PT32x031开发板使用说明

概述

PT32x031开发板是澎湃微电子开发一款基于澎湃CortexM0（PT32F031(LQFP32)）芯片的开发板，开发板支持串口TTL转USB，4段数码管显示，模拟电压采集，蜂鸣器以及RGB三色灯显示。PT32x031是一颗兼容2.2V至5.5V的MCU主控芯片，具有很高的集成度和较高的性能，芯片使用M0内核，有比较齐全的数字、模拟外设资源

MCU特征

- 工作电压：2.2V~5.5V,支持上电复位电压5V和1.8V
- 深度睡眠时功耗 8uA，在睡眠模式下，提供多种唤醒源（IO唤醒，定时唤醒等）
- 最高CPU运行时钟频率48MHz
- 指令存储器：2K-Byte/4K-Byte/8K-Byte/16K-Byte/32K-Byte FLASH（不同型号规格）
- 数据存储器：512-Byte/1K-Byte/2K-Byte/4K-Byte SRAM（不同型号规格）
- 2路标准IIC串口,支持主/从模式,支持标准速率（100Kb/s），快速速率（400Kb/s）
- 2路标准SPI接口,支持主/从模式
- 时钟源：内部高频24MHz RC 振荡器，内部低频32KHz RC 振荡器和外部晶体震荡
- 1个24位系统定时器（M0内核自带）
- 2个16位通用定时器，timer0 & timer1，每个定时器支持4路输入捕捉，4路比较输出，输入捕捉支持上升沿捕捉，下降沿捕捉，上升沿至下降沿捕捉，下降沿至上升沿捕捉，比较输出支持死区时间可调，支持互补PWM输出，支持输入中断，输出中断和溢出中断以及刹车输入可用于触发ADC转换
- 数码管驱动外设，最多支持4COM
- 2路模拟比较器,比较结果可触发中断
- 单独外设模块可输出不同频率方波驱动蜂鸣器
- 2个8位通用定时器，timer2 & timer3
- 1个16位唤醒系统定时器，支持16位递增计数，使用内部低速振荡32KHz时钟作为计数时钟，可唤醒系统
- 12位高精度ADC，变化速率为500Ksps/1Msps，支持12路通道输入，可转换内部bandgap 2V电压,支持单次转换（single mode）/连续转换（continuous scan mode）,支持外部I/O触发一次转换（上升沿，下降沿，任意电平切换）,支持内部定时器timer0/timer1定时触发一次转换
- 片上看门狗，32 位递减计数，使用系统时钟作为计数时钟
- 独立看门狗，32 位递减计数，使用内部低速振荡 32kHz 时钟作为计数时钟
- 低电压监控，当电压低于安全值时，输出中断或复位，触发阈值支持：4V，3.5V，3V，2.75V，2.5V，2.2V，2.0V，1.8V
- 提供至多 28 个 GPIO 口，每个 GPIO 口均可提供外部中断并用于唤醒系统，支持 4 个 GPIO 承受较大灌电流，电流强度可达 110mA，支持弱上拉（下拉）功能，上拉（下拉）电阻为 50kΩ，支持输出强驱动，普通驱动电流为 5mA，强输出驱动电流为 20mA，支持开漏功能，sink current 为 5mA，支持模拟模式（作为 ADC 输入或 OPA 的输入输出）
- 调试接口，使用 SWD 标准两线制调试接口

应用领域

- 物联网
- 家用电子
- 医疗电子
- 电机控制

封装信息

- LQFP32(7*7)
- QFN32 (4*4)
- QFN28 (4*4)
- TSSOP20
- TSSOP28

开发板电路参考

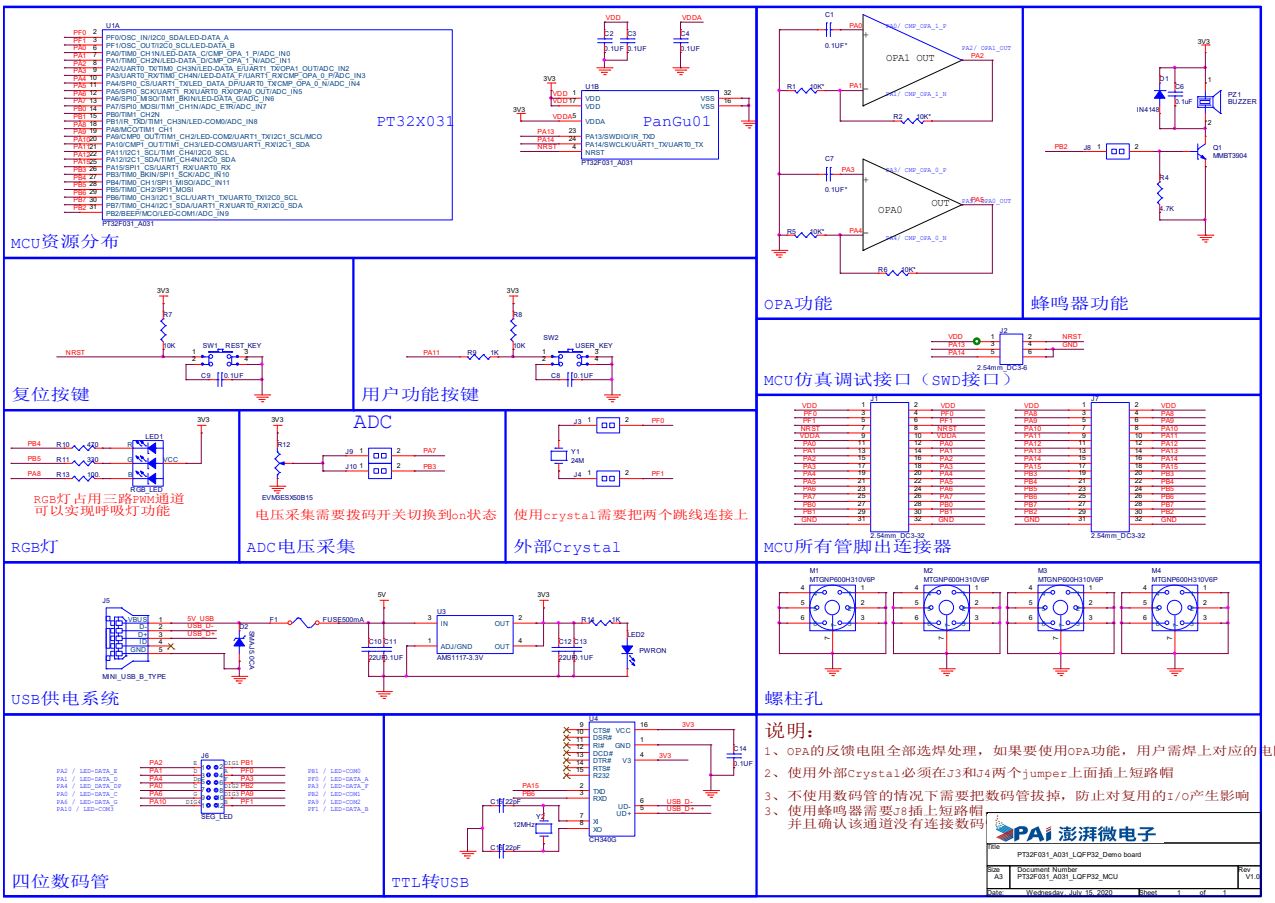

图 1 开发板电路图

开发板电路资源分配

- 开发板上直接可用的外设资源有四位数码管、按键输入、RGB三色彩灯、ADC模拟量输入、蜂鸣器、TTL转USB串口芯片、24MHz外部晶振，具体引脚资源分配详见图1的资源分配表格；
- 一些外设资源存在共用MCU引脚的情况，存在共用的引脚均有通过jumpper进行隔离，比如驱动蜂鸣器的引脚以及数码管驱动引脚共用，具体jumpper短路帽的接法请参考Demo板原理图；
- 预留了复位按键，程序Download完成后可直接通过复位按键启动；
- 直接通过MinUsb口5V供电，同时提供一路串口方便调测，使用该串口前需要安装CH340驱动；
- MCU的所有管脚均连接J1和J7两个双排插针，相邻一排的排针对应一个管脚，管脚号见丝印信息，方便用户可以直接通过杜邦线搭建他们自己的应用环境；
- 开发板提供一个PWRON指示灯，可确认开发板是否正常供电；
- 预留了用于OPA功能的反馈电阻以及滤波电容，这些阻容默认不焊，用户根据应用需求

选择合适的阻值以及容值焊接；

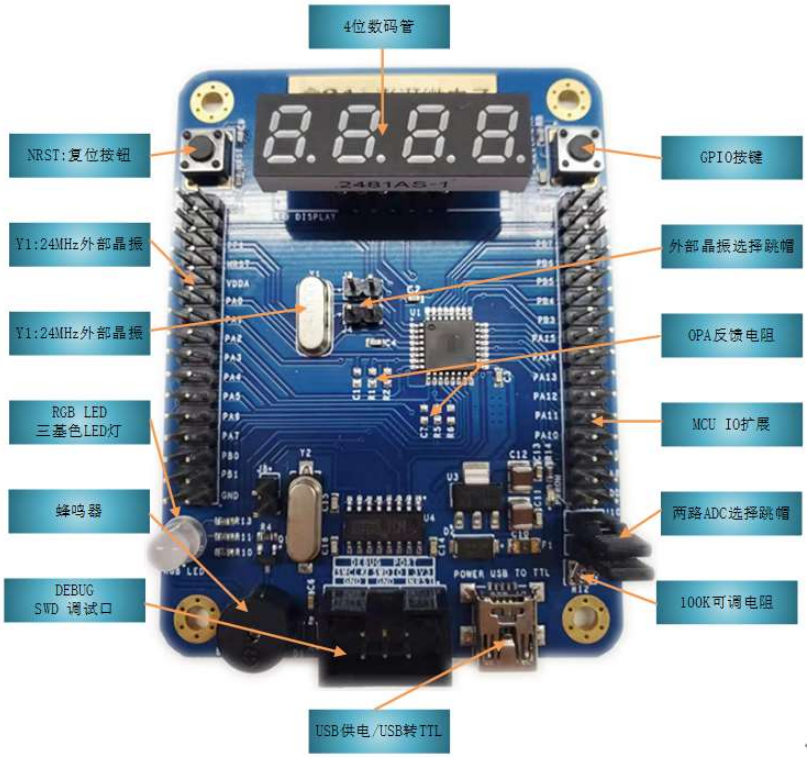

图 2 开发板资源分配

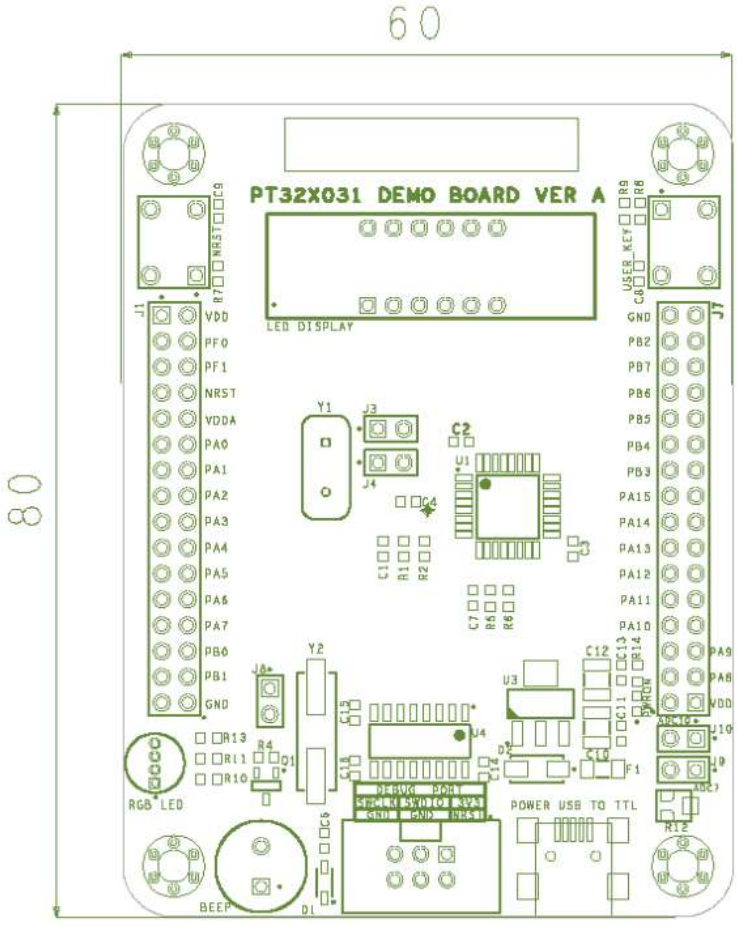

图 3 开发板布局

管脚定义

管脚	主功能	替换功能0	替换功能1	替换功能2	替换功能3	替换功能4	替换功能5	模拟功能	管脚分配
VDD	VDD								
PF0	PF0	OSC_IN	I2C0_SDA	LED-DATA_A					晶振/数码管
PF1	PF1	OSC_OUT	I2C0_SCL	LED-DATA_B					晶振/数码管
NRST	NRST								复位按键
VDDA	VDDA								
PA0	PA0		TIM0_CH1N	LED-DATA_C				ADC_IN0 CMP_OPA_1_P	数码管
PA1	PA1		TIM0_CH2N	LED-DATA_D				ADC_IN1 CMP_OPA_1_N	数码管
PA2	PA2	UART0_TX	TIM0_CH3N	LED-DATA_E	UART1_TX			ADC_IN2 OPA1_OUT	数码管
PA3	PA3	UART0_RX	TIM0_CH4N	LED-DATA_F	UART1_RX			ADC_IN3 CMP_OPA_0_P	数码管
PA4	PA4	SPI0_CS	UART1_TX	LED_DATA_DP	UART0_TX			ADC_IN4 CMP_OPA_0_N	数码管
PA5	PA5	SPI0_SCK	UART1_RX		UART0_RX			ADC_IN5 OPA0_OUT	
PA6	PA6	SPI0_MISO	TIM1_BKIN	LED-DATA_G				ADC_IN6	数码管
PA7	PA7	SPI0_MOSI	TIM1_CH1N	ADC_ETR				ADC_IN7	ADC0
PB0	PB0		TIM1_CH2N						
PB1	PB1	IR_TXD	TIM1_CH3N	LED-COM0				ADC_IN8	数码管
VSS	VSS								
VDD	VDD								
PA8	PA8	MCO	TIM1_CH1						B_LED
PA9	PA9	CMP0_OUT	TIM1_CH2	LED-COM2	UART1_TX	I2C1_SCL	MCO		数码管
PA10	PA10	CMP1_OUT	TIM1_CH3	LED-COM3	UART1_RX	I2C1_SDA			数码管
PA11	PA11	I2C1_SCL	TIM1_CH4	I2C0_SCL					用户按键
PA12	PA12	I2C1_SDA	TIM1_CH4N	I2C0_SDA					
PA13	PA13	SWDIO	IR_TXD						调试口
PA14	PA14	SWCLK		UART1_TX	UART0_TX				调试口
PA15	PA15		SPI1_CS	UART1_RX	UART0_RX				串口
PB3	PB3	TIM0_BKIN	SPI1_SCK					ADC_IN10	ADC1
PB4	PB4	TIM0_CH1	SPI1_MISO					ADC_IN11	R_LED
PB5	PB5	TIM0_CH2	SPI1_MOSI						G_LED
PB6	PB6	TIM0_CH3	I2C1_SCL	UART1_TX	UART0_TX	I2C0_SCL			串口
PB7	PB7	TIM0_CH4	I2C1_SDA	UART1_RX	UART0_RX	I2C0_SDA			
PB2	PB2	BEEP	MCO	LED-COM1				ADC_IN9	数码管/蜂鸣器
VSS	VSS								

表 1 芯片管脚资源分配

开发软件安装

安装MDK5以及配置FLASH驱动

- **注意事项**

1. 安装路径不能带中文，必须是英文路径。
2. 安装目录不能与C51 的 KEIL 或者 KEIL4 冲突，三者目录必须分开

- **获取KEIL5**

要想获得 KEIL5 的安装包，可到 KEIL 的官网下载，我们这里面 KEIL5 的版本是 MDK5.15，后续有新版本可使用更高版本，https://www.keil.com/download/product/

图 4 MDK工具下载路径

- **安装步骤**

1. 双击KEIL5安装包，开始安装，点击“Next”

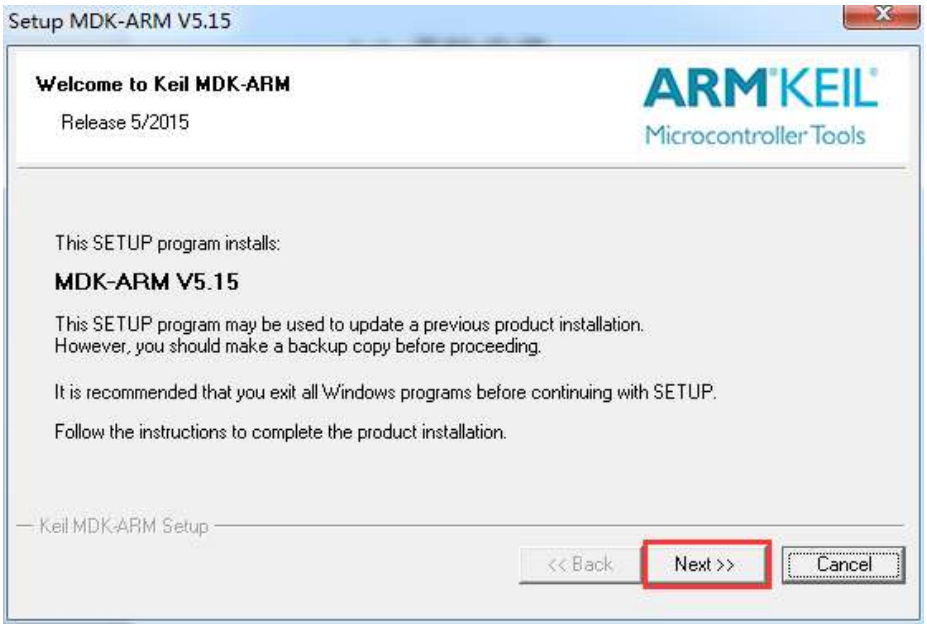

图 5 安装MDK工具

2. 勾选“agree”，并点击“Next”

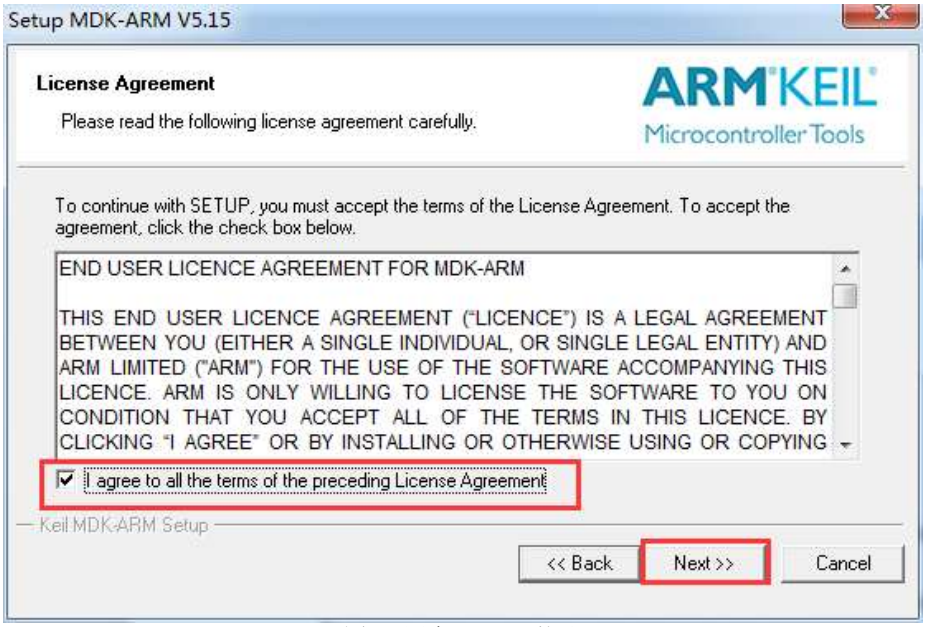

图 6 同意license协议

3. 选择安装路径，路径不能带中文，并点击Next”

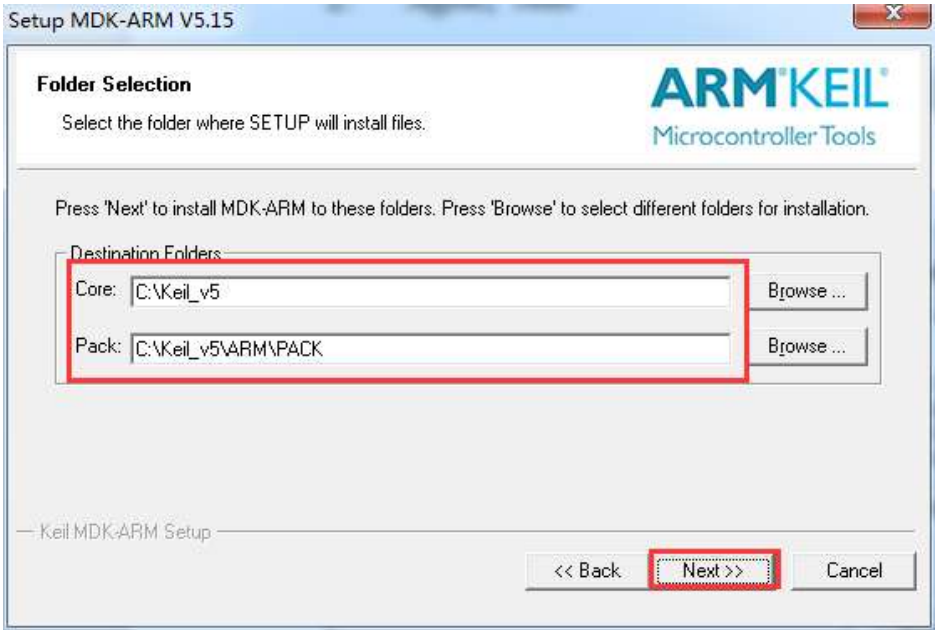

图 7 选择安装路径

4. 填写用户信息，全部填空格（键盘的 space 键）即可，next

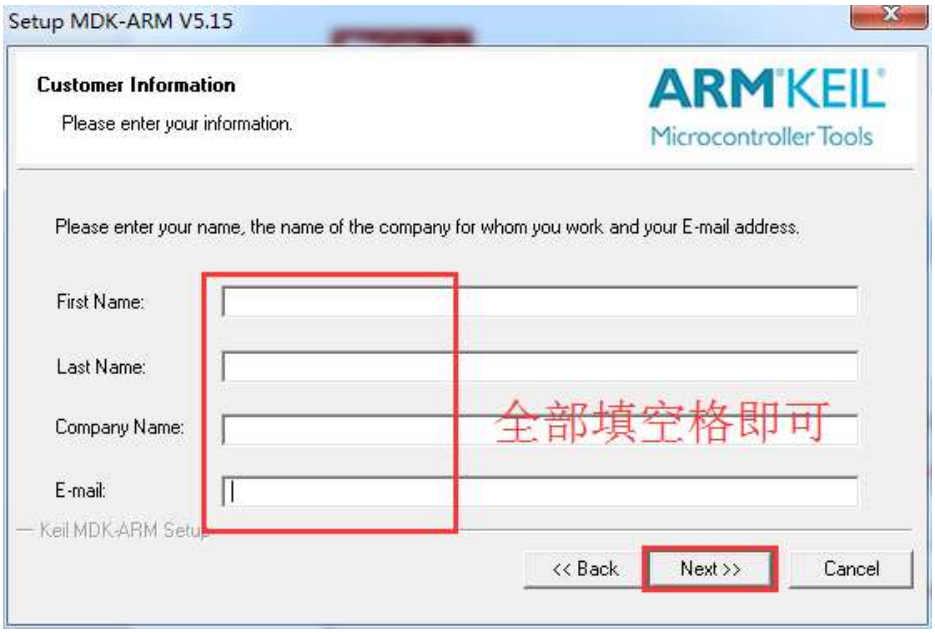

图 8 填写用户信息

5. Finish，安装完毕

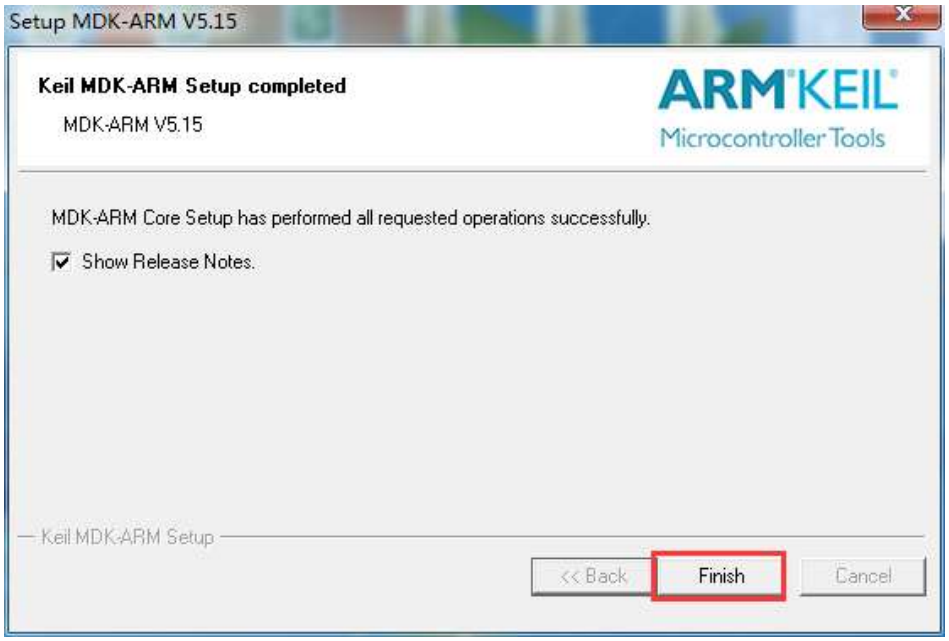

图 9 完成软件安装

- **添加PT32x031的FLASH驱动文件**

PT32x031还未添加到Keil5的MCU库中，因此需要手动添加FLASH的驱动文件后才可以顺利进行代码的Download。

1. 把PT32x031.FLM文件拷贝到KEIL5安装目录下ARM文件夹下的FLASH目录内，参考路径如下（KEIL5安装于C盘路径）：C:\Keil_v5\ARM\Flash
2. 使用KEIL5打开一个Demo例程，点击”Options for target”弹出对话框后选择

“Debug”选型框并点击该选项框内的“Setings”按钮，弹出“Target Driver Setup”的对话框

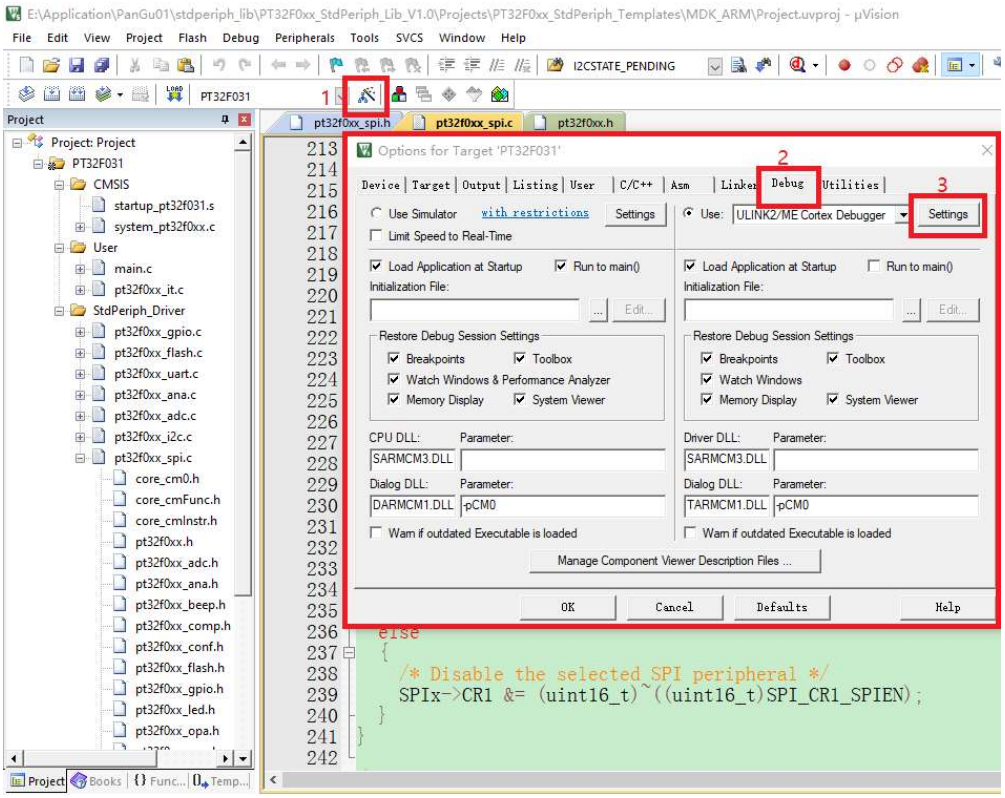

图 10 进入Options for target对话框

3. 在弹出“Target Driver Setup”的对话框中选择“Add”按钮添加FLASH 编程文件；

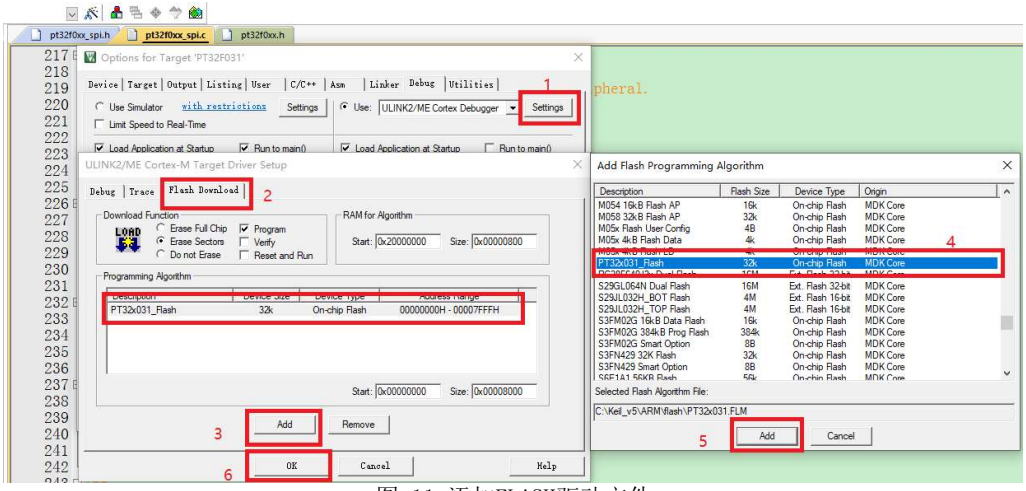

图 11 添加FLASH驱动文件

➢ 确认工具及开发板成功连接

1. 打开MDK5，点击”Options for target”，确认选中正确的调试工具型号

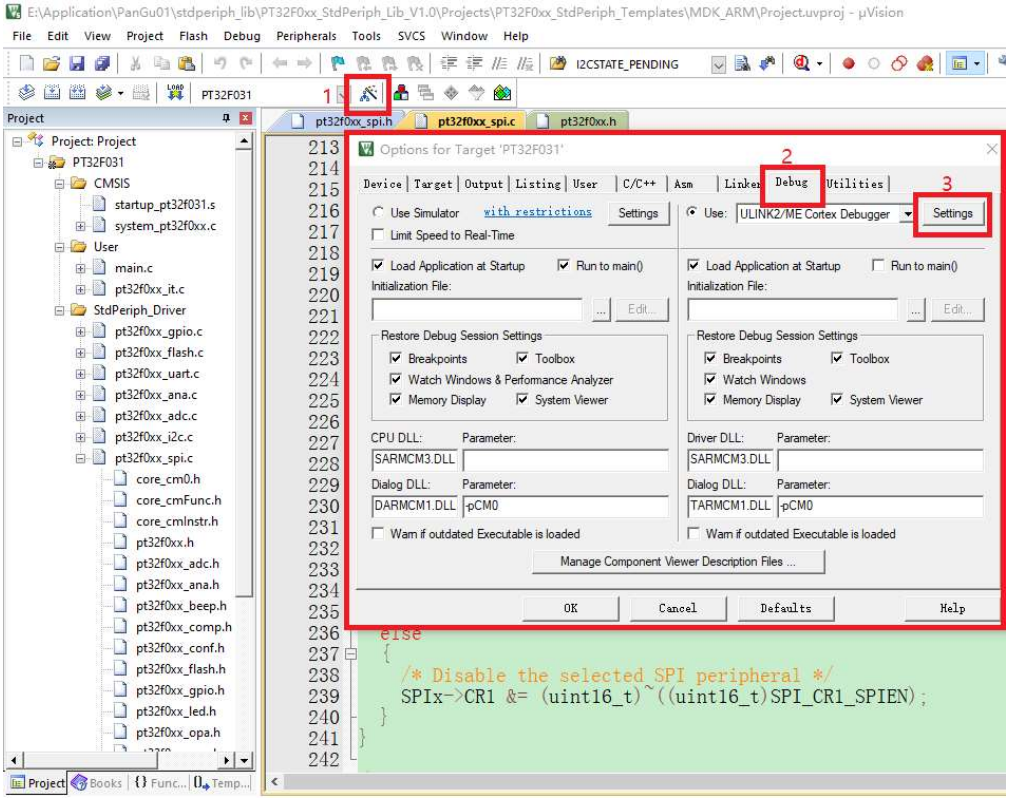

图 12 查看仿真器识别MCU型号

2. 点击Setting，勾选SWJ选项，再下拉列表中选SW模式,再右边的SW Device 下应该能看到开发板芯片信息，表示工具已经正确连接，并可以下载程序到开发板

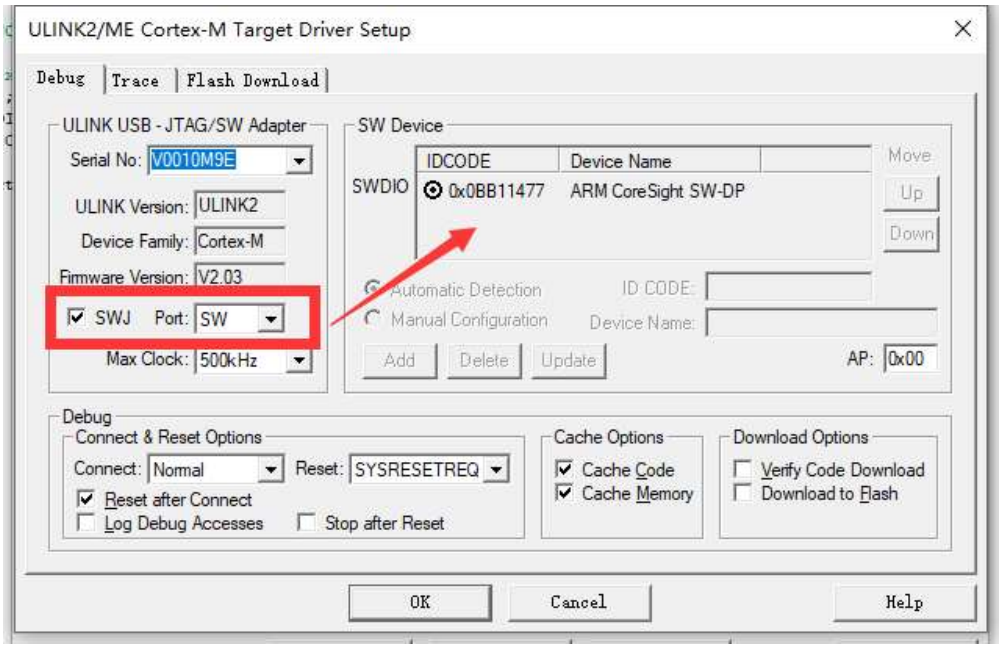

图 13 仿真器识别到MCU型号

➢ 手动添加PT32x031的SVD(系统视图描述)文件

SVD文件中定义了PT32x031中的的每一个外设的硬件寄存器，每一个寄存器中每一个数据位的值，以及详细的说明信息，方便用户在Keil的调试模式下查看所有外设寄存器

1. SVD文件PT32F0xx.SFR目录：..\PT32F0xx_StdPeriph_Lib_V1.2A\Libraries\SVD
2. 打开一个Keil工程点击“Options for Target”并选择“Target”的选项卡

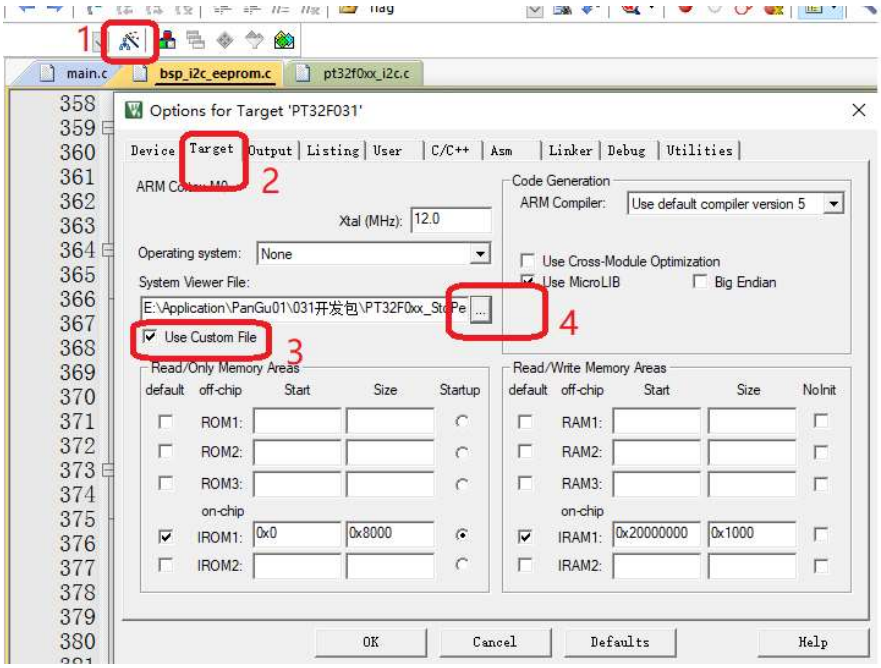

图 14 手动添加SVD文件

3. 进入Debug模式下，选择“Peripperals”菜单下的“System Viewer”选择所需要查看的外设单元寄存器；

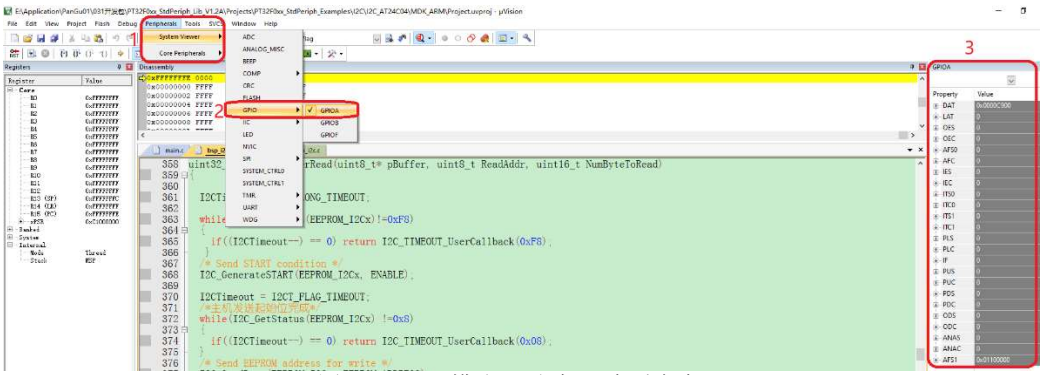

图 15 debug模式下通过SVD查看寄存器

开发包软件目录结构

- 解压 PT32F0xx_StdPeriph_Lib_V1.2A 压缩包的文件结构如下

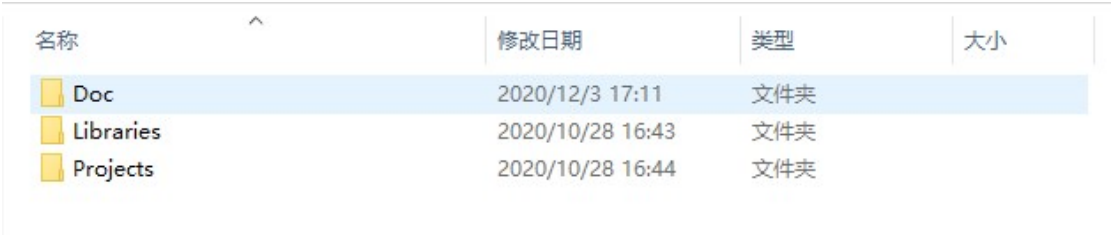

图 16 软件开发包目录结构

- 目录描述

目录	文件夹	内容
Doc	文档 txt 文件	
Libraries	CMSIS	M0 内核驱动库
	PT32F0xx_StdPeriph_Lib_Driver	模块驱动代码
	SVD	寄存器可视化配置文件
Projects	PT32F0xx_StdPeriph_Examples	例子工程
	PT32F0xx_StdPeriph_Templates	项目模板

表 2 开发软件包目录明细

PT32x031时钟系统配置

- **PT32x031的系统时钟有以下时钟源**

1. 外部高速晶体振荡时钟，时钟频率4MHz~25MHz；
2. 高速两倍频时钟，时钟频率64kHz~50MHz。时钟源可以为内部低速RC振荡时钟（时钟频率为32.768kHz），内部高速RC振荡时钟（时钟频率为24MHz）或者外部高速晶体振荡时钟；
3. 内部低速RC振荡时钟，时钟频率32.768kHz；
4. 内部高速RC振荡时钟，时钟频率24MHz；
5. 外部输入高频时钟，时钟频率4MHz~25MHz；

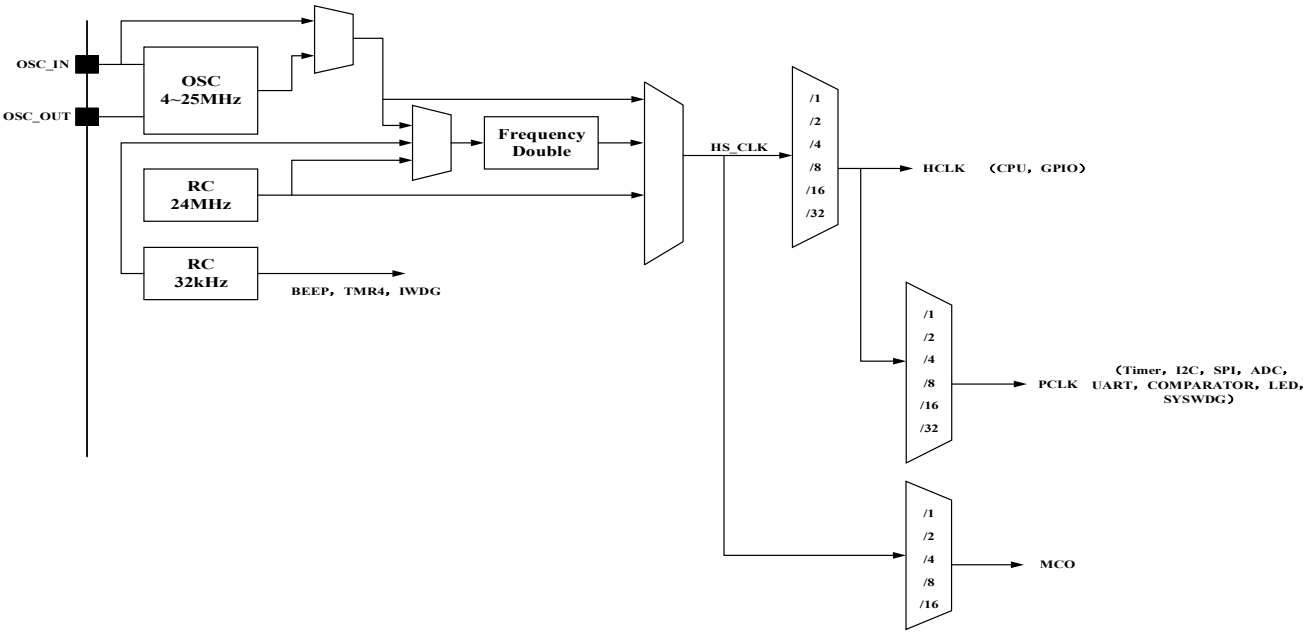

图 17 系统时钟树（HCLK:CPU系统主频，PCLK:外设时钟，MCO：时钟输出管脚）

- **Keil软件配置系统运行时钟注意事项**

1. 打开一个KEIL工程，这里以demo为例简述如何通过向导配置MCU的系统时钟。
2. 双击“Project”选项树中“CMSIS”文件夹下的System_pt32x031.c便可以打开“Configuration Wizard”的界面，在该界面下可以实现MCU所有系统时钟的可视化配置，勾选完所需要的所有配置项之后，重新编译一次工程即可。

注意：如果需要用到外部时钟源（Crystal或者高速时钟），那么用户在向导配置后，在用户代码实现过程中，PF0和PF1作为Crystal的复用管脚不可以变更；

➢ Keil配置步骤

1. 打开工程项目,再项目窗口中打开system_pt32x031.c文件，右边窗口选择configure wizard 选项卡，进行系统时钟配置页面；

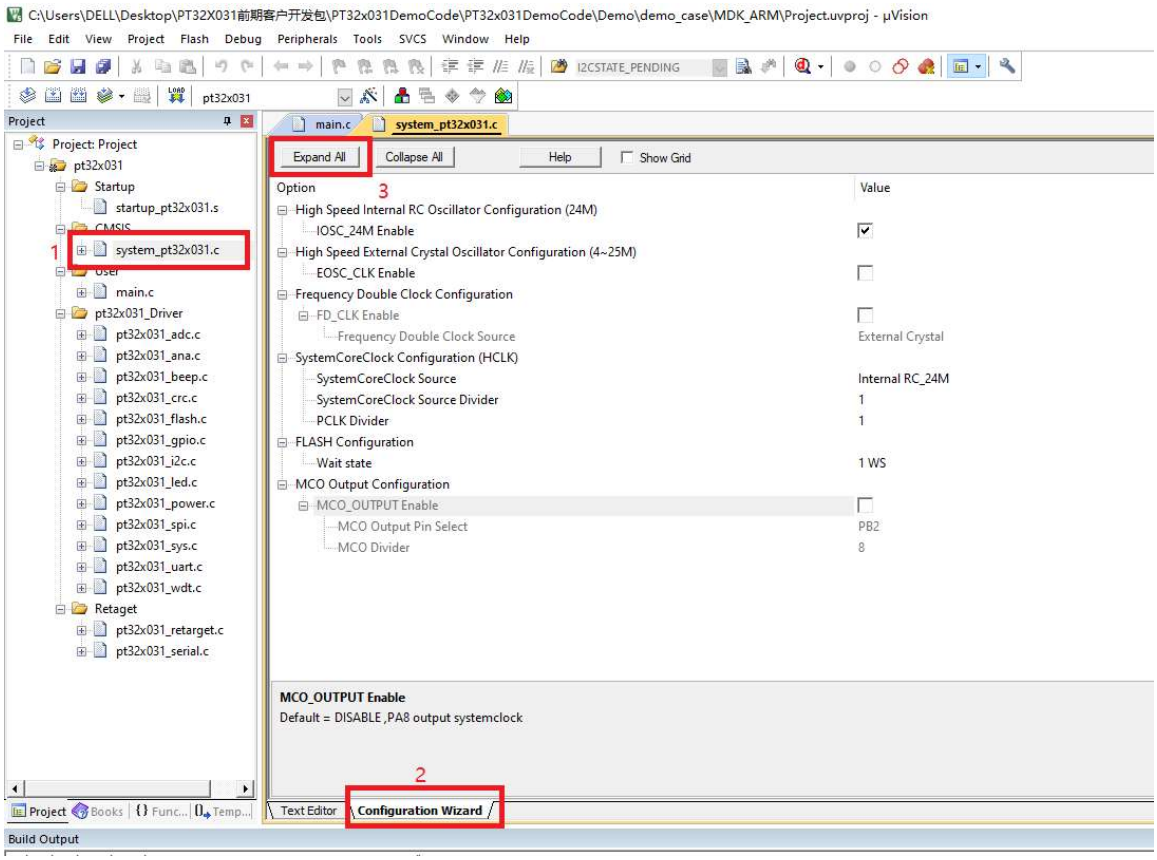

图 18 系统时钟配置向导窗口

2. 配置选项详细说明

图 19 配置选型与系统时钟对应关系

➢ 常用的系统时钟配置方法

1. 时钟源选择内部高速RC 24MHz，系统时钟和外设时钟均为24MHz

1) 使能内部RC 24MHz时钟模块

2) 系统时钟源选择 内部RC 24MHz时钟 作为时钟源

3) 关闭时钟倍频模块

4) 设置HCLK ,PCLK 时钟分频系数为1

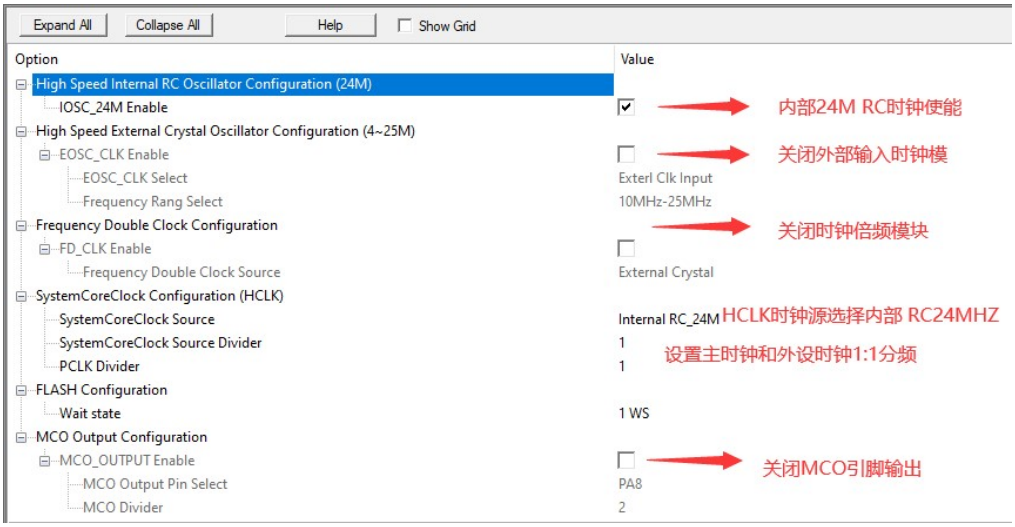

图 20 系统时钟源为内部24MHz RC

2. 系统时钟源选择内部高速RC 24MHz，系统时钟和外设时钟均为48MHz

1) 使能内部RC 24MHz时钟模块

2) 系统时钟源选择选择“倍频模块输出时钟”作为时钟源关闭时钟倍频模块

3) 设置HCLK ,PCLK 时钟分频系数为1

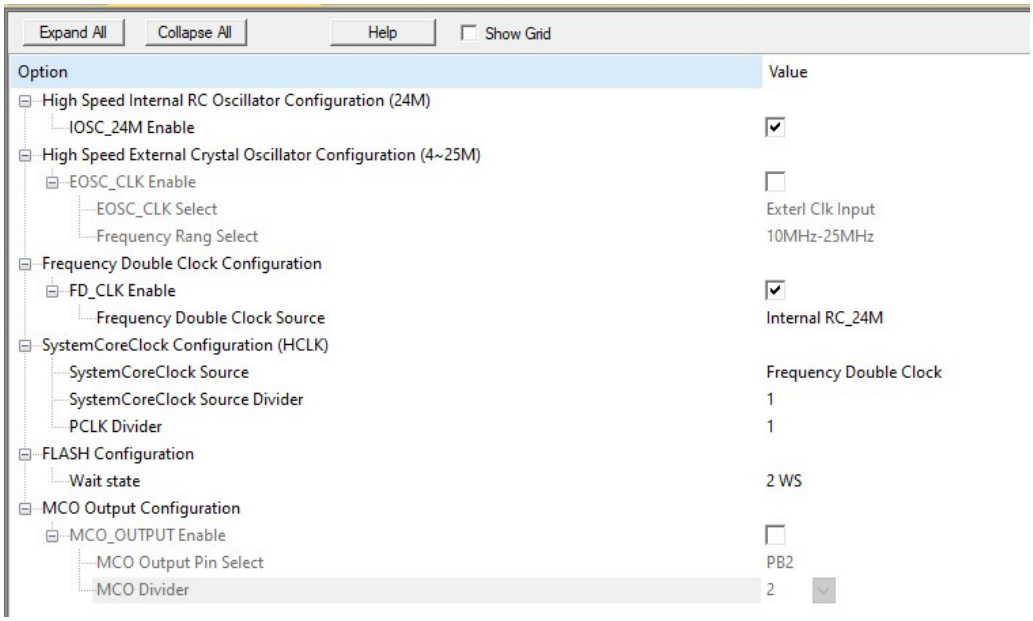

图 21 系统时钟源为内部24MHz RC MCU主频48M

3. 时钟源选择外部晶振24MHz,系统时钟和外设均为24MHz

1) 使能外部时钟模块，并选择时钟源为外部晶体（占用PF0,PF1引脚）

2) 关闭内部RC时钟模块

3) 系统时钟源选择 外部时钟模块 作为时钟源

4) 关闭时钟倍频模块

5) 设置HCLK ,PCLK 时钟分频系数为1

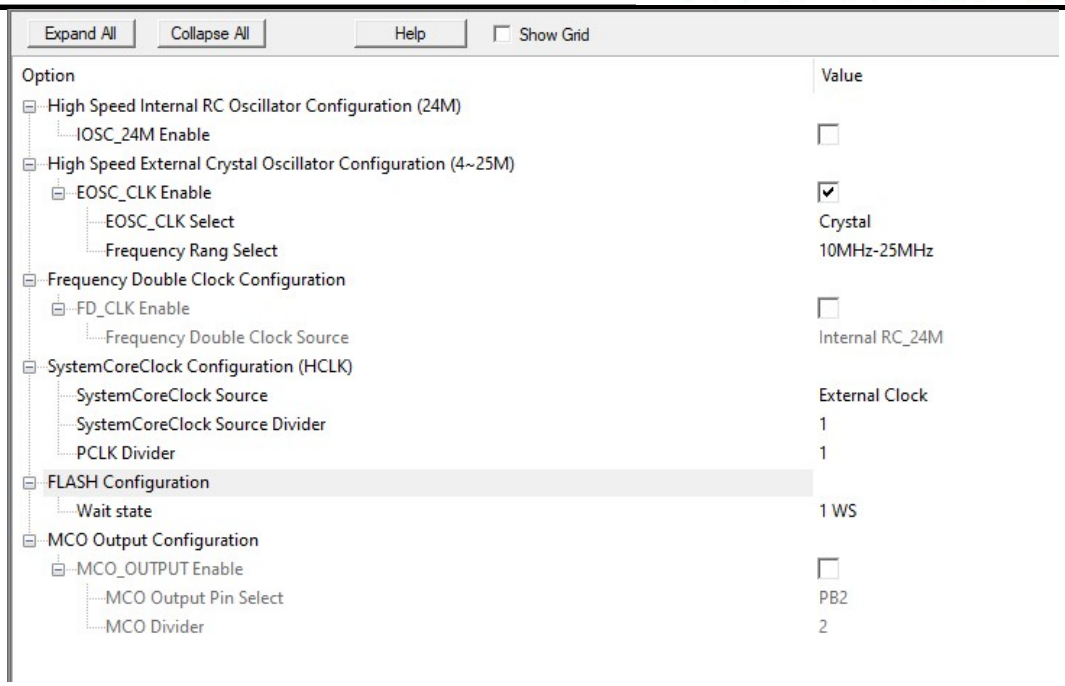

图 22 系统时钟源为外部24M晶体

4. 时钟源选择外部晶振24MHz,系统时钟和外设均为48MHz

1) 使能外部时钟模块，并选择时钟源为外部晶体（占用PF0,PF1引脚）

2) 关闭内部RC时钟模块

3) 系统时钟源选择选择“倍频模块输出时钟”作为时钟源关闭时钟倍频模块

4) 使能时钟倍频模块

5) 设置HCLK ,PCLK 时钟分频系数为1

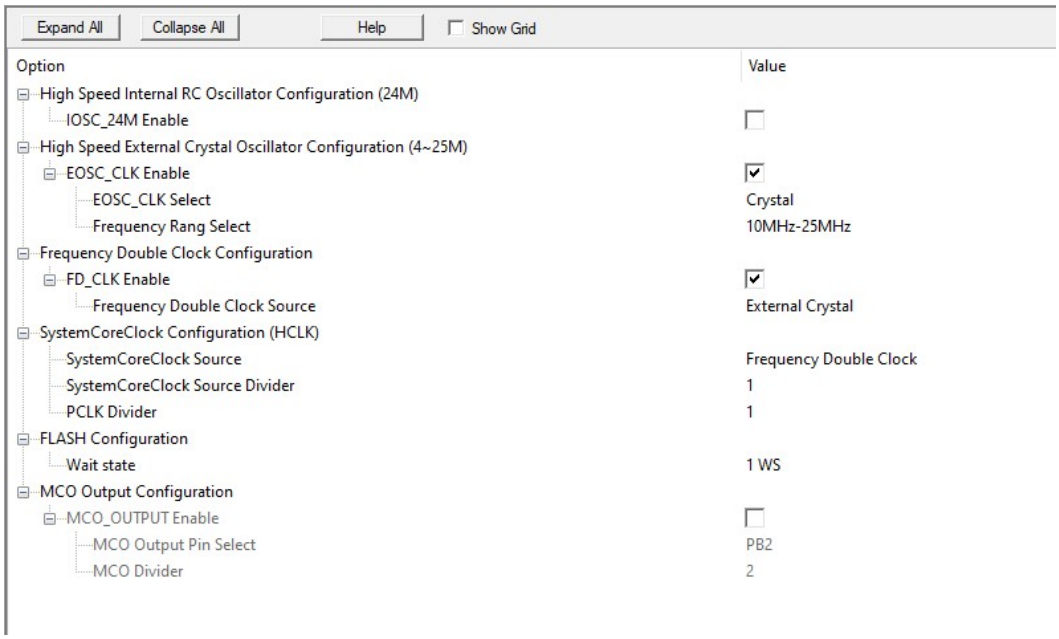

图 23 系统时钟源为外部24M晶体 MCU主频倍频

➢ I/O输入输出例程

1. 通过判别GPIO输入状态进行按键检测

按键在没有被按下的时候，GPIO引脚的输入状态为高电平(按键所在的电路不通，引脚上拉到VDD)，当按键按下时，GPIO引脚的输入状态为低电平(按键所在的电路导通，引脚接到GND)。只要我们检测引脚的输入电平，即可判断按键是否被按下。

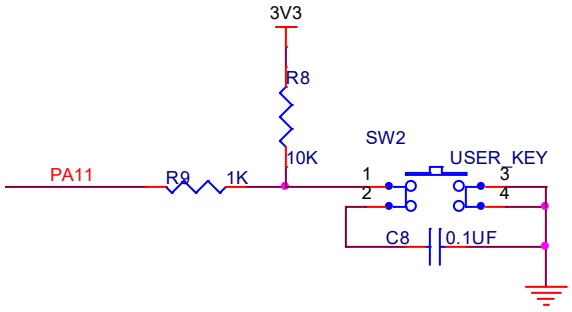

图 24 按键输入硬件原理

2. 通过配置GPIO输出控制RGB灯的亮灭

RGB灯的阳极引出连接到3.3V 电源，阴极各经过1 个限流电阻引入至MCU的3 个GPIO 引脚中，所以我们只要控制这三个引脚输出高低电平，即可控制其所连接LED 灯的亮灭。例如把GPIO 的引脚设置成推挽输出模式并且默认下拉，输出低电平，这样就能让LED 灯亮起来了。

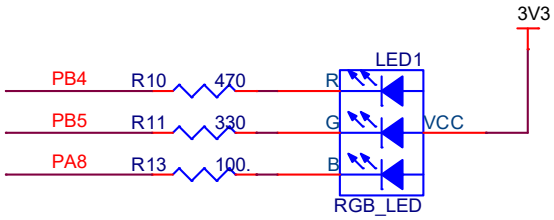

图 25 RGB灯控制硬件原理

3. GPIO例程说明

例程名称	例程说明
GPIO_INPUT&OUTPUT	轮询按键状态(PA11)控制绿灯亮或者灭
GPIO_Interrupt	按键（PA11）的输入中断函数中控制绿灯亮或者灭
GPIO_OUTPUT	轮流点亮 RGB 灯显示红绿蓝黄紫青白七种颜色

表 4 按键例程说明

➢ 蜂鸣器输出例程

1. 蜂鸣器外设可产生250Hz-8KHz的方波，工作时钟由内部32KHz RC提供。

   使用蜂鸣器的时候需要把J8插上短路帽，由于PB2管脚与数码管驱动管脚复用，因此建议拔掉数码显示管后再进行蜂鸣器的编程。

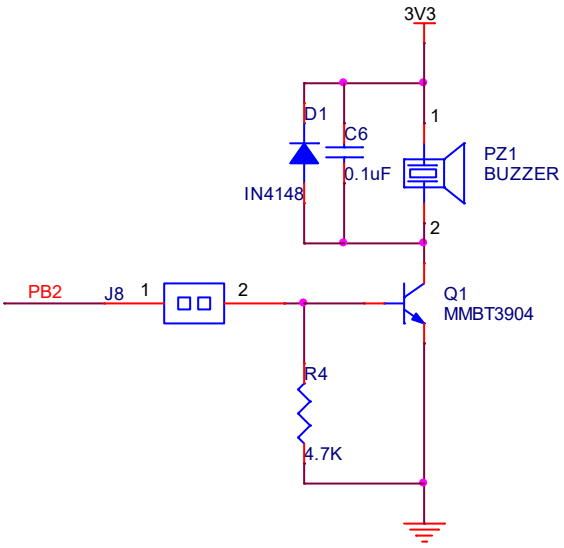

图 26 蜂鸣器控制硬件原理

2. 蜂鸣器例程说明

例程名称	例程说明
BEEP	按键单击，蜂鸣器以频率 1 鸣叫；按键双击，蜂鸣器以频率 2 鸣叫；按键长按，蜂鸣器关闭

表 5 蜂鸣器例程说明

➢ 比较器及OPA功能

1. 比较器用于比较两路模拟输入信号，根据正负两端的输入信号来输出相应的比较结果。比较器的比较结果，可通过寄存器位来查询，用户可根据应用需求，由比较结果产生相应的内部中断，或者将比较结果输出至芯片管脚（比较器0对应PA9的数字输出，比较器1对应PA10的数字输出）,比较器的输出经数字滤波模块，用以滤除不定状态下产生的毛刺，确保内部中断及输出至芯片管脚的比较结果的可靠性。
2. 比较器还可以配置为OPA模式进行工作（OPA0对应PA5模拟输出，OPA2对应PA2模拟输出）开发板预留了用于测试OPA功能的反馈电阻,默认情况下没有焊接，如果需要用到OPA，可以根据实际放大需求加焊不同阻值的电阻。

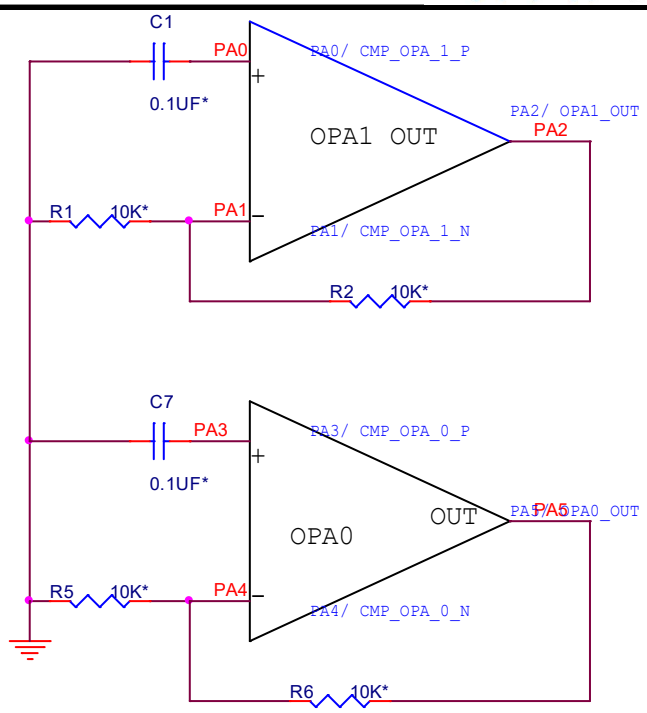

图 27 运放反馈电路图

3. 比较器及OPA例程说明

例程名称	例程说明
COMP	配置 CMP 的复用功能并且使能比较器 0，外灌电压到 PA3 和 PA4 管脚，通过示波器查看比较器输出管脚 PA9 的电平状态
OPA	配置 OPA 的输入复用功能以及输出模拟功能，外灌电压到 PA3 管脚，万用表测量 OPA 输出电压，查看电压值是否与预期一致

表 6 比较器运放例程说明

- **访问片内FLASH**

1. Pt32x031片上提供最大为32KB的Flash存储空间用于存放程序代码，在FLASH的最后一个sector（地址范围0x0000_7E00 – 0x0000_0x7FFF）保存着用户可配置参数信息，这些参数可由用户擦写，芯片上电后自动读取这些参数并映射到对应的寄存器，用户不可访问该区域，可以通过对应的映射寄存器查看参数。该区域内不可运行程序（即PC指针不能往此区域跳转）。

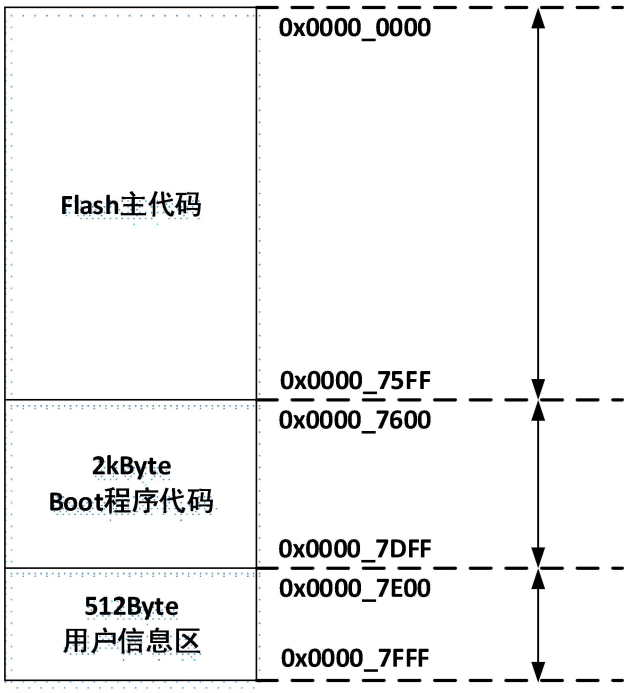

图 28 FLASH地址空间分配

2. FLASH例程说明

例程名称	例程说明
FLASH_Operation	擦除写读片上 FLASH 的一部分地址空间，例程中操作 FLASH 的地址空间为 0x00006E00~0x00007600

表 7 FLASH操作说明

➢ **串口通讯**

1. 为利用UART实现Demo板与电脑通讯，需要用到一个USB转UART的IC，我们选择CH340G 芯片来实现这个功能。板级将CH340G的TXD引脚与UART0的RX（PA15）连接，CH340G的RXD引脚与UART0的TX(PB6)连接。CH340G已经芯片集成在开发板上。

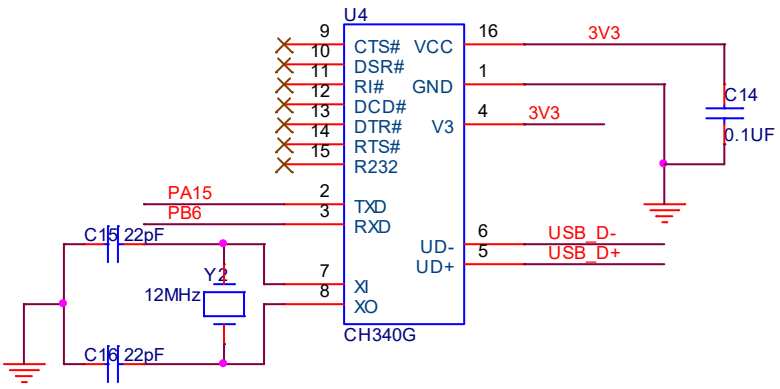

图 29 USB转串口硬件设计

2. 串口例程说明

例程名称	例程说明
UART_Printf	通过串口发送配置指令控制 RGB 灯的颜色
UART_Selftransmitting	通过串口接收中断实现数据的回传
UART_Shell	通过串口实现 Shell 命令

表 8 串口操作说明

注意：熟悉串口操作流程建议使用串口终端工具，比如SecureCRT或者AccessPort

- **ADC电压采集**

1. 开发板载一个贴片滑动变阻器，贴片滑动变阻器的动触点通过连接至MCU的两个ADC通道引脚（PA7和PB3），当我们使用旋转滑动变阻器调节旋钮时，其动触点电压也会随之改变，电压变化范围为0-3.3V，亦是开发板默认的ADC电压采集范围。

注意：使用两路ADC需要把电位计旁边的两个插针插上跳线帽

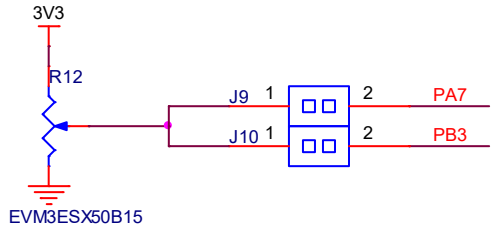

图 30 ADC电位计电路原理图

2. 串口例程说明

例程名称	例程说明
ADC_Continue	ADC 连续采集模式，采集到的数据显示到数码管
ADC_Continue_Disable	间隔打开以及关闭 ADC 连续采集模式，采集到的数据显示到数码管
ADC_KEY	按键中断中处理一次 ADC 单次转换，采集到的数据显示到数码管
ADC_Software	软件循环处理 ADC 单次转换，采集到的数据显示到数码管
ADC_Timer	定时器触发 ADC 转换，采集到的数据显示到数码管

表 9 ADC操作说明

- **数码管显示**

1. Pt32x031支持驱动7段数码管显示功能，最大支持4位数码管显示，COM口的灌电流最大支持110mA，因此通过I/O口可以直接驱动数码管显示。软件可配置T1 和T2 的时间来达到流水显示数字的目的，每次一个数字点亮T1 时间后，软件可选择是否产生中断。T1 和T2 的时间软件可配。T2 时间段，GPIO 的输出使能会被强制关闭。具体的寄存器定义以及配置流程请参考用户手册。

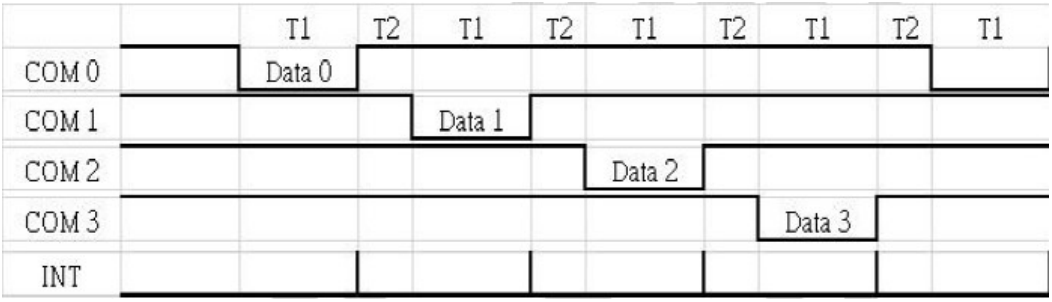

图 31 数码管驱动波形

2. Demo板提供了一个数码管座子，方便不使用数码管的时候可以直接移除数码管，对应的数码管驱动管脚可以复用为其他功能。

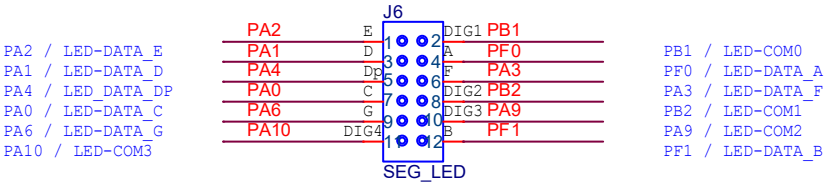

图 32 数码管电路图

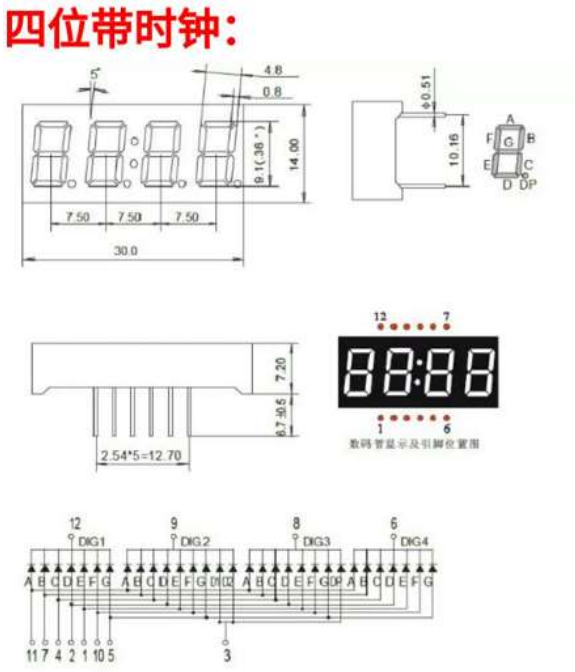

图 33 数码管脚位排布

3. 数码管驱动例程说明

例程名称	例程说明
LED_DISPLAY	系统定时器中断中轮流显示 0-8 之间的数字

表 10 数码管驱动例程操作说明

- **PWM及捕获（高级定时器）**

1. PT32x031内设置有2个16位宽的高级定时器，TMR0、TMR1，带8位宽的预分频，支持递增/递减/递增递减交替计数模式，各支持4路捕捉输入和4路单独PWM输出功能，支持刹车输入。

2. 高级定时器例程说明

例程名称	例程说明
Advance_Timer	配置高级定时器 TIMER0,定时周期为 1S 进行 LED 灯的闪烁
Capture_KeyPressTime	捕获两次按键的时间间隔
LED_SingleColorBreathing	PWM 实现呼吸灯功能
PWM_3CH_TmrOv_ADC	TIMER 作为 ADC 触发源进行 ADC 数据采集
TIM_ComplementarySignals	带死区功能的互补 PWM 输出

表 11 PWM例程说明

- **普通定时器**

1. PT32x031内设置有2个16位宽的普通定时器TIMER2和TIMER3，工作于外设时钟下，可选择递增或者递减的计数方式。

2. 普通定时器例程说明

例程名称	例程说明
Base_Timer	配置高级定时器 TIMER3,定时周期为 1S 进行 LED 灯的闪烁

表 12 普通定时器例程说明

- **低功耗模式**

1. PT32x031有3种工作模式，正常工作模式，休眠模式和深度休眠模式。其中休眠模式和深度休眠模式为低功耗模式。使用ARM Cortex-M0的Wait for Interrupt (WFI)和wait for Event (WFE)两条指令可以使芯片进入休眠模式或深度睡眠模式。当执行WFI或WFE指令后，芯片进入哪种低功耗模式，由系统控制寄存器（SCR）的SLEEPDEEP位决定，具体请参考用户手册关于低功耗章节的详细描述。

注意：WFE为事件触发方式，无需开启NVIC或者中断使能也可进行唤醒

2. 芯片进入休眠模式后，可以通过以下方式唤醒：
   1) 外部复位
   2) 调试模式请求
   3) 所有使能的中断源（WFE方式休眠唤醒无需使能中断）
3. 芯片进入深度休眠模式后，可以通过以下方式唤醒：
   1) 外部复位
   2) 外部中断（GPIO的电平中断）（WFE方式休眠唤醒无需使能中断）
   3) 内部中断（tmr4中断，IWDG中断，比较器中断）（WFE方式休眠唤醒无需使能中断）
   4) 调试模式请求
4. 低功耗例程说明

例程名称	例程说明
Deepsleep_GPIO_WFE	深度休眠，通过按键唤醒
Deepsleep_TIMER4_WFE	深度休眠，通过 TIMER4 唤醒
Sleep_GPIO_WFE	普通休眠，通过按键唤醒

表 13 低功耗例程说明

- **低电压监控**

1. 当芯片的供电电压低于安全值时，LVD检测电压支持 4V，3.5V，3V，2.75V，2.5V，2.2V，2.0V，1.8V，当电压低于设定的阈值时，产生LVD中断或者复位。设置寄存器详见模拟功能杂项控制寄存器中的LVD控制寄存器以及系统控制寄存器中的复位使能控制寄存器；
2. LVD例程说明

例程名称	例程说明
PVD_2V	关闭 LVD 复位，配置 LVD 为 2V，如果发生 LVD 复位则产生中断，在 LVD 中断中点亮红灯

表 14 低电压检测例程说明

- SPI

1. SPI通讯是全双工的，发送或接收数据都是高位在先，在数据发送的同时进行数据接收。按通讯时钟的不同提供方式，SPI通讯分主机和从机两种。SPI主机的时钟由本地产生，通讯的发起和结束完全由自己主动控制；SPI从机的时钟为外部输入（来自SPI主机），被动响应主机的通讯。
2. PT32x031的SPI模块支持主/从模式工作. 基本特性如下：

1) 帧数据传输4~16位可编程
2) 主机最高传输速率为10Mbps（系统时钟为20MHz时）
3) 主机模式下支持4路从机片选输出
4) 最大8级接收FIFO缓冲队列
5) 最大8级发送FIFO缓冲队列
6) 内部时钟预分频和后分频实现主机可编程SPI传输速率

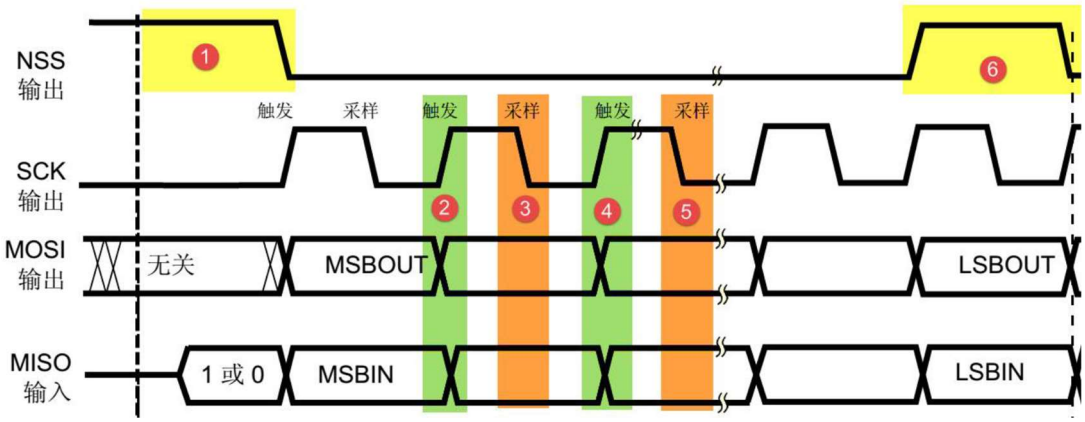

图 34 SPI通信时序

3. 取决于SPI模块控制寄存器中的SPO（时钟极性）位和SPH（时钟相位）位设定，SPI共有4中不同的工作模式，分别为模式0、模式1、模式2和模式3，实际中采用较多的是模式0和模式3。

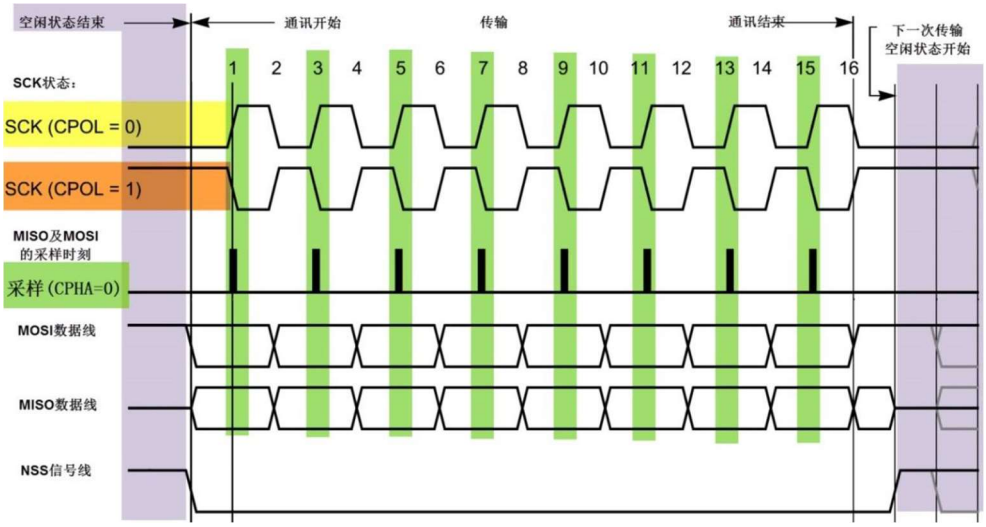

图 35 SPH=0 时的SPI 通讯模式

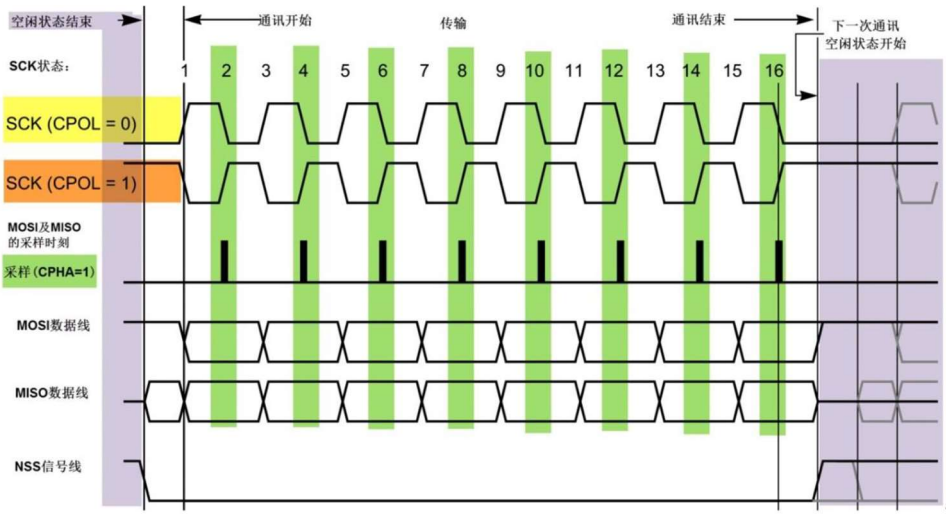

图 34 SPH=1 时的SPI 通讯模式

SPI 模式	SPO	SPH	空闲时 SCK	采样时刻
0	0	0	低电平	奇数边沿
1	0	1	低电平	偶数边沿
2	1	0	高电平	奇数边沿
2	1	1	高电平	偶数边沿

表 15 SPI的四种工作模式

1. SPI例程说明

例程名称	例程说明
SPI_MasterInt_SlaveInt_mode00	SPI0 和 SPI1 配置为主从互发模式，中断接收
SPI_MasterPoll_SlaveInt_mode03	SPI0 和 SPI1 配置为主从互发模式，轮询接收
SPI_W25Q64	访问 SPI FLASH 器件 W25Q64，开发板无该器件，需要外接

表 16 SPI操作例程说明

- IIC

1. PT32x031片内I2C模块支持主模式和从模式通讯方式，其基本特性如下
   1) 内含并行数据/串行I2C协议转换器
   2) 提供总线仲裁机制，支持多个主机并存
   3) 支持7位从机寻址模式
   4) 支持广播呼叫
   5) 提供数据发送和接收状态标识
   6) 提供字节传输结束标识
   7) 通讯错误检测
   8) 支持标准（100Kbps）或快速（400Kbps）I2C通讯时序
2. 作为I2C主机时，其特性包括：
   9) 产生总线通讯时钟
   10) 实现总线仲裁
   11) 7位地址寻址从机，并控制数据读写方向
   12) 产生总线通讯起始位、重复起始位和停止位
   13) 检测通讯错误
3. 作为I2C从机时，其特性包括：
   14) 检测起始位、重复起始位和停止位
   15) 可编程I2C7位地址匹配寻址
   16) 传输过程中检测起始和停止条件
   17) 检测通讯错误
4. PT32x031芯片的I2C模块有多个事件可产生同一个中断标识（I2C_CTLSET_SI）。各事件由I2C_STAT寄存器中的位[7:3]状态信息，在外设工作时，控制逻辑会根据外设的工作状态修改“状态寄存器(SR)”，我们只要读取这些寄存器相关的寄存器位，就可以了解I2C的工作状态

事件	I2C_STAT[7:3]	SR	I²C 总线状态	I2C_CTLSET 相关位信息				I²C 模块的下一步应对
				STA	STO	SI	AA	
1	00000	0x00	主模式或从模式被寻址下 I²C 通讯错误	0	1	0	X	I²C 总线释放
2	00001	0x08	主模式发送起始位(START) 完成	X	0	0	X	发送从机地址+写控制位，接收 ACK
3	00010	0x10	主模式发送重复起始位(Repeated START) 完成	X	0	0	X	同发送起始位（00001） 发送从机地址+读控制位，主机转入接收模式
4	00011	0x18	主模式发送从机地址+写控制位完成，收到 ACK 响应	0	0	0	X	发送数据字节并接收 ACK
				1	0	0	X	发送重复起始位
				0	1	0	X	发送停止位，STO 标志置位
				1	1	0	X	停止位后紧接着发送起始位，STO 标志清零
5	00100	0x20	主模式发送从机地址+写控制位完成，无 ACK 响应	0	0	0	X	发送数据字节并接收 ACK
				1	0	0	X	发送重复起始位
				0	1	0	X	发送停止位，STO 标志置位
				1	1	0	X	停止位后紧接着发送起始位，STO 标志清零
6	00101	0x28	主模式发送数据字节完成，收到 ACK 响应	0	0	0	X	发送数据字节并接收 ACK
				1	0	0	X	发送重复起始位
				0	1	0	X	发送停止位，STO 标志置位
				1	1	0	X	停止位后紧接着发送起始位，STO 标志清零
7	00110	0x30	主模式发送数据字节完成，无 ACK 响应	0	0	0	X	发送数据字节并接收 ACK
				0	0	0	X	发送重复起始位
				1	0	0	X	发送停止位，STO 标志置位
				1	0	0	X	停止位后紧接着发送起始位，STO 标志清零
8	00111	0x38	主模式在发送地址或数据时总线仲裁丢失	0	0	0	X	I²C 总线被释放
				1	0	0	X	I²C 总线空闲时发送起始位
9	01000	0x40	主模式发送从机地址+读控制位完成，收到 ACK 响应	0	0	0	0	接收数据，无 ACK 响应
				1	0	0	1	接收数据，有 ACK 响应

事件	I2C_STAT[7:3]	SR	I²C 总线状态	I2C_CTLSET 相关位信息				I²C 模块的下一步应对
				STA	STO	SI	AA	
10	01001	0x48	从机地址+读控制位发送完成，无 ACK 响应	1	0	0	X	发送重复起始位
				0	1	0	X	发送停止位，STO 标志置位
				1	1	0	X	停止位后紧接着发送起始位，STO 标志清零
11	01010	0x50	数据字节接收完成，回送 ACK 响应	0	0	0	0	接收数据，无 ACK 响应
				0	0	0	1	接收数据，有 ACK 响应
12	01011	0x58	数据字节接收完成，回送 NACK 响应	1	0	0	X	发送重复起始位
				0	1	0	X	发送停止位，STO 标志置位
				1	1	0	X	停止位后紧接着发送起始位，STO 标志清零
13	01100	0x60	从模式地址+写控制位接收完成，回送 ACK 响应	X	0	0	0	接收数据，回应 NACK
				X	0	0	1	接收数据，回应 ACK
14	01101	0x68	主模式下总线仲裁丢失，作为从模式收到地址+写控制位，并回送 ACK 响应	X	0	0	0	接收数据，回应 NACK
				X	0	0	1	接收数据，回应 ACK
15	01110	0x70	收到广播寻址（0x00），并回送 ACK 响应	X	0	0	0	接收数据，回应 NACK
				X	0	0	1	接收数据，回应 ACK
16	01111	0x78	主模式下总线仲裁丢失，作为从模式收到广播寻址，并回送 ACK 响应	X	0	0	0	接收数据，回应 NACK
				X	0	0	1	接收数据，回应 ACK
17	010000	0x80	从模式被寻址下收到一个数据字节，，并回送 ACK 响应	X	0	0	0	接收数据，回应 NACK
				X	0	0	1	接收数据，回应 ACK
18	10001	0x88	从模式被寻址下收到一个数据字节，并回送 NACK 响应	0	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址
				0	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址
				1	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址；当总线空闲时将发送起始位
				1	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址；当总线空闲时将发送起始位
19	10010	0x90	广播被寻址下收到一个数据字节，并回送 ACK 响应	X	0	0	0	接收数据，回应 NACK
				X	0	0	1	接收数据，回应 ACK
20	10011	0x98	广播被寻址下收到一个数据字节，并回送 NACK 响应	0	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址
				0	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址
				1	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址；当总线空闲时将发送起始位
				1	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址；当总线空闲时将发送起始位

事件	I2C_STAT[7:3]	SR	I²C 总线状态	I2C_CTLSET 相关位信息				I²C 模块的下一步应对
				STA	STO	SI	AA	
21	10100	0xA0	从模式被寻址下收到停止位（STOP）或重复起始位（Repeated START）	0	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址
				0	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址
				1	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址；当总线空闲时将发送起始位
				1	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址；当总线空闲时将发送起始位
22	10101	0xA8	从模式地址+读控制位接收完成，回送 ACK 响应	X	0	0	0	发送最后一个字节，接收 ACK 应答
				X	0	0	1	发送一个字节，接收 ACK 应答
23	10110	0xB0	主模式下总线仲裁丢失，作为从模式收到地址+读控制位，并回送 ACK 响应	X	0	0	0	发送最后一个字节，接收 ACK 应答
				X	0	0	1	发送一个字节，接收 ACK 应答
24	10111	0xB8	从模式发送数据字节完成，收到 ACK 响应	X	0	0	0	发送最后一个字节，接收 ACK 应答
				X	0	0	1	发送一个字节，接收 ACK 应答
25	11000	0xC0	从模式发送数据字节完成，无 ACK 响应	0	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址
				0	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址
				1	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址；当总线空闲时将发送起始位
				1	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址；当总线空闲时将发送起始位
26	11001	0xC8	从模式发送最后一个数据字节完成，收到 ACK 响应	0	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址
				0	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址
				1	0	0	0	模块转为非寻址下的从模式，不再识别地址和广播寻址；当总线空闲时将发送起始位
				1	0	0	1	模块转为非寻址下的从模式，可识别地址和广播寻址；当总线空闲时将发送起始位
27	11111	0xF8	总线空闲	0	0	0	0	-

表 17 IIC事件与状态表

5. IIC通讯过程

1) IIC作为MASTER并且向外部发送数据的过程

a) 控制产生起始信号(S)，当发生起始信号后，它产生事件2，并会对SR寄存器写0x08，表示起始信号已经发送；

b) 紧接着发送设备地址+“写”的控制位并等待应答信号，若有从机应答，则产生事件“4”，并会对SR寄存器写0x18，若无从机应答，则产生事件“6”，并会对SR寄存器写0x20;

c) 以上步骤正常执行后，往I2C 的“数据寄存器DR”写入要发送的数据， I2C 外

设通过SDA 信号线一位位把数据发送出去后,若有从机应答，又会产生“6”事件，并会对SR寄存器写0x28，重复这个过程，就可以发送多个字节数据了；

d) 当我们发送数据完成后，控制I2C 设备产生一个停止信号(P)，这个时候会产生27事件，并会对SR寄存器写0xF8，表示通讯结束,总线空闲状态。

总线动作	S	ADD （W）	ACK	D1	ACK	D2	ACK	…	D2	ACK	P
状态寄存器	0x08		0x18		0x28		0x28			0x28	

表 18 IIC发送过程

2) IIC作为MASTER并且从外部接收数据的过程

a) 控制产生起始信号(S)，当发生起始信号后，它产生事件2，并会对SR寄存器写0x08，表示起始信号已经发送；

b) 紧接着发送设备地址+“写”的控制位并等待应答信号，若有从机应答，则产生事件“4”，并会对SR寄存器写0x18，若无从机应答，则产生事件“6”,并会对SR寄存器写0x20;

c) 以上步骤正常执行后，往I2C 的“数据寄存器DR”写入要发送的数据， I2C 外设通过SDA 信号线一位位把数据发送出去后,若有从机应答，又会产生“6”事件，并会对SR寄存器写0x28，重复这个过程，就可以发送多个字节数据了；

d) (3) 从机端接收到地址后，开始向主机端发送数据。当主机接收到这些数据后，会产生“11”事件，并会对SR寄存器写0x50，表示接收数据寄存器非空，我们读取该寄存器后，可对数据寄存器清空，以便接收下一次数据。此时我们可以控制I2C 发送应答信号(ACK)或非应答信号(NACK)，若应答，则重复以上步骤接收数据，若非应答，则停止传输；当我们发送数据完成后，控制I2C 设备产生一个停止信号(P)，这个时候会产生27事件，并会对SR寄存器写0xF8，表示通讯结束,总线空闲状态。

总线动作	S	ADD （R）	ACK	D1	ACK	D2	ACK	…	D2	NACK	P
状态寄存器	0x08		0x50		0x50		0x50			0x50	

表 19 IIC接收过程

6. IIC例程说明

例程名称	例程说明
I2C_AT24C04	通过 IIC 访问 EEPROM，轮询方式，开发板无该器件，需要外接
I2C_MasterWr_SlaveRd	IIC0 与 IIC1 分别作为主从互相发送合接收，中断方式

表 20 IIC例程说明

- **系统定时器**

1. SysTick—系统定时器是属于CM0内核中的一个外设，内嵌在NVIC 中。系统定时器是一个24bit 的向下递减的计数器，计数器每计数一次的时间为1/SYSCLK，一般我们设置系统时钟SYSCLK 等于24M。当重装载数值寄存器的值递减到0 的时候，系统定时器就产生一次中断，以此循环往复。
2. 系统定时器例程说明

例程名称	例程说明
SYSTICK	采用系统定时器产生 us/ms 级的延时，并以 1S 的时基控制 LED 闪烁

- **Demo例程说明**

1. 本例程是比较完整的DEMO程序，主要用来验证开发板的硬件是否正常，主要包含以下功能:
   1) 按键事件中断程序响应
   2) 4位数码管驱动
   3) PWM驱动RGB灯多彩变化
   4) 按键控制LED灯
   5) ADC转换
   6) 4位数码管显示ADC转换值
   7) 串口打印ADC采样的电压值
2. 工程路径:
   \PT32F0xx_StdPeriph_Lib_V1.1A\Projects\PT32F0xx_StdPeriph_Examples\Demo\Demo_Main\MDK_ARM
3. 基于开发板操作说明
   1) 代码编译成功后，烧写到F031开发板
   2) 接上4位数码管，数码管显示来自PB3引脚的输入电压的ADC转换结果.
   3) 同时电脑段打开串口工具，波特率设置为19200,8,N,1,可以看到打印的数据
   4) 程序启动PWM0定时器，并启动周期溢出中断来更新PWM占空比，在PB4,PB5输出PWM信号点亮LED灯。
   5) 按键检测配置为中断模式，按住按键，PA8引脚大约每300毫米输出状态翻转一次。（LED灯闪烁）