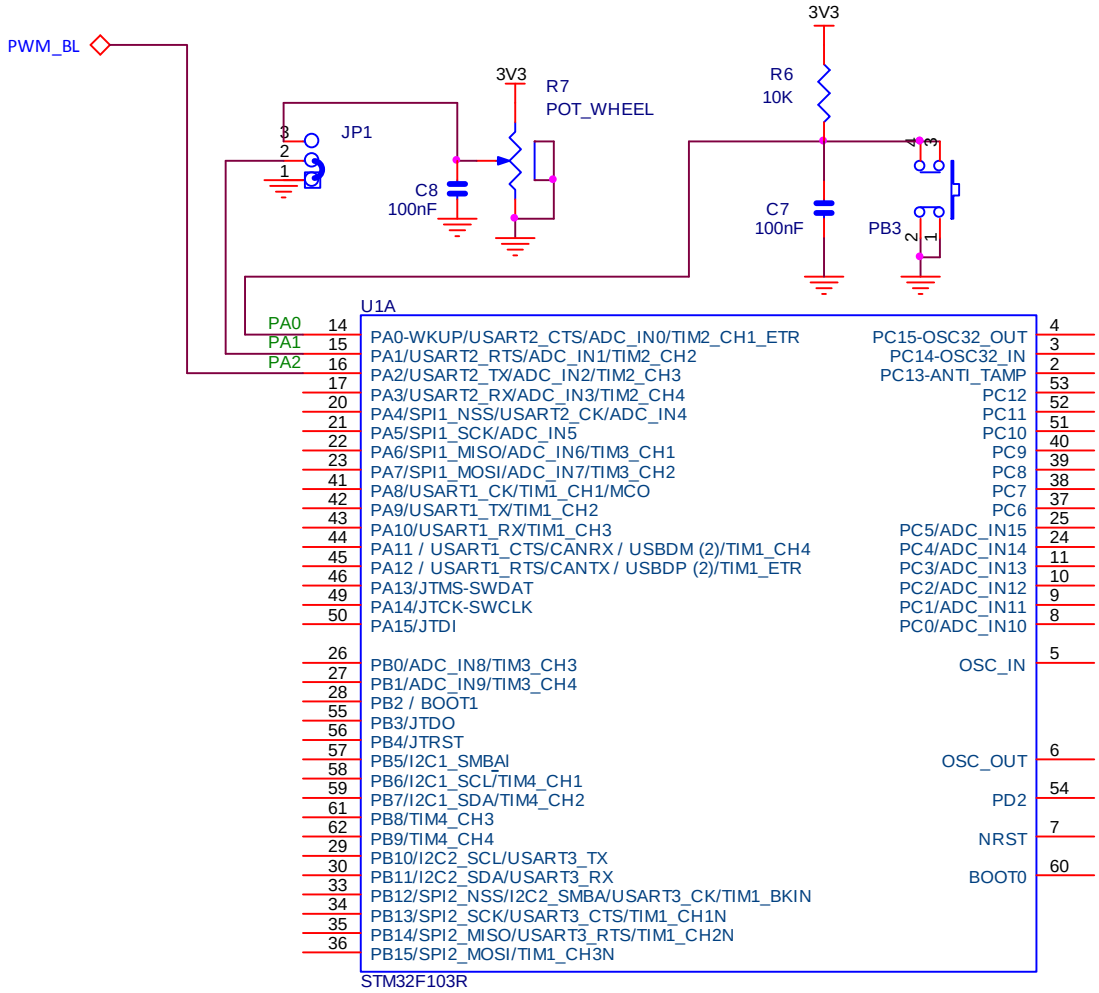
这段程序运行后，按下游戏杆的任何一个按键，发光管都会发光，在液晶上有相应的位置变成红色。

软件 1：游戏

有了液晶显示和输入做基础我们可以写个小游戏：贪吃蛇。大家自己看看吧。

第九章 ADC 试验

硬件



需要短接跳线 JP1 的 2、3 脚。我们读取 PA1、2、3 脚的电压。

软件

我们通过 DMA 读取 ADC 的结果，初始化如下

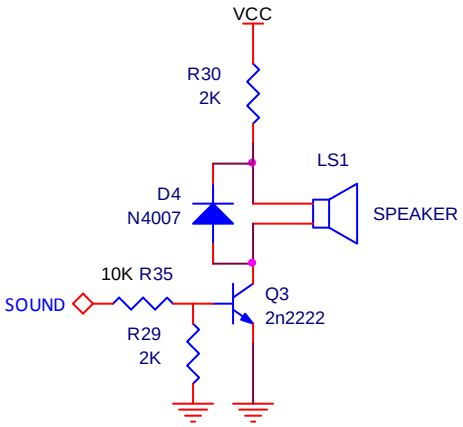
```
void adc_Init (void) {

//   GPIOA->CRL &= ~0x0000000F;                    /* set PIN1 analog input (see
stm32_Init.c) */

  RCC->AHBENR |= (1<<0);                        /* enable periperal clock for DMA       */
```

```
  DMA_Channel1->CMAR  = (unsigned long)&analog; /* set chn1 memory address     */
  DMA_Channel1->CPAR  = (unsigned long)&(ADC1->DR);     /* set chn1 peripheral address
*/
  DMA_Channel1->CNDTR = 3;                    /* transmit 3 words                    */
//   DMA_Channel1->CCR    =  0x00002520;                /*  configure  DMA  channel  1
*/
  DMA_Channel1->CCR   = 0x000025A0;           /* configure DMA channel 1             */
                                             /* circular mode, memory increment mode */
                                             /* memory & peripheral size 16bit      */
                                             /* channel priotity high               */
  DMA_Channel1->CCR  |= (1 << 0);             /* enable DMA Channe1                  */

  RCC->APB2ENR |= (1<<9);                     /* enable periperal clock for ADC1     */

  ADC1->SQR1  = 0x00200000;                   /* three conversions                   */
  ADC1->SQR3  = (3<<10) | (2<<5) | (1<<0);    /* set order to chn1 - chn2 - chn3     */
  ADC1->SMPR2 = (5<< 9) | (5<<6) | (5<<3);    /* set sample time (55,5 cycles)       */

  ADC1->CR1   =  0x00000100;                  /* use independant mode, SCAN mode     */
  ADC1->CR2   =  0x000E0103;                  /* data align right, cont. conversion  */
                                             /* EXTSEL = SWSTART                    */
                                             /* enable ADC, DMA mode                */
  ADC1->CR2  |=  0x00500000;                  /* start SW conversion                 */
  }
```

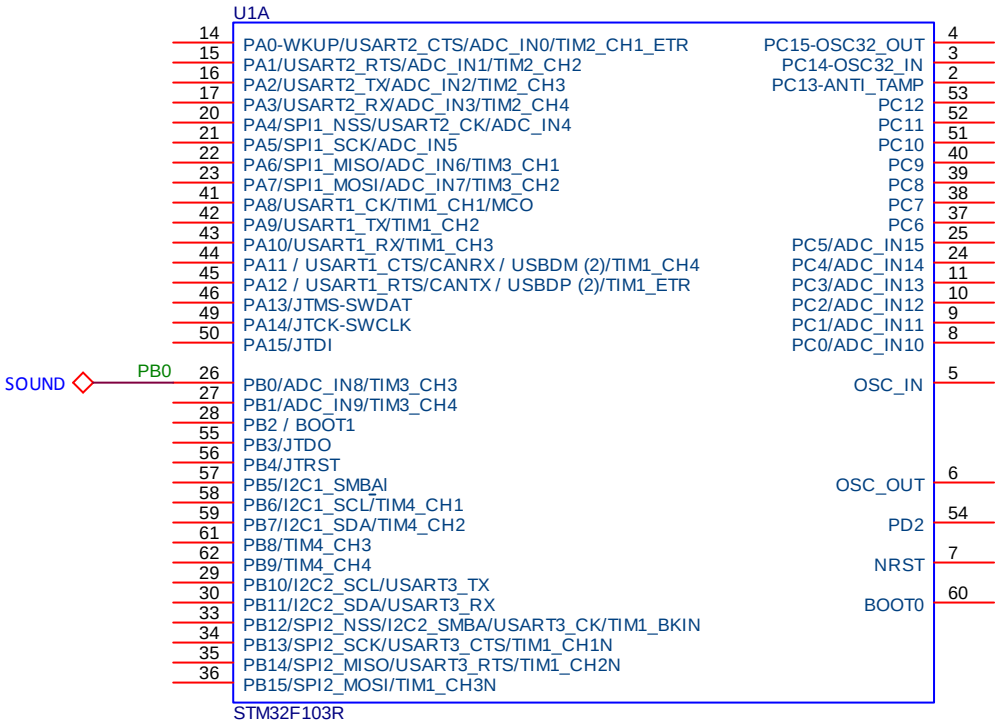
软件运行后，我们调节电位器 R7，在液晶屏上可以看到电压显示。

第十章 PWM 输出试验：电子琴

硬件



SOUND 接到 CPU 的 PB0 上，即定时器 3 的第 3 通道：



软件

在上一章的基础上我们通过电位器的电压控制扬声器的频率，以下是程序的主要代码：

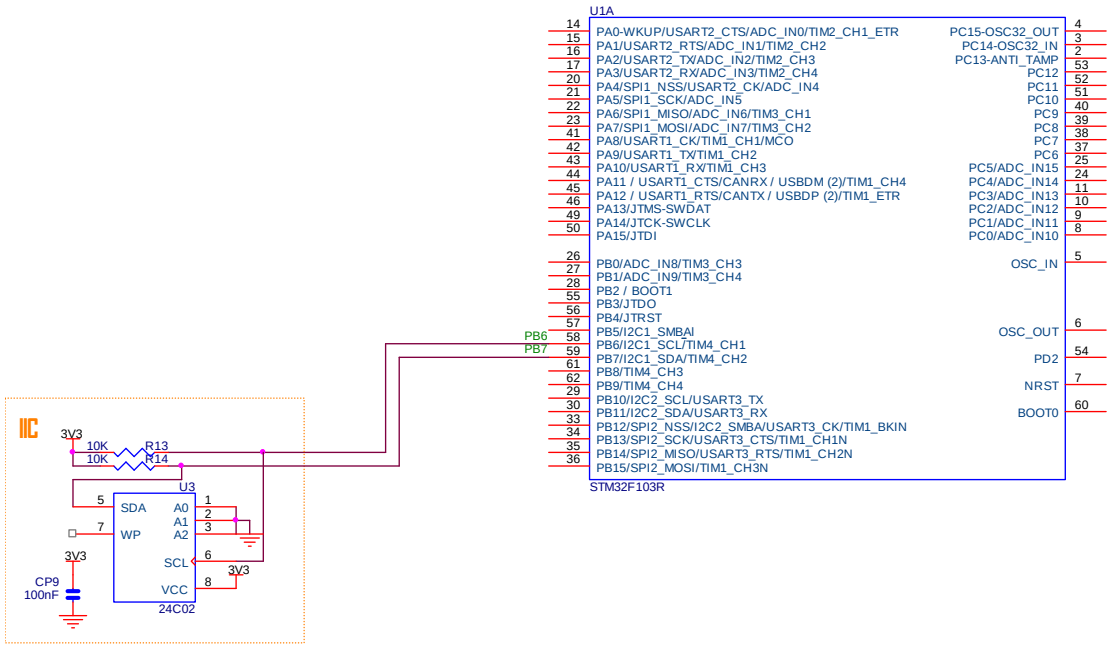
```
TIM3->ARR = i;
TIM3->CCR3 = i/2;
```

这里控制了输出频率。当然，我们需要在 STM32_Init.c 的 Configuration Wizard 中配置 PWM，

大家自己打开例子看吧。

第十一章 I2C 试验：读写 EEPROM

硬件



软件

采用查询方式的 I2C 子程序：

```
/*-------------------------------------------------------------*\
 | SOFTWARE DATA                                               |
\*-------------------------------------------------------------*/
#define I2C_Speed     100000
#define I2C_OwnAddr   0xA0

I2C_InitTypeDef i = {
     I2C_Mode_I2C,
     I2C_DutyCycle_2,
     I2C_OwnAddr,
     I2C_Ack_Enable,
     I2C_AcknowledgedAddress_7bit,
     I2C_Speed
     };
/*-------------------------------------------------------------*\
 | I2C Initialisation                                          |
\*-------------------------------------------------------------*/
```

```
void I2C_Initialisation( void ) {
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
  /* Enable I2C1 clocks */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C1, ENABLE);
  I2C_DeInit(I2C1);
  I2C_Init(I2C1, &i);

  I2C_Cmd (I2C1, ENABLE);
  }
u32 waitForEvent(u32 event) {
  int i;
  for(i = 2000000; i > 0; i--) {
    u32 s = I2C1->SR1;
    //if (s & (I2C_FLAG_TIMEOUT | I2C_FLAG_PECERR | I2C_FLAG_OVR | I2C_FLAG_AF |
I2C_FLAG_ARLO | I2C_FLAG_BERR | I2C_FLAG_STOPF));
    if (s & event) return s;
    }
  return 0;
  }
void sendStart(void) {
    I2C_GenerateSTART(I2C1, ENABLE);                    // send start when bus
becomes available
    waitForEvent(I2C_FLAG_SB);
    I2C1->DR = I2C_OwnAddr;                              //  send  slave
address for transmission
    waitForEvent(I2C_FLAG_ADDR);
    I2C1->SR2;                                          // clear event
  }
void sendData(u8 data){
  //waitForEvent(I2C_FLAG_TXE);
  waitForEvent(I2C_FLAG_BTF);
  I2C1->DR = data;                                      // send byte
  }
void sendStop(void) {
  waitForEvent(I2C_FLAG_TXE);
  I2C_GenerateSTOP(I2C1, ENABLE);                   // send stop after current
byte
  }
u8 receiveByte(void) {
  I2C_GenerateSTART(I2C1, ENABLE);                      // send start when bus
becomes available
  waitForEvent(I2C_FLAG_SB);
  I2C_AcknowledgeConfig(I2C1, DISABLE);                 // only one byte will be
read
```

```
  I2C1->DR = I2C_OwnAddr | 0x01;                          // send slave address
for reception
  waitForEvent(I2C_FLAG_ADDR);
  I2C1->SR2;                                              // clear event

  waitForEvent(I2C_FLAG_RXNE);
  I2C_GenerateSTOP(I2C1, ENABLE);                         // send stop after current
byte
  return I2C1->DR;                                        // receive byte
  }
/*-------------------------------------------------------------*\
 | I2C Write Byte                                            |
\*-------------------------------------------------------------*/
void WriteByte(u16 addr, u8 data) {
  /* Enable I2C1 acknowledgement if it is already disabled by other function */
  //I2C_AcknowledgeConfig(I2C1, ENABLE);
  //I2C_AcknowledgeConfig(I2C1, DISABLE);                 // only one byte will be
read

  sendStart();
  sendData( addr & 0xFF );
  //I2C_AcknowledgeConfig(I2C1, DISABLE);                 // only one byte will be
read
  sendData( data );
  waitForEvent(I2C_FLAG_BTF);
  sendStop();
  }
/*-------------------------------------------------------------*\
 | I2C Read Byte                                             |
\*-------------------------------------------------------------*/
u8 ReadByte(u16 addr) {
  /* Enable I2C1 acknowledgement if it is already disabled by other function */
  //I2C_AcknowledgeConfig(I2C1, ENABLE);

  sendStart();
  sendData( addr & 0xFF );
  //sendStart();
  //sendStop();
  return receiveByte();
  }
/*-------------------------------------------------------------*\
 |  Delay                                                    |
 |  延时 Inserts a delay time.                               |
 |  nCount: 延时时间                                         |
```

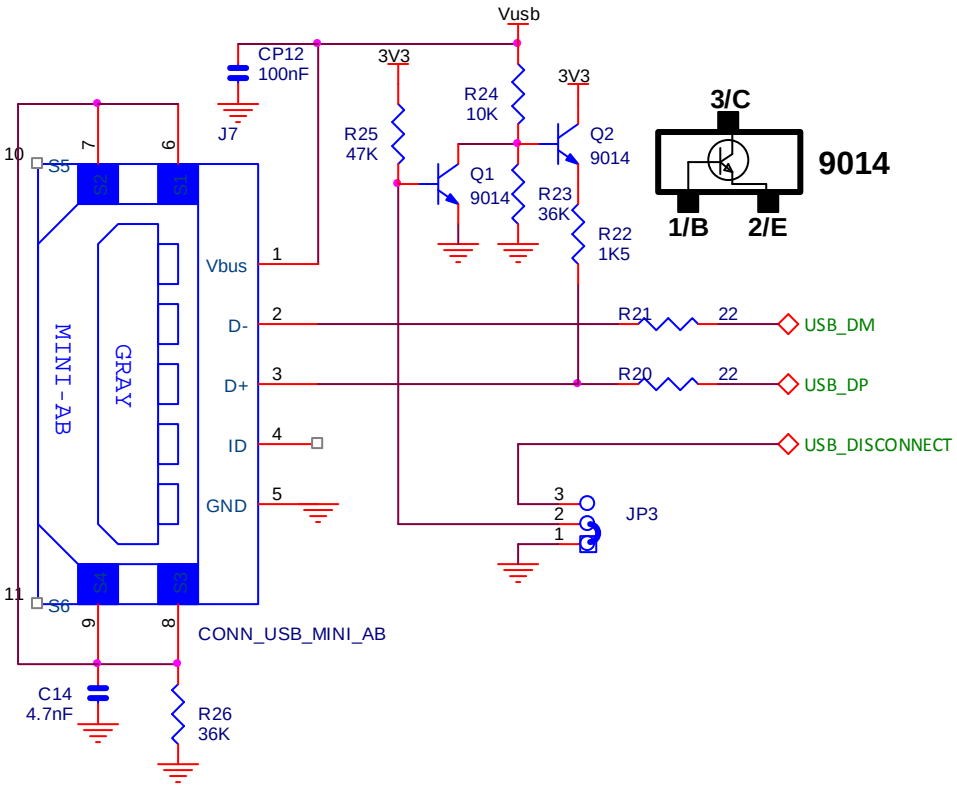
```
 |  nCount: specifies the delay time length.                  |
\*-----------------------------------------------------------*/
void Delay(vu32 nCount) {
  for(; nCount != 0; nCount--);
  }
/*-----------------------------------------------------------*\
 | MIAN ENTRY                                                 |
\*-----------------------------------------------------------*/
int main (void) {
  char s[20];

  stm32_Init ();                                  // STM32 setup
  LCD_Init();
  I2C_Initialisation();

  LCD_Clear_Screen(Blue);
  Font = 0;
  LCD_PutString(30,0,"STM32F 开发板",Cyan,Blue);
  LCD_PutString(10,113,"智林测控技术研究所",Yellow,Blue);
  LCD_PutString(40,20,"I2C Test",Blue,Cyan);
  LCD_PutString(5,38,"24C02 Address 0x00",Cyan,Blue);

  LCD_PutString(4,62,"Write:0x55",Red,Yellow);
  WriteByte(0x00, 0x55);
  Delay(30000);
  sprintf(s, "Read:0x%2X", ReadByte(0x00) );
  LCD_PutString(4,82,s,Red,Yellow);

  for(;;) {
    GPIOB->ODR &= ~LED;                           // switch on LED
    Delay(2000000);
    GPIOB->ODR |=  LED;                           // switch off LED
    Delay(2000000);
    }
  }
/*-----------------------------------------------------------*\
 | END OF FILE                                                |
\*-----------------------------------------------------------*/、
```

程序运行后，我们可以在液晶屏上看到写入 0x55，读出 0x55。

第十二章 USB 接口

STM32F103 系列处理器带有片上 USB2.0 设备接口，硬件连接非常简单。

硬件



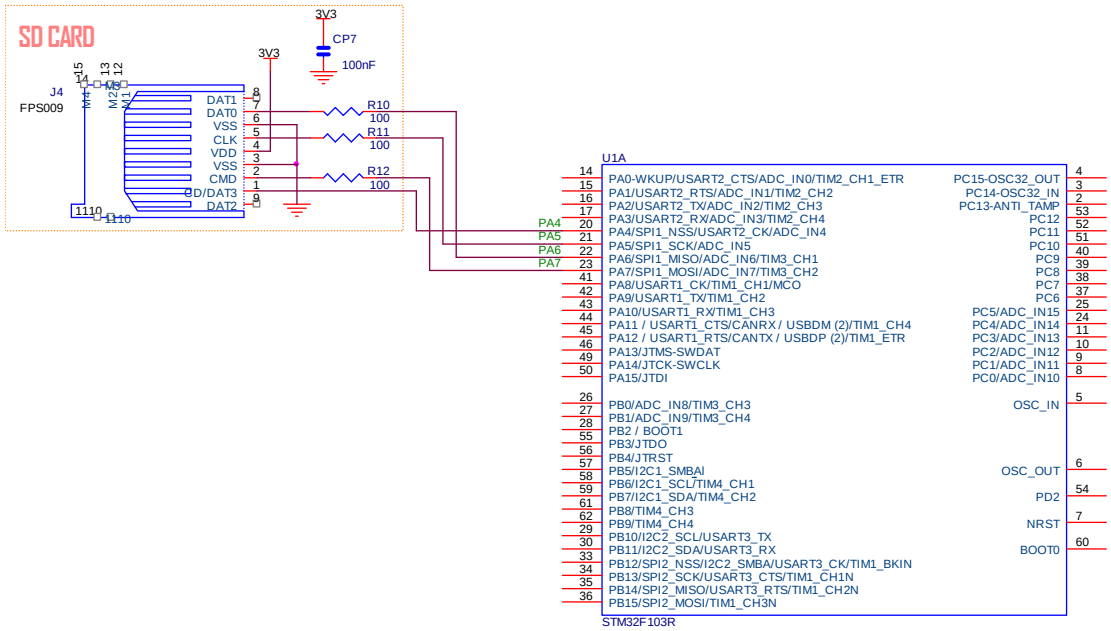
做 USB 试验时，要短接跳线 JP3 的 2、3。

软件

USB 软件很复杂，好在 Keil 公司提供了可以立即投入使用的协议栈，例子程序模拟了一个 U 盘，上电后，插入 USB 口，Windows 自动识别。

第十三章 SD 卡试验

硬件



软件

这个例子里我们使用了开源文件系统 FatFS（http://elm-chan.org/fsw/ff/00index_e.html）来操作 SD 卡。

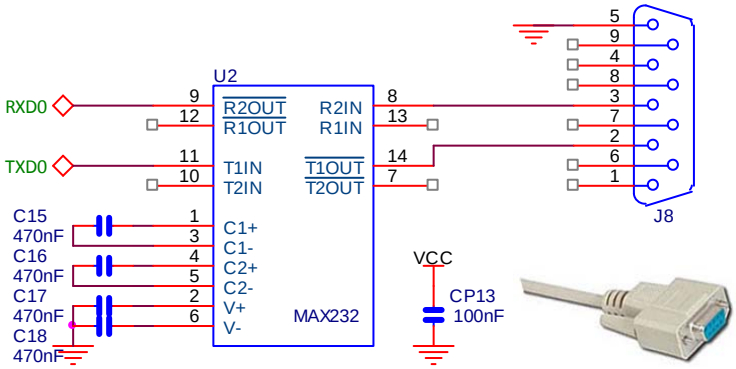
看主要代码：

```
f_mount(0, &fs);

fres = f_open (&F, filename, FA_READ | FA_WRITE | FA_CREATE_ALWAYS);
//printf("status f_open: %d\n\r", fres); //status 0 = success
f_sync (&F);

fres = f_write (&F, buf1, sizeof (buf1) - 1, &bytesWritten);
f_sync (&F);
//printf("status f_write: %d\n\r", fres); //status 0 = success

f_close (&F);
f_mount(0, NULL);
```

我们建立了一个文件并写入一段文字，我们插入 SD 卡，程序运行后，就会多了一个 test.txt 的文件，我们可以用读卡器在 PC 机上看到。

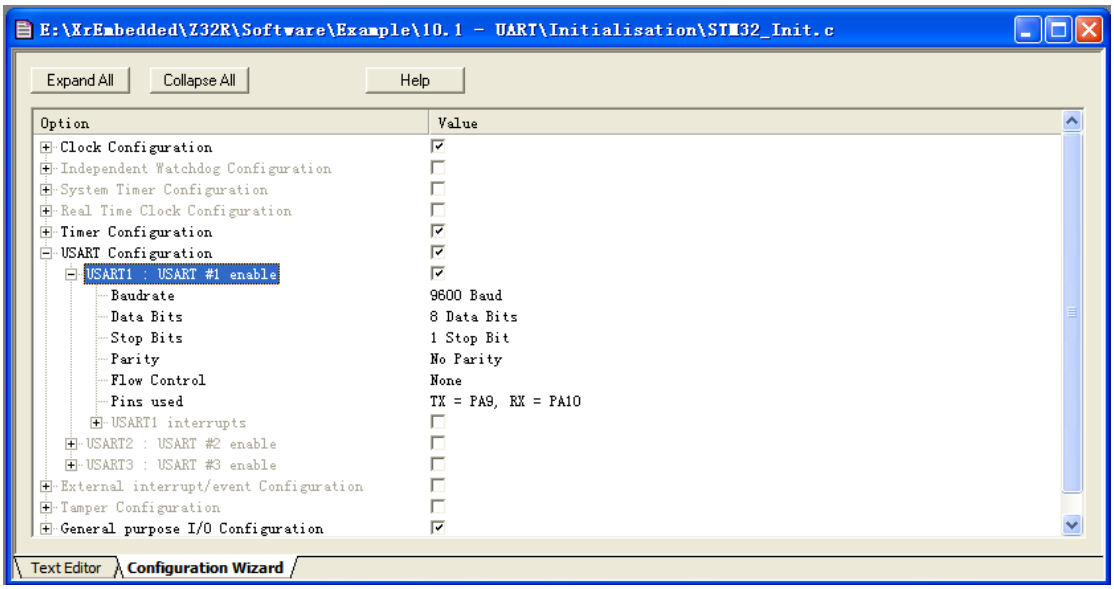
第十四章 串口通信试验

硬件



软件一：UART0 查询方式

用串口线连接 J8(UART0)到 PC 机串口，打开超级终端，设置波特率为 9600，程序运行后我们可以键入字符，板子会反馈我们的字符。首先设置如下：




```
/*-------------------------------------------------------------*\
 | SendChar                                                    |
 | Write character to Serial Port.                               |
\*-------------------------------------------------------------*/
int SendChar (int ch)  {

  while (!(USART1->SR & USART_FLAG_TXE));
```

```
  USART1->DR = (ch & 0x1FF);

  return (ch);
}
/*-------------------------------------------------------*\
 | GetKey                                                |
 | Read character to Serial Port.                          |
\*-------------------------------------------------------*/
int GetKey (void)  {

  while (!(USART1->SR & USART_FLAG_RXNE));

  return ((int)(USART1->DR & 0x1FF));
}
/*-------------------------------------------------------*\
 | MIAN ENTRY                                            |
\*-------------------------------------------------------*/
int main (void) {
  stm32_Init ();                              // STM32 setup
  LCD_Init();
  LCD_Clear_Screen(Blue);

  Font = 0;
  LCD_PutString(30,0,"STM32F 开发板",Cyan,Blue);
  LCD_PutString(15,16,"UART 查询方式通信",Red,Yellow);
  Font = 1;
  LCD_PutString(38,32,"Version 1.0",Green,Blue);
  Font = 0;
  LCD_PutString(10,94,"智林测控技术研究所",Yellow,Blue);
  LCD_PutString(36,111,"www.the0.net",Magenta,Blue);

  printf ("Polling mode Serial I/O Example\r\n\r\n");
  for(;;) {
    unsigned char c;

    printf ("Press a key. ");
    c = getchar ();
    printf ("\r\n");
    printf ("You pressed '%c'.\r\n\r\n", c);
    }
  }
/*-------------------------------------------------------*\
 | END OF FILE                                           |
\*-------------------------------------------------------*/
```

软件二：UART0 中断方式

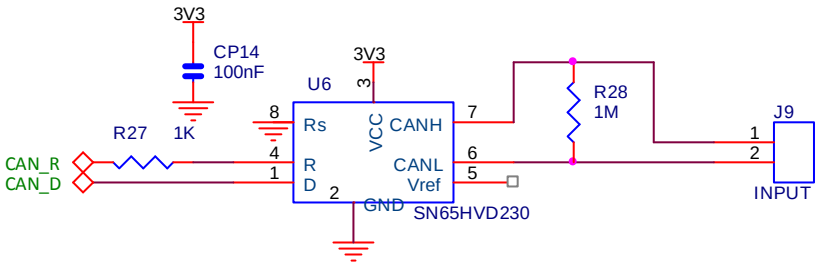
```
/*-------------------------------------------------------*\
 | USART1_IRQHandler                                     |
 | Handles USART1 global interrupt request.              |
\*-------------------------------------------------------*/
void USART1_IRQHandler (void) {
  volatile unsigned int IIR;
  struct buf_st *p;

    IIR = USART1->SR;
    if (IIR & USART_FLAG_RXNE) {                 // read interrupt
      USART1->SR &= ~USART_FLAG_RXNE;           // clear interrupt

      p = &rbuf;

      if (((p->in - p->out) & ~(RBUF_SIZE-1)) == 0) {
        p->buf [p->in & (RBUF_SIZE-1)] = (USART1->DR & 0x1FF);
        p->in++;
      }
    }

    if (IIR & USART_FLAG_TXE) {
      USART1->SR &= ~USART_FLAG_TXE;            // clear interrupt

      p = &tbuf;

      if (p->in != p->out) {
        USART1->DR = (p->buf [p->out & (TBUF_SIZE-1)] & 0x1FF);
        p->out++;
        tx_restart = 0;
      }
      else {
        tx_restart = 1;
         USART1->CR1 &= ~USART_FLAG_TXE;             // disable TX interrupt if nothing
to send

      }
    }
}

/*-------------------------------------------------------*\
 | buffer_Init                                           |
 | initialize the buffers                                |
```

```
\*-------------------------------------------------------*/
void buffer_Init (void) {

  tbuf.in = 0;                                  // Clear com buffer indexes
  tbuf.out = 0;
  tx_restart = 1;

  rbuf.in = 0;
  rbuf.out = 0;
}

/*-------------------------------------------------------*\
 | SendChar                                              |
 | transmit a character                                   |
\*-------------------------------------------------------*/
int SendChar (int c) {
  struct buf_st *p = &tbuf;

                                               // If the buffer is full, return an error
value
  if (SIO_TBUFLEN >= TBUF_SIZE)
    return (-1);

  p->buf [p->in & (TBUF_SIZE - 1)] = c;          // Add data to the transmit buffer.
  p->in++;

  if (tx_restart) {                             // If transmit interrupt is disabled,
enable it
    tx_restart = 0;
     USART1->CR1 |= USART_FLAG_TXE;                 // enable TX interrupt
  }

  return (0);
}

/*-------------------------------------------------------*\
 | GetKey                                                |
 | receive a character                                    |
\*-------------------------------------------------------*/
int GetKey (void) {
  struct buf_st *p = &rbuf;

  if (SIO_RBUFLEN == 0)
    return (-1);
```

```
  return (p->buf [(p->out++) & (RBUF_SIZE - 1)]);
}

/*-------------------------------------------------------*\
 |  Delay                                                |
 |  延时 Inserts a delay time.                            |
 |  nCount: 延时时间                                      |
 |  nCount: specifies the delay time length.              |
\*-------------------------------------------------------*/
void Delay(vu32 nCount) {
  for(; nCount != 0; nCount--);
  }
/*-------------------------------------------------------*\
 | MIAN ENTRY                                            |
\*-------------------------------------------------------*/
int main (void) {
  buffer_Init();                    // init RX / TX buffers
  stm32_Init ();                            // STM32 setup
  LCD_Init();
  LCD_Clear_Screen(Blue);

  Font = 0;
  LCD_PutString(30,0,"STM32F 开发板",Cyan,Blue);
  LCD_PutString(15,16,"UART中断方式通信",Red,Yellow);
  Font = 1;
  LCD_PutString(38,32,"Version 1.0",Green,Blue);
  Font = 0;
  LCD_PutString(10,94,"智林测控技术研究所",Yellow,Blue);
  LCD_PutString(36,111,"www.the0.net",Magenta,Blue);

  printf ("Interrupt driven Serial I/O Example\r\n\r\n");
  for(;;) {
    unsigned char c;

    printf ("Press a key. ");
    c = getchar ();
    printf ("\r\n");
    printf ("You pressed '%c'.\r\n\r\n", c);
    }
  }
/*-------------------------------------------------------*\
 | END OF FILE                                           |
\*-------------------------------------------------------*/
```

第十五章 CAN 总线通信试验

硬件



软件

CAN 的底层操作实现在 CAN.c 中：

```
#include <stm32f10x_lib.h>                      // STM32F10x Library Definitions
#include "STM32_Reg.h"                          // STM32 register and bit Definitions
#include "STM32_Init.h"                         // STM32 Initialization
#include "CAN.h"                                // STM32 CAN adaption layer

CAN_msg       CAN_TxMsg;                        // CAN messge for sending
CAN_msg       CAN_RxMsg;                        // CAN message for receiving

unsigned int  CAN_TxRdy = 0;                    // CAN HW ready to transmit a message
unsigned int  CAN_RxRdy = 0;                    // CAN HW received a message

/*----------------------------------------------------------------------------
  setup CAN interface
 *----------------------------------------------------------------------------*/
void CAN_setup (void)  {
  unsigned int brp = stm32_GetPCLK1();

  RCC->APB1ENR |= RCC_APB1ENR_CANEN;             // enable clock for CAN

                                                 // Note: MCBSTM32 uses PB8 and PB9 for CAN
  RCC->APB2ENR |= RCC_APB2ENR_AFIOEN;            // enable clock for Alternate Function
  AFIO->MAPR   &= 0xFFFF9FFF;                    // reset CAN remap
  AFIO->MAPR   |= 0x00004000;                    //   set CAN remap, use PB8, PB9

  RCC->APB2ENR |= RCC_APB2ENR_IOPBEN;            // enable clock for GPIO B
  GPIOB->CRH &= ~(0x0F<<0);
```

```
  GPIOB->CRH |=  (0x08<<0);                     // CAN RX pin PB.8 input push pull

  GPIOB->CRH &= ~(0x0F<<4);
  GPIOB->CRH |=  (0x0B<<4);                     // CAN TX pin PB.9 alternate output push
pull

  NVIC->Enable[0] |= (1 << (USB_HP_CAN_TX_IRQChannel  & 0x1F));// enable interrupt
  NVIC->Enable[0] |= (1 << (USB_LP_CAN_RX0_IRQChannel & 0x1F));// enable interrupt

  CAN->MCR  = (CAN_MCR_NART  |  CAN_MCR_INRQ);          //  init  mode,  disable  auto.
retransmission
                                              // Note: only FIFO 0, transmit mailbox 0
used
  CAN->IER = (CAN_IER_FMPIE0 | CAN_IER_TMEIE);    // FIFO 0 msg pending, Transmit mbx
empty

  /* Note: this calculations fit for PCLK1 = 36MHz */
  brp  = (brp / 18) / 500000;                   // baudrate is set to 500k bit/s

  /* set BTR register so that sample point is at about 72% bit time from bit start */
  /* TSEG1 = 12, TSEG2 = 5, SJW = 4 => 1 CAN bit = 18 TQ, sample at 72%    */
  CAN->BTR &= ~(((        0x03) << 24) | ((        0x07) << 20) | ((         0x0F) << 16)
| (         0x1FF));
  CAN->BTR |=  ((((4-1) & 0x03) << 24) | (((5-1) & 0x07) << 20) | (((12-1) & 0x0F) <<
16) | ((brp-1) & 0x1FF));
}


/*----------------------------------------------------------------------------
  leave initialisation mode
 *----------------------------------------------------------------------------*/
void CAN_start (void)  {

  CAN->MCR &= ~CAN_MCR_INRQ;                    // normal operating mode, reset INRQ
  while (CAN->MSR & CAN_MCR_INRQ);

}

/*----------------------------------------------------------------------------
  set the testmode
 *----------------------------------------------------------------------------*/
void CAN_testmode (unsigned int testmode) {

  CAN->BTR &= ~(CAN_BTR_SILM | CAN_BTR_LBKM);     // set testmode
```

```
  CAN->BTR |=  (testmode & (CAN_BTR_SILM | CAN_BTR_LBKM));
}

/*----------------------------------------------------------------------------
  check if transmit mailbox is empty
 *----------------------------------------------------------------------------*/
void CAN_waitReady (void)  {

  while ((CAN->TSR & CAN_TSR_TME0) == 0);        // Transmit mailbox 0 is empty
  CAN_TxRdy = 1;

}

/*----------------------------------------------------------------------------
  wite a message to CAN peripheral and transmit it
 *----------------------------------------------------------------------------*/
void CAN_wrMsg (CAN_msg *msg)  {

  CAN->sTxMailBox[0].TIR  = (unsigned int)0;      // Reset TIR register
                                                 // Setup identifier information
  if (msg->format == STANDARD_FORMAT)  {         //    Standard ID
      CAN->sTxMailBox[0].TIR |= (unsigned int)(msg->id << 21) | CAN_ID_STD;
  }  else  {                                      // Extended ID
      CAN->sTxMailBox[0].TIR |= (unsigned int)(msg->id <<  3) | CAN_ID_EXT;
  }
                                                 // Setup type information
  if (msg->type == DATA_FRAME)  {                // DATA FRAME
      CAN->sTxMailBox[0].TIR |= CAN_RTR_DATA;
  }
  else {                                         // REMOTE FRAME
      CAN->sTxMailBox[0].TIR |= CAN_RTR_REMOTE;
  }
                                                 // Setup data bytes
  CAN->sTxMailBox[0].TDLR = (((unsigned int)msg->data[3] << 24) |
                             ((unsigned int)msg->data[2] << 16) |
                             ((unsigned int)msg->data[1] <<  8) |
                             ((unsigned int)msg->data[0])       );
  CAN->sTxMailBox[0].TDHR = (((unsigned int)msg->data[7] << 24) |
                             ((unsigned int)msg->data[6] << 16) |
                             ((unsigned int)msg->data[5] <<  8) |
                             ((unsigned int)msg->data[4])       );
                                                 // Setup length
  CAN->sTxMailBox[0].TDTR &= ~CAN_TDTxR_DLC;
  CAN->sTxMailBox[0].TDTR |=  (msg->len & CAN_TDTxR_DLC);
```

```
  CAN->IER |= CAN_IER_TMEIE;                    // enable  TME interrupt
  CAN->sTxMailBox[0].TIR |=  CAN_TIxR_TXRQ;      // transmit message
}

/*----------------------------------------------------------------------------
  read a message from CAN peripheral and release it
 *----------------------------------------------------------------------------*/
void CAN_rdMsg (CAN_msg *msg)  {
                                               // Read identifier information
  if ((CAN->sFIFOMailBox[0].RIR & CAN_ID_EXT) == 0) { // Standard ID
    msg->format = STANDARD_FORMAT;
    msg->id     = (u32)0x000007FF & (CAN->sFIFOMailBox[0].RIR >> 21);
  }  else  {                                    // Extended ID
    msg->format = EXTENDED_FORMAT;
    msg->id     = (u32)0x0003FFFF & (CAN->sFIFOMailBox[0].RIR >> 3);
  }
                                               // Read type information
  if ((CAN->sFIFOMailBox[0].RIR & CAN_RTR_REMOTE) == 0) {
    msg->type =   DATA_FRAME;                   // DATA   FRAME
  }  else  {
    msg->type = REMOTE_FRAME;                   // REMOTE FRAME
  }
                                               // Read length (number of received bytes)
  msg->len = (unsigned char)0x0000000F & CAN->sFIFOMailBox[0].RDTR;
                                               // Read data bytes
  msg->data[0] = (unsigned int)0x000000FF & (CAN->sFIFOMailBox[0].RDLR);
  msg->data[1] = (unsigned int)0x000000FF & (CAN->sFIFOMailBox[0].RDLR >> 8);
  msg->data[2] = (unsigned int)0x000000FF & (CAN->sFIFOMailBox[0].RDLR >> 16);
  msg->data[3] = (unsigned int)0x000000FF & (CAN->sFIFOMailBox[0].RDLR >> 24);

  msg->data[4] = (unsigned int)0x000000FF & (CAN->sFIFOMailBox[0].RDHR);
  msg->data[5] = (unsigned int)0x000000FF & (CAN->sFIFOMailBox[0].RDHR >> 8);
  msg->data[6] = (unsigned int)0x000000FF & (CAN->sFIFOMailBox[0].RDHR >> 16);
  msg->data[7] = (unsigned int)0x000000FF & (CAN->sFIFOMailBox[0].RDHR >> 24);

  CAN->RF0R |= CAN_RF0R_RFOM0;                  // Release FIFO 0 output mailbox
}


void CAN_wrFilter (unsigned int id, unsigned char format)  {
  static unsigned short CAN_filterIdx = 0;
         unsigned int   CAN_msgId     = 0;
```

```
  if (CAN_filterIdx > 13) {                     // check if Filter Memory is full
    return;
  }
                                               // Setup identifier information
  if (format == STANDARD_FORMAT)  {              // Standard ID
      CAN_msgId  |= (unsigned int)(id << 21) | CAN_ID_STD;
  }  else  {                                     // Extended ID
      CAN_msgId  |= (unsigned int)(id <<  3) | CAN_ID_EXT;
  }

  CAN->FMR  |=  CAN_FMR_FINIT;                   // set Initialisation mode for filter
banks
  CAN->FA0R &=  ~(unsigned int)(1 << CAN_filterIdx); // deactivate filter

                                               // initialize filter
  CAN->FS0R |= (unsigned int)(1 << CAN_filterIdx);// set 32-bit scale configuration
  CAN->FM0R |= (unsigned int)(1 << CAN_filterIdx);// set 2 32-bit identifier list mode

  CAN->sFilterRegister[CAN_filterIdx].FR0 = CAN_msgId; //  32-bit identifier
  CAN->sFilterRegister[CAN_filterIdx].FR1 = CAN_msgId; //  32-bit identifier

  CAN->FFA0R &= ~(unsigned int)(1 << CAN_filterIdx);  // assign filter to FIFO 0
  CAN->FA0R  |=  (unsigned int)(1 << CAN_filterIdx);  // activate filter

  CAN->FMR &= ~CAN_FMR_FINIT;                   // reset Initialisation mode for filter
banks

  CAN_filterIdx += 1;                           // increase filter index
}

/*----------------------------------------------------------------------------
  CAN transmit interrupt handler
 *----------------------------------------------------------------------------*/
void USB_HP_CAN_TX_IRQHandler (void) {

  if (CAN->TSR & CAN_TSR_RQCP0) {               // request completed mbx 0
    CAN->TSR |= CAN_TSR_RQCP0;                  // reset request complete mbx 0
    CAN->IER &= ~CAN_IER_TMEIE;                 // disable  TME interrupt

    CAN_TxRdy = 1;
  }
}

/*----------------------------------------------------------------------------
```

```
  CAN receive interrupt handler
 *--------------------------------------------------------------------------*/
void USB_LP_CAN_RX0_IRQHandler (void) {

  if (CAN->RF0R & CAN_RF0R_FMP0) {              // message pending ?
     CAN_rdMsg (&CAN_RxMsg);                    // read the message

    CAN_RxRdy = 1;                             // set receive flag
  }
}
```

在主程序中，我们采集 ADC 转换的结果发送出去，同时也接受发来的信息。

```
  for(;;) {
     i = analog[0] * 150 / 0xFFF;
      //sprintf(s, "%d", i);
      //LCD_PutString(5,80,s,Red,Yellow);
     LCD_Rectangle( 5,80,5+i,86, Magenta );
     LCD_Rectangle( 6+i,80,155,86, Green );

    if (CAN_TxRdy) {
      CAN_TxRdy = 0;

      CAN_TxMsg.data[0] = analog[0]>>4;          // data[0] field = ADC value
      CAN_wrMsg (&CAN_TxMsg);                   // transmit message
      val_Tx = CAN_TxMsg.data[0];
      }

    Delay(10000);                               // Wait a while to receive the message

    if (CAN_RxRdy) {
      CAN_RxRdy = 0;

      val_Rx = CAN_RxMsg.data[0];
      }

    sprintf(s, "Tx:%2X", val_Tx );
    LCD_PutString(4+52,42,s,Red,Yellow);

    sprintf(s, "Rx:%2X", val_Rx );
    LCD_PutString(4+52,60,s,Red,Yellow);

    Delay(10000);                               // Wait a while to receive the message
     }
```

这里我们用自检方式来做实验，我们在液晶上看到的数据是自己发的，当我们把两块相同的 STM32 板子的 CAN 总线连接到一起时需要修改 CAN 总线控制器初始化 void can_Init (void)函

数：

```
/*----------------------------------------------------------*\
 |initialize CAN interface                                  |
\*----------------------------------------------------------*/
void can_Init (void) {

  CAN_setup ();                                 // setup CAN interface
  CAN_wrFilter (33, STANDARD_FORMAT);            // Enable reception of messages

  /* COMMENT THE LINE BELOW TO ENABLE DEVICE TO PARTICIPATE IN CAN NETWORK   */
  CAN_testmode(CAN_BTR_SILM | CAN_BTR_LBKM);      // Loopback, Silent Mode (self-test)

  CAN_start ();                                 // leave init mode

  CAN_waitReady ();                              // wait til mbx is empty
}
```

其中的

```
  CAN_testmode(CAN_BTR_SILM | CAN_BTR_LBKM);      // Loopback, Silent Mode (self-test)
```

要注释掉，就可以实现硬件的双机通讯了。

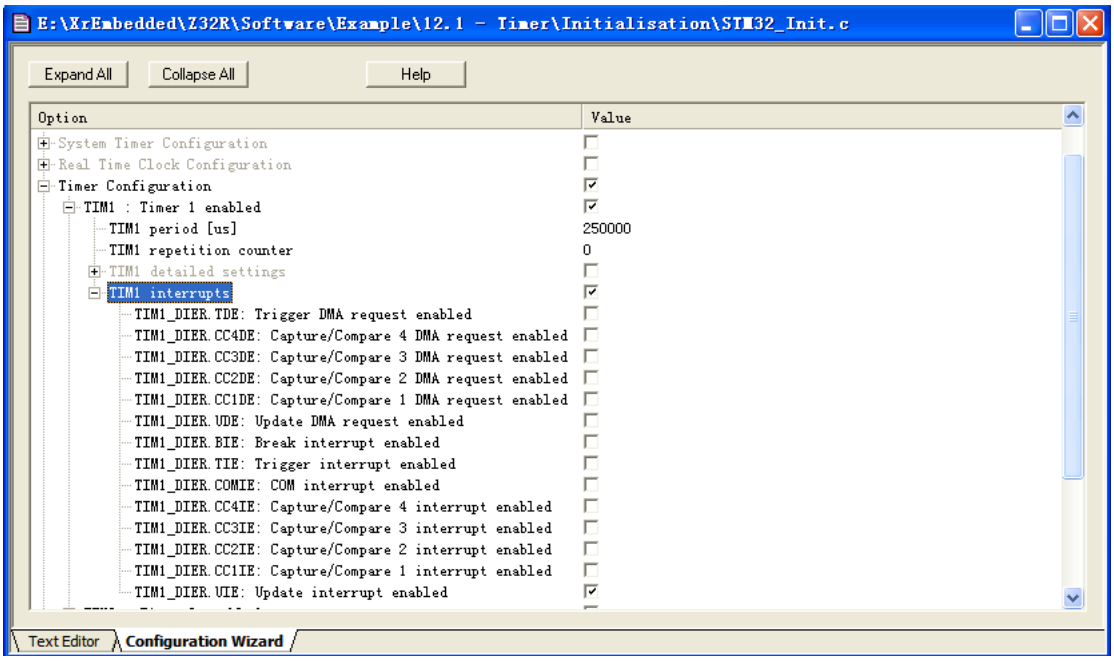
第十六章 定时器试验：使用中断方式

这章通过定时器 1 的例子来学习使用中断，程序很短，但说明问题。以下是程序的完整代码：

```
/*-------------------------------------------------------------*\
 |  引入相关芯片的头文件                                       |
\*-------------------------------------------------------------*/
#include <stdio.h>
#include <stm32f10x_lib.h>    // STM32F10x Library Definitions
#include "STM32_Init.h"       // STM32 Initialization
#include "TFT018.h"

int ledLight = 0;
/*-------------------------------------------------------------*\
 | HARDWARE DEFINE                                             |
\*-------------------------------------------------------------*/
#define LED            ( 1 << 5 )              // PB5: LED D2
/*-------------------------------------------------------------*\
 | Timer1 Update Interrupt Handler                             |
\*-------------------------------------------------------------*/
void TIM1_UP_IRQHandler (void) {

  if ((TIM1->SR & 0x0001) != 0) {               // check interrupt source

    ledLight = ~ledLight;
    if( ledLight )
        GPIOB->ODR &= ~LED;                         // switch on LED
    else
        GPIOB->ODR |=  LED;                         // switch off LED

    TIM1->SR &= ~(1<<0);                        // clear UIF flag
 }
} // end TIM1_UP_IRQHandler
/*-------------------------------------------------------------*\
 | MIAN ENTRY                                                  |
\*-------------------------------------------------------------*/
int main (void) {
  stm32_Init ();                                // STM32 setup
  LCD_Init();
  LCD_Clear_Screen(Blue);

  Font = 0;
```

```
  LCD_PutString(30,0,"STM32F 开发板",Cyan,Blue);
  LCD_PutString(25,16,"定时器中断实验",Red,Yellow);
  Font = 1;
  LCD_PutString(38,32,"Version 1.0",Green,Blue);
  Font = 0;
  LCD_PutString(10,94,"智林测控技术研究所",Yellow,Blue);
  LCD_PutString(36,111,"www.the0.net",Magenta,Blue);

  for(;;) {
    }
  }
/*-------------------------------------------------------------*\
 | END OF FILE                                                 |
\*-------------------------------------------------------------*/
```

当然要配置一下：

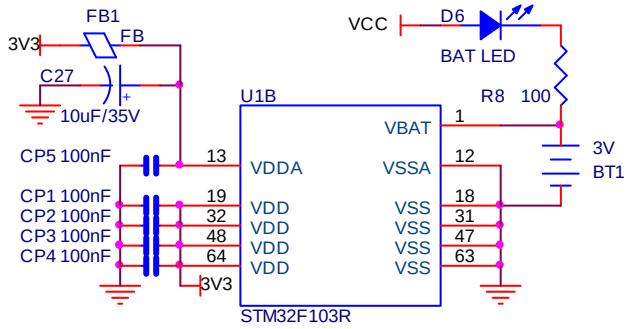


程序运行后，我们看到发光管闪烁。

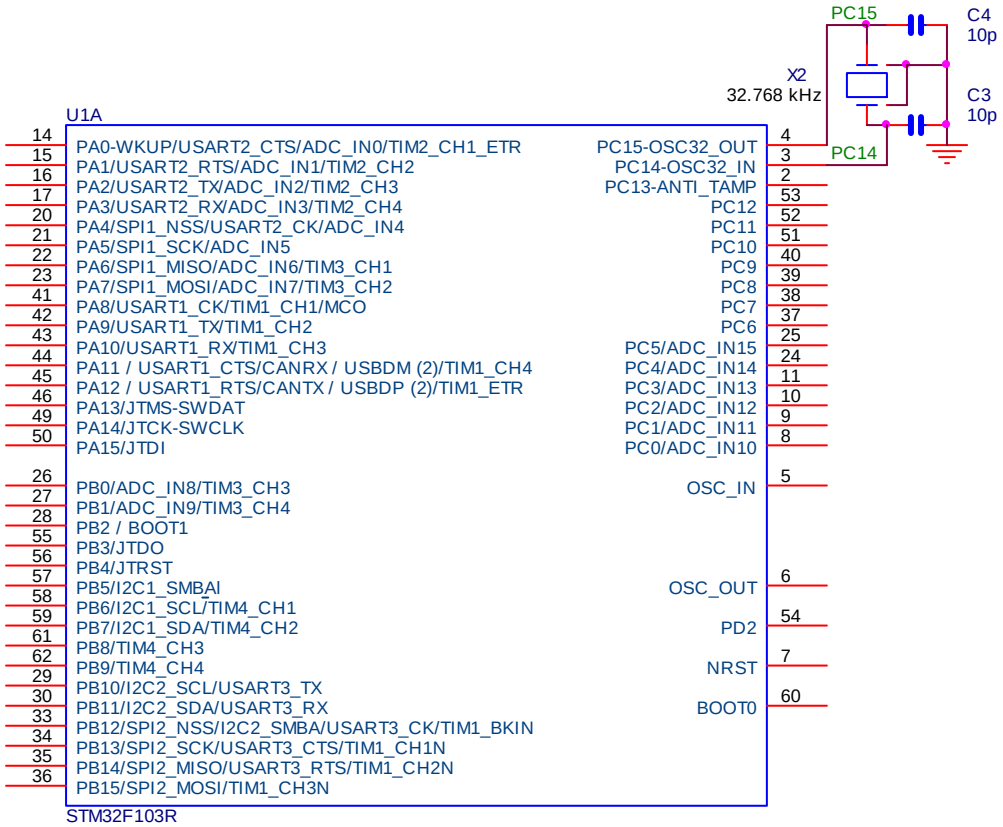
第十七章 RTC 试验

硬件

STM32 支持电池供电的实时时钟（RTC）。



还需要一个 32.768KHz 的晶振:



软件

```
/*------------------------------------------------------------*\
 | HARDWARE DEFINE                                            |
```

```
\*-------------------------------------------------------*/
#define LED             ( 1 << 5 )                  // PB5: LED D2
/*-------------------------------------------------------*\
 | RTC Interrupt Handler                                 |
\*-------------------------------------------------------*/
void RTC_IRQHandler(void) {
  if (RTC->CRL & (1<<0) ) {                         // check second flag
    RTC->CRL &= ~(1<<0);                            // clear second flag

    ledLight = ~ledLight;
    if( ledLight )
        GPIOB->ODR &= ~LED;                             // switch on LED
    else
        GPIOB->ODR |=  LED;                             // switch off LED
    }
  if (RTC->CRL & (1<<1) ) {                         // check alarm flag
    RTC->CRL &= ~(1<<1);                            // clear alarm flag
    Alarm = 1;
    }
  } // end TIM1_UP_IRQHandler
/*-------------------------------------------------------*\
 | RTC Interrupt Handler                                 |
\*-------------------------------------------------------*/
void DisplayTime(void) {
  unsigned int TimeVar=0,  THH = 0, TMM = 0, TSS = 0;
  char s[30];

  TimeVar = RTC->CNTH << 16 | RTC->CNTL;
  /* Compute  hours */
  THH = TimeVar/3600;
  /* Compute minutes */
  TMM = (TimeVar % 3600)/60;
  /* Compute seconds */
  TSS = (TimeVar % 3600)% 60;

  sprintf(s,"%2d 时%2d 分%2d 秒",THH, TMM, TSS);
  LCD_PutString(30,50,s,Red,Yellow);
  }
/*-------------------------------------------------------*\
 | MIAN ENTRY                                            |
\*-------------------------------------------------------*/
int main (void) {
  stm32_Init ();                                // STM32 setup
  LCD_Init();
```

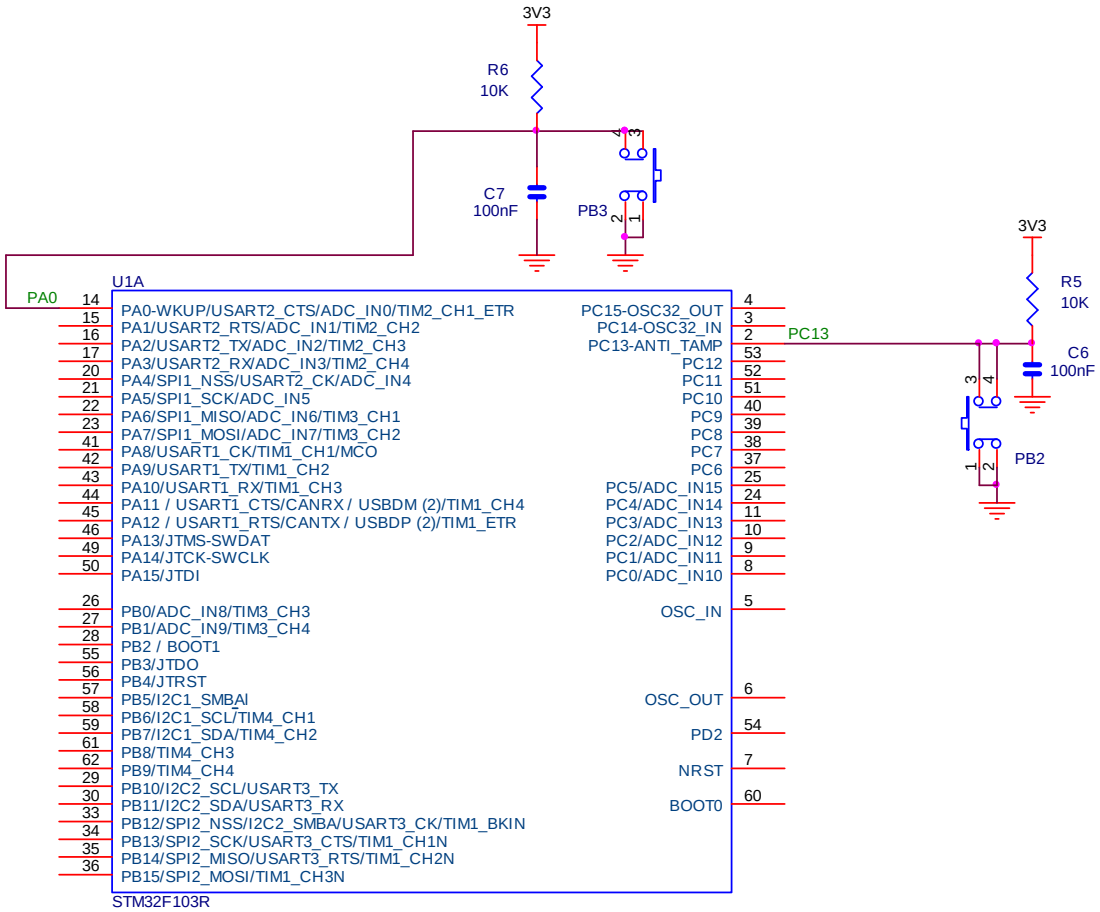
```
  LCD_Clear_Screen(Blue);

  Font = 0;
  LCD_PutString(30,0,"STM32F 开发板",Cyan,Blue);
  LCD_PutString(40,16,"RTC 中断实验",Red,Yellow);
  Font = 1;
  LCD_PutString(38,32,"Version 1.0",Green,Blue);
  Font = 0;
  LCD_PutString(10,94,"智林测控技术研究所",Yellow,Blue);
  LCD_PutString(36,111,"www.the0.net",Magenta,Blue);

  for(;;) {
    DisplayTime();
    if(Alarm == 1)
        LCD_PutString(60,70,"报警",Yellow,Red);
    }
  }
/*-------------------------------------------------------------*\
 | END OF FILE                                                 |
\*-------------------------------------------------------------*/
```

这段程序在液晶屏幕上显示当前时间。同时中断程序使发光管闪烁。

首先，初始化 RTC，接着设置一个时间，然后读取时钟数据并且显示。

第十八章 外部中断试验

硬件



我们板上有两个按钮 PB2 和 PB3，这个实验我们用这两个按钮产生外部中断。

软件

```
/*-------------------------------------------------------------*\
 |   引入相关芯片的头文件                                      |
\*-------------------------------------------------------------*/
#include <stdio.h>
#include <stm32f10x_lib.h>    // STM32F10x Library Definitions
#include "STM32_Init.h"       // STM32 Initialization
#include "TFT018.h"

int ledLight = 0;
int Alarm = 0;
```

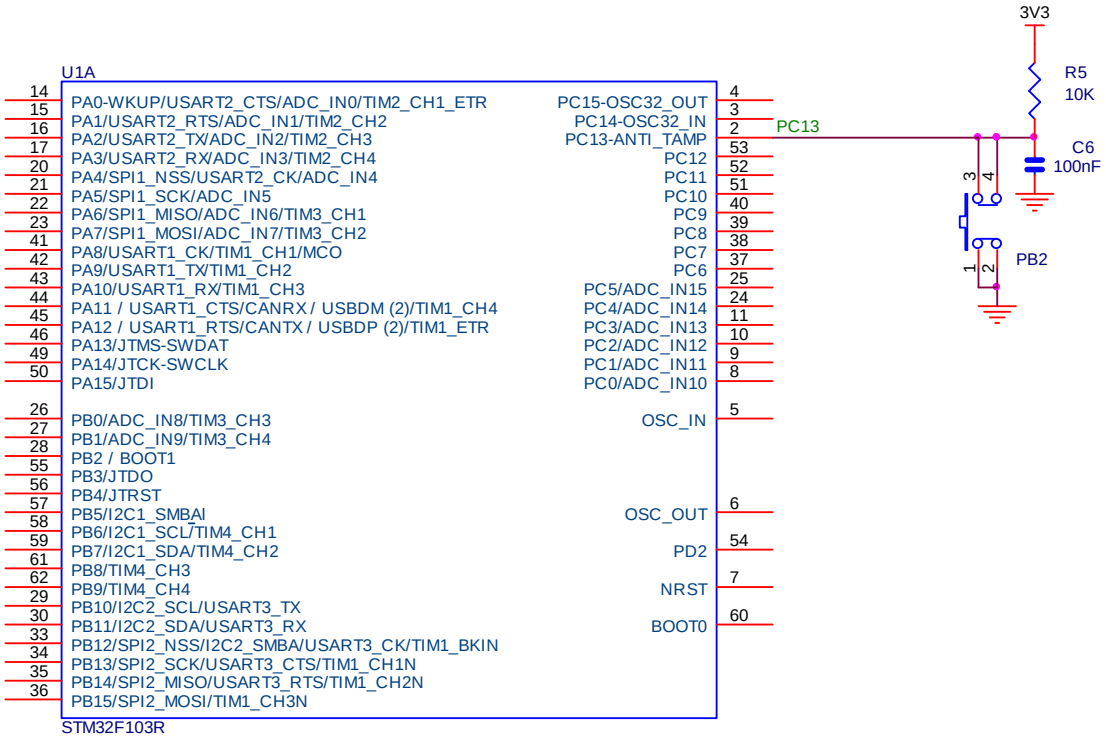
```
/*-------------------------------------------------------*\
 | HARDWARE DEFINE                                       |
\*-------------------------------------------------------*/
#define LED             ( 1 << 5 )                  // PB5: LED D2
/*-------------------------------------------------------*\
 | EXTI0 Interrupt Handler                               |
\*-------------------------------------------------------*/
void EXTI0_IRQHandler(void)
{
  if (EXTI->PR & (1<<0)) {                          // EXTI13 interrupt pending?
    if ((ledLight ^=1) == 0)
        GPIOB->ODR &= ~LED;                             // switch on LED
    else
        GPIOB->ODR |=  LED;                             // switch off LED

    EXTI->PR |= (1<<0);                             // clear pending interrupt
  }
}

/*-------------------------------------------------------*\
 | EXTI15..10 Interrupt Handler                          |
\*-------------------------------------------------------*/
void EXTI15_10_IRQHandler(void)
{
  if (EXTI->PR & (1<<13)) {                         // EXTI0 interrupt pending?
    if ((ledLight ^=1) == 0)
        GPIOB->ODR &= ~LED;                             // switch on LED
    else
        GPIOB->ODR |=  LED;                             // switch off LED

    EXTI->PR |= (1<<13);                            // clear pending interrupt
  }
}
/*-------------------------------------------------------*\
 | MIAN ENTRY                                            |
\*-------------------------------------------------------*/
int main (void) {
  stm32_Init ();                                  // STM32 setup
  LCD_Init();
  LCD_Clear_Screen(Blue);

  Font = 0;
  LCD_PutString(30,0,"STM32F 开发板",Cyan,Blue);
  LCD_PutString(35,16,"外部中断实验",Red,Yellow);
```

```
  Font = 1;
  LCD_PutString(38,32,"Version 1.0",Green,Blue);
  Font = 0;
  LCD_PutString(10,94,"智林测控技术研究所",Yellow,Blue);
  LCD_PutString(36,111,"www.the0.net",Magenta,Blue);

  for(;;) {
    }
  }
/*-------------------------------------------------------------*\
 | END OF FILE                                                 |
\*-------------------------------------------------------------*/
```

软件运行后，按按钮 PB2 和 PB3，都会导致中断发生，反转发光管。

第十九章 入侵检测试验

硬件



按钮 PB2 接在侵入检测管脚上。

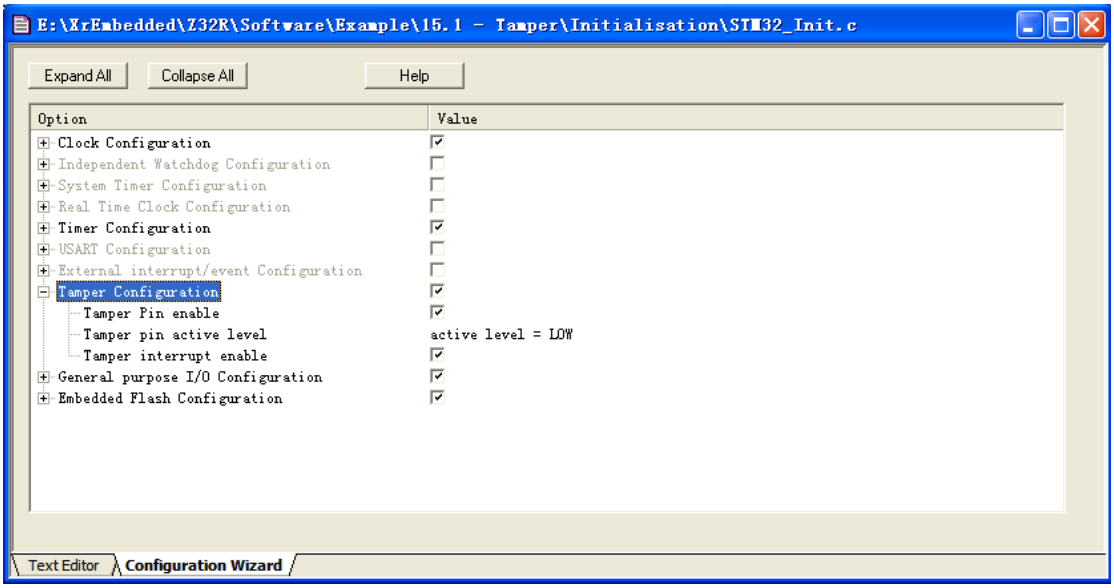
软件

```
/*-------------------------------------------------------------*\
 |  引入相关芯片的头文件                                       |
\*-------------------------------------------------------------*/
#include <stdio.h>
#include <stm32f10x_lib.h>    // STM32F10x Library Definitions
#include "STM32_Init.h"       // STM32 Initialization
#include "TFT018.h"

int ledLight = 0;
/*-------------------------------------------------------------*\
 | HARDWARE DEFINE                                             |
\*-------------------------------------------------------------*/
#define LED            ( 1 << 5 )              // PB5: LED D2
/*-------------------------------------------------------------*\
```

```
 | TAMPER Interrupt Handler                                    |
\*-------------------------------------------------------------*/
void TAMPER_IRQHandler(void)
{
  if (BKP->CSR & (1<<9) ) {                    // Tamper interrupt flag
    PWR->CR  |=  (1<<8);                      // enable access to RTC,
BDC registers
    BKP->CSR |=  (1<<1);                      // clear Tamper Interrupt
    BKP->CSR |=  (1<<0);                      // clear tamper Event
    PWR->CR  &= ~(1<<8);                      // disable access to RTC,
BDC registers
    if ((BKP->DR1 == 0) &&
        (BKP->DR2 == 0)   )
      GPIOB->ODR |=  LED;                       // switch off LED
    else
      GPIOB->ODR &= ~LED;                       // switch on LED
    }
}

/*-------------------------------------------------------------*\
 | MIAN ENTRY                                                  |
\*-------------------------------------------------------------*/
int main (void) {
  stm32_Init ();                              // STM32 setup

  PWR->CR  |=  (1<<8);                        // enable access to RTC,
BDC registers
  BKP->DR1 = 0x55AA;                          // fill BKP_DR1 register
  BKP->DR2 = 0x33CC;                          // fill BKP_DR2 register
  PWR->CR  &= ~(1<<8);                        // disable access to RTC,
BDC registers

  LCD_Init();
  LCD_Clear_Screen(Blue);

  Font = 0;
  LCD_PutString(30,0,"STM32F 开发板",Cyan,Blue);
  LCD_PutString(35,16,"侵入检测实验",Red,Yellow);
  Font = 1;
  LCD_PutString(38,32,"Version 1.0",Green,Blue);
  Font = 0;
  LCD_PutString(10,94,"智林测控技术研究所",Yellow,Blue);
  LCD_PutString(36,111,"www.the0.net",Magenta,Blue);
```

```
  for(;;) {
    }
  }
/*-------------------------------------------------------------*\
 | END OF FILE                                                 |
\*-------------------------------------------------------------*/
```

当然，需要配置：



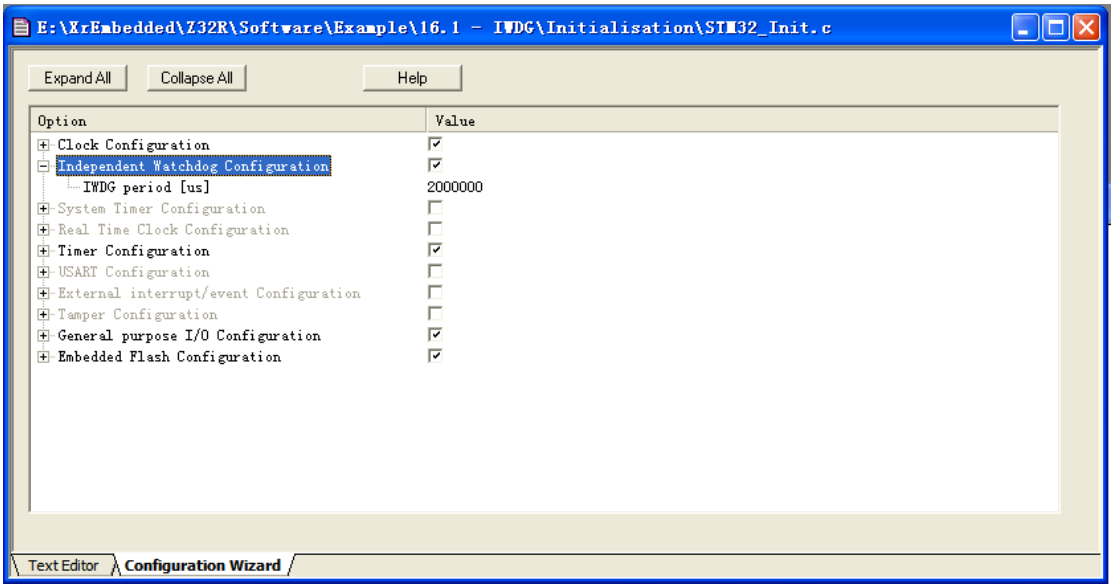
第二十章 看门狗试验

软件

```
/*-------------------------------------------------------------*\
 |  引入相关芯片的头文件                                       |
\*-------------------------------------------------------------*/
#include <stdio.h>
#include <stm32f10x_lib.h>    // STM32F10x Library Definitions
#include "STM32_Init.h"       // STM32 Initialization
#include "TFT018.h"

int ledLight = 0;
/*-------------------------------------------------------------*\
 | HARDWARE DEFINE                                             |
\*-------------------------------------------------------------*/
#define LED            ( 1 << 5 )              // PB5: LED D2
/*-------------------------------------------------------------*\
 |  Delay                                                      |
 |  延时 Inserts a delay time.                                  |
 |  nCount: 延时时间                                           |
 |  nCount: specifies the delay time length.                     |
\*-------------------------------------------------------------*/
void Delay(unsigned int nCount) {
  for(; nCount != 0; nCount--);
  }
/*-------------------------------------------------------------*\
 | MIAN ENTRY                                                  |
\*-------------------------------------------------------------*/
int main (void) {
  int i;

  stm32_Init ();                                // STM32 setup

  LCD_Init();
  LCD_Clear_Screen(Blue);

  Font = 0;
  LCD_PutString(30,0,"STM32F 开发板",Cyan,Blue);
  LCD_PutString(25,16,"看门狗复位实验",Red,Yellow);
  Font = 1;
```

```
  LCD_PutString(38,32,"Version 1.0",Green,Blue);
  Font = 0;
  LCD_PutString(10,94,"智林测控技术研究所",Yellow,Blue);
  LCD_PutString(36,111,"www.the0.net",Magenta,Blue);

  if (RCC->CSR & (1<<29)) {                    // IWDG Reset Flag set
    RCC->CSR |= (1<<24);                       // Clear Reset Flags
    GPIOB->ODR |=  LED;                        // switch on LED
    LCD_PutString(45,50,"看门狗复位",Yellow,Red);
    }
  else {
    GPIOB->ODR &= ~LED;                        // switch off LED
    LCD_PutString(45,50," 开机复位 ",Blue,Yellow);
    }

  for (i = 0; i < 10; i++) {
    Delay(1000000);                           // wait less than watchdog
interval
    IWDG->KR  = 0xAAAA;                       // reload the watchdog
  }
  GPIOB->ODR |=  LED;                        // switch on LED

  for(;;) {
    }
  }
/*-------------------------------------------------------------*\
 | END OF FILE                                                 |
\*-------------------------------------------------------------*/
```

配置：



程序运行后，会看到硬件复位信息，当程序停止喂狗后，会显示出看门狗复位信息。

第二十一章 软件中断试验

软件

先看看软件中断处理函数：

```
__asm void SVC_Handler (void) {
       PRESERVE8

               TST     LR,#4                   ; Called from Handler Mode?
               MRSNE   R12,PSP                 ; Yes, use PSP
               MOVEQ   R12,SP                  ; No, use MSP
               LDR     R12,[R12,#24]           ; Read Saved PC from Stack
               LDRH    R12,[R12,#-2]           ; Load Halfword
               BICS    R12,R12,#0xFF00         ; Extract SVC Number

               PUSH    {R4,LR}                 ; Save Registers
               LDR     LR,=SVC_Count
               LDR     LR,[LR]
               CMP     R12,LR
               BHS     SVC_Dead                ; Overflow
               LDR     LR,=SVC_Table
               LDR     R12,[LR,R12,LSL #2]     ; Load SVC Function Address
               BLX     R12                     ; Call SVC Function

               POP     {R4,LR}
               TST     LR,#4
               MRSNE   R12,PSP
               MOVEQ   R12,SP
               STM     R12,{R0-R3}             ; Function return values
               BX      LR                      ; RETI

SVC_Dead
               B       SVC_Dead                ; None Existing SVC

SVC_Cnt         EQU    (SVC_End-SVC_Table)/4
SVC_Count       DCD     SVC_Cnt

; Import user SVC functions here.
               IMPORT  __SVC_0
               IMPORT  __SVC_1
               IMPORT  __SVC_2
               IMPORT  __SVC_3
```

```
SVC_Table
; Insert user SVC functions here
                DCD     __SVC_0                 ; SVC 0 Function Entry
                DCD     __SVC_1                 ; SVC 1 Function Entry
                DCD     __SVC_2                 ; SVC 2 Function Entry
                DCD     __SVC_3                 ; SVC 2 Function Entry

SVC_End

                ALIGN
}

/*-----------------------------------------------------------------------------
 * end of file
 *-----------------------------------------------------------------------------*/
```

软件中断后，按照软件中断号分别执行__SVC_0、__SVC_1、__SVC_2、__SVC_3。这四个函数定义如下：

```
int __svc(0) add (int i1, int i2);
int __SVC_0         (int i1, int i2) {
   return (i1 + i2);
}

int __svc(1) mul4(int i);
int __SVC_1         (int i) {
   return (i << 2);
}

int __svc(2) div (int i1, int i2);
int __SVC_2         (int i1, int i2) {
   return (i1 / i2);
}

int __svc(3) mod (int i1, int i2);
int __SVC_3         (int i1, int i2) {
   return (i1 % i2);
}
```

在主函数里测试：

```
int res;
char s[30];
/*-----------------------------------------------------------------------------
-------
  Test Function
```

```
*-------------------------------------------------------------------
------*/
void test_t (void) {
  res  = div (res, 10);                          // Call SWI Functions
  sprintf(s,"div:%d",res);
  LCD_PutString(10,80,s,Magenta,Blue);

  res  = mod (res,  3);
  sprintf(s,"mod:%d",res);
  LCD_PutString(90,80,s,Magenta,Blue);
}


/*-------------------------------------------------------------------
-------
  Test Function

*-------------------------------------------------------------------
------*/
void test_a (void) {
  res  = add (74,  27);                          // Call SWI Functions
  sprintf(s,"add:%d",res);
  LCD_PutString(10,60,s,Magenta,Blue);

  res += mul4(res);
  sprintf(s,"mul4:%d",res);
  LCD_PutString(90,60,s,Magenta,Blue);
}


/*-----------------------------------------------------------*\
 | MIAN ENTRY                                                |
\*-----------------------------------------------------------*/
int main (void) {
  stm32_Init ();                                 // STM32 setup
  LCD_Init();
  LCD_Clear_Screen(Blue);

  Font = 0;
  LCD_PutString(30,0,"STM32F 开发板",Cyan,Blue);
  LCD_PutString(30,16,"软件中断实验",Red,Yellow);
  Font = 1;
  LCD_PutString(38,32,"Version 1.0",Green,Blue);
  Font = 0;
```

```
  LCD_PutString(10,94,"智林测控技术研究所",Yellow,Blue);
  LCD_PutString(36,111,"www.the0.net",Magenta,Blue);

  sprintf(s,"res:%d",res);
  LCD_PutString(20,40,s,Magenta,Blue);

  test_a();
  test_t();

  for(;;) {
    }
  }
/*-------------------------------------------------------------*\
 | END OF FILE                                                 |
\*-------------------------------------------------------------*/
```

第二十二章 实时操作系统试验：uC/OS-II

首先，版权问题。

uC/OS-II 作者致中国用户的公开信

致 Micrium 产品的使用者:
Micrium 公司产品包括 µC/OS-II, µC/GUI，µC/FS，µC/TCP-IP，µC/ USB 等。 Micrium 公司提供嵌入式系统应用方面的产品，并对其软件拥有知识产权。 我公司花费了大量的时间和财力为嵌入式领域提供高质量的软件产品。我们的所有产品都以源代码的形式提供给客户，具有极大的适用性。我们的产品不是免费软件，也不是开放源码的软件，因此，不能免费使用。 您可以通过购买我关于 µC/OS-II 的书而得到 µC/OS-II 源代码，当您从芯片厂商那里购买评估板时,您也可以获得 µC/OS-II 源代码,但您必须同时购买一本含有 CD 的我的书。当您用于商业目的，您必须购买使用授权，这在书中有明确规定。 我公司其它软件如 µC/GUI，µC/FS，µC/TCP-IP，µC/USB 等的销售模式与 µC/OS-II 不同，如果您没有购买使用授权，您完全不可以拥有该源代码，也不能将源代码用于产品。 如果我们的努力工作不能得到回报，我们将不能继续设计并生产这些优秀的软件产品。因此，请不要非法使用和发布我们的软件。如果您已经有了 µC/OS-II 的代码,但您并没有我的关于 µC/OS-II 的书，请您购买一本。 如果您正在将 µC/OS-II 用于您的产品，您需要购买并获得正式使用授权。如果您有一份我公司其它产品的拷贝，您必须销毁非法拷贝并购买使用授权。 Micrium 公司系列产品和使用授权由北京麦克泰软件技术有限公司在中国独家代理。 感谢您的诚实合作和理解!

只是一个提醒，如果大家在量产产品中使用，被抓到罚款，可就不划算了。

我们对操作系统使用的看法

我的意见是，很多场合 RTOS 是可以不用的，有时候用了反倒麻烦。像 NXP 系列的 ARM 我个人称其为“小 ARM”,就是片上带有 Flash 的这种，我们可以把这样 ARM 当作一个功能强大的单片机来用。当然，我不是反对使用 RTOS，要看具体项目来定，需要用就用。

软件

这是个简单的例子，建立 2 个任务，运行后一个任务使 LED 闪烁，另一个在液晶上显示任务切换数。源码在光盘上。