STM32F030 使用手册

作者：向仔州

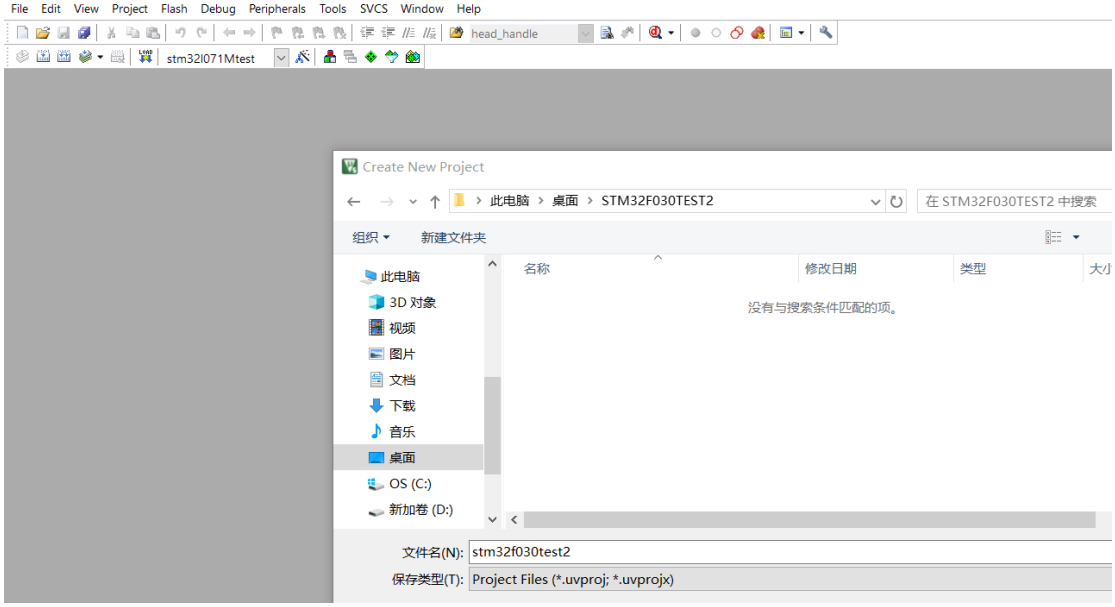
目录

开发环境搭建 2
GPIO 输入输出功能 7
GPIO 输出功能 7
延时函数实现 7
GPIO 输入功能 8
GPIO 外部中断触发功能 8
串口 1 发送接收实现 10
串口 1 发送 10
串口 1 打印输出实现 11
串口 1 接收中断实现 11
串口 1 DMA 空闲中断接收数据 13
定时器 TIME3 使用 15

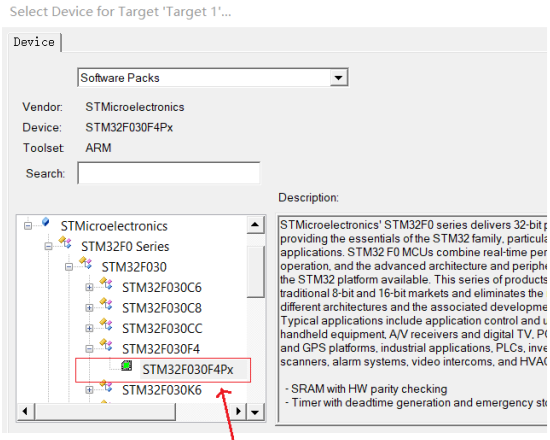
开发环境搭建

MDK5.0 keil 开发安装环境按照网上的安装

Keil.STM32F0xx_DFP.2.1.1 MDK 安装完成之后，安装 STM32F0 补丁包。

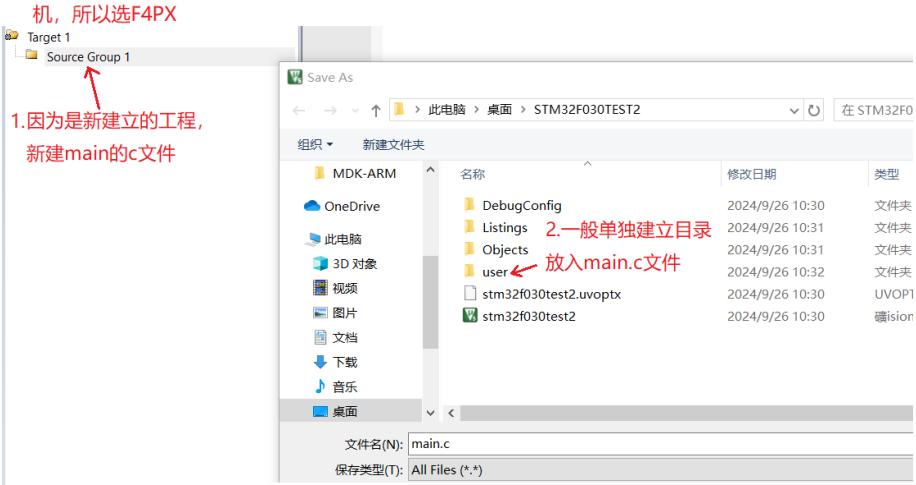


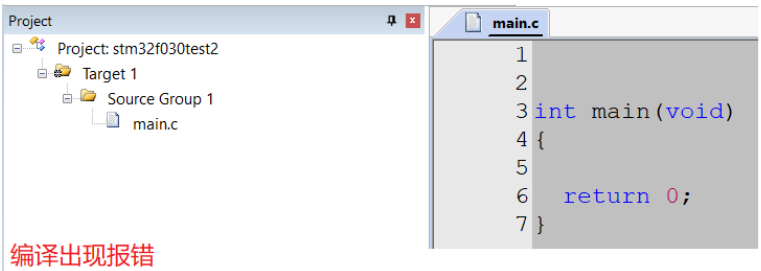
新建 STM32F0 工程



因为我们使用的是STM32F030F4P6 引脚20个的小单片机，所以选F4PX

keil5会弹出该对话框，点击下面OK就是了。

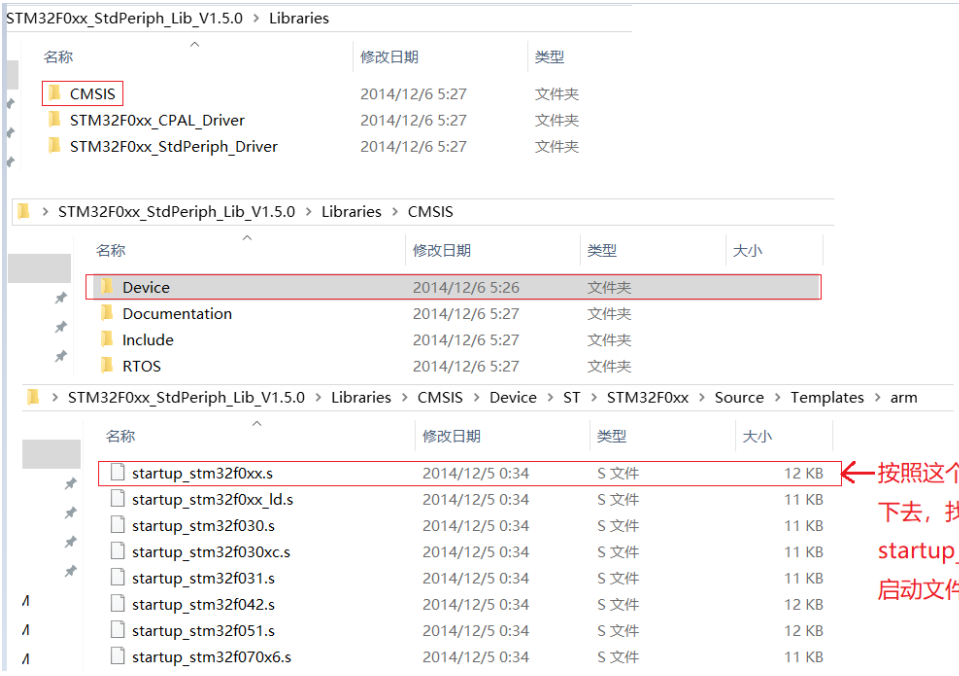




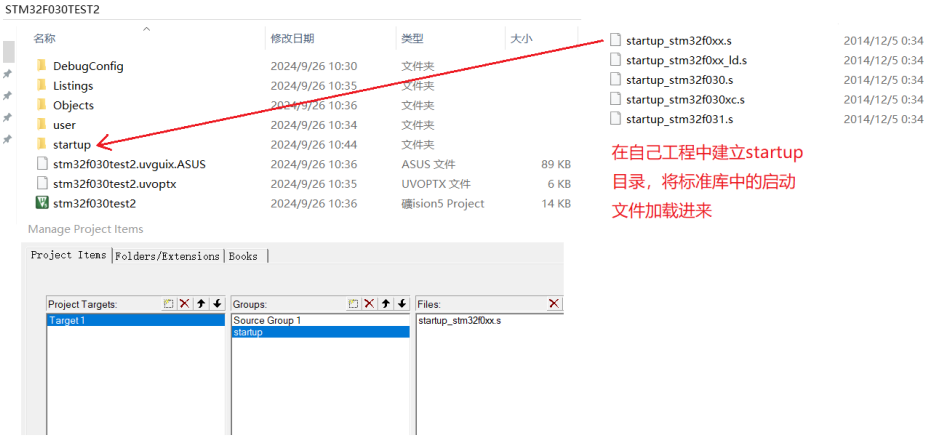
编译出现报错

error: L6236E: No section matches selector - no section to be FIRST/LAST.

去网上寻找标准库STM32F0xx_StdPeriph_Lib_V1.5.0

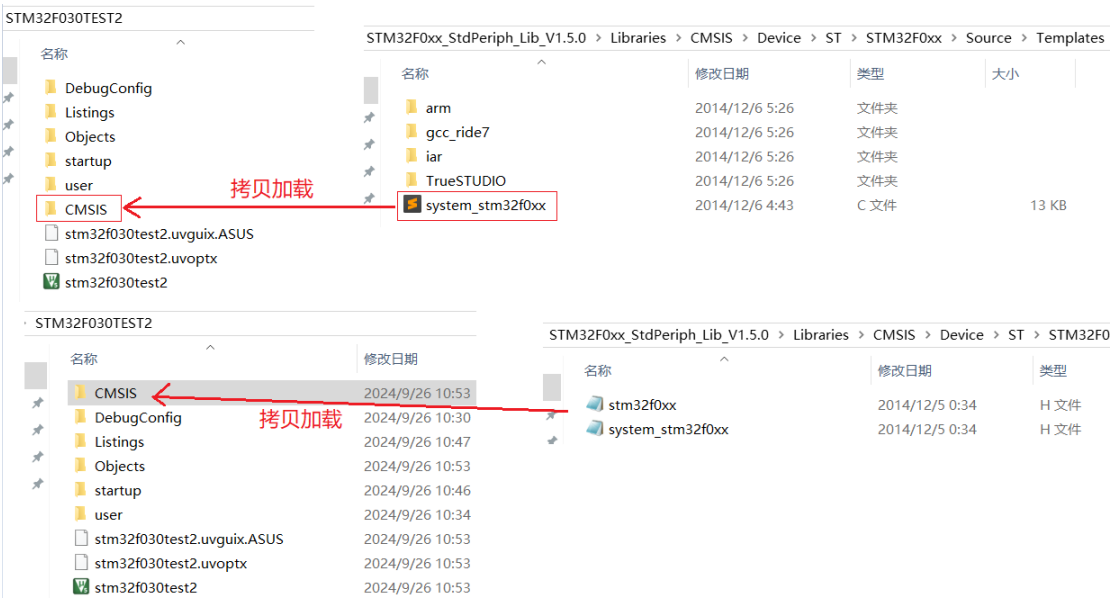


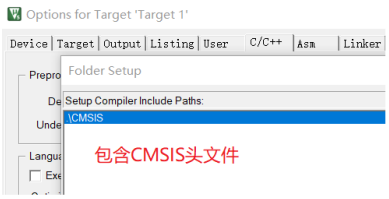
按照这个路径一路找下去，找到启动文件startup_stm32f0xx.s启动文件



在自己工程中建立startup目录，将标准库中的启动文件加载进来

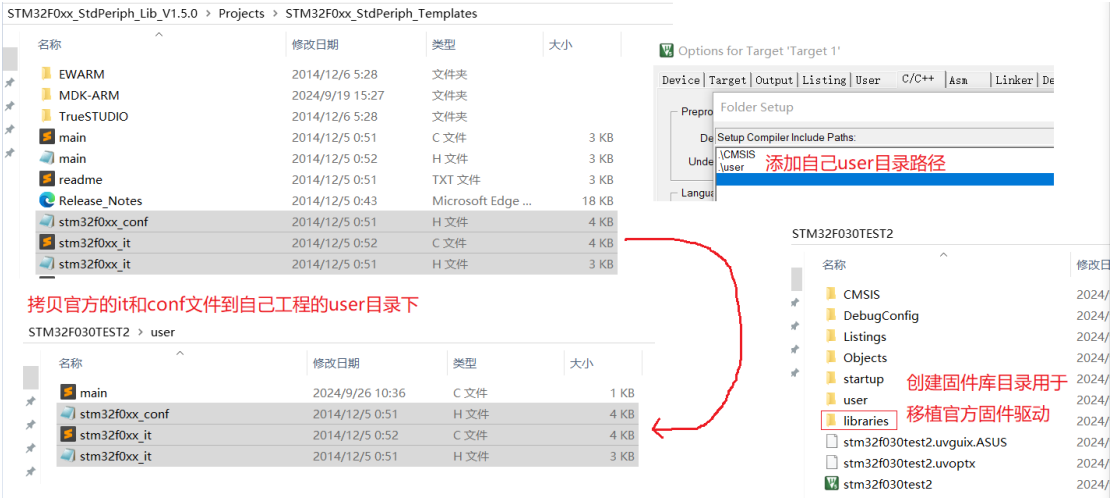
进行再次编译，还是报错 Error: L6218E: Undefined symbol SystemInit (referred from startup_stm32f0xx.o).



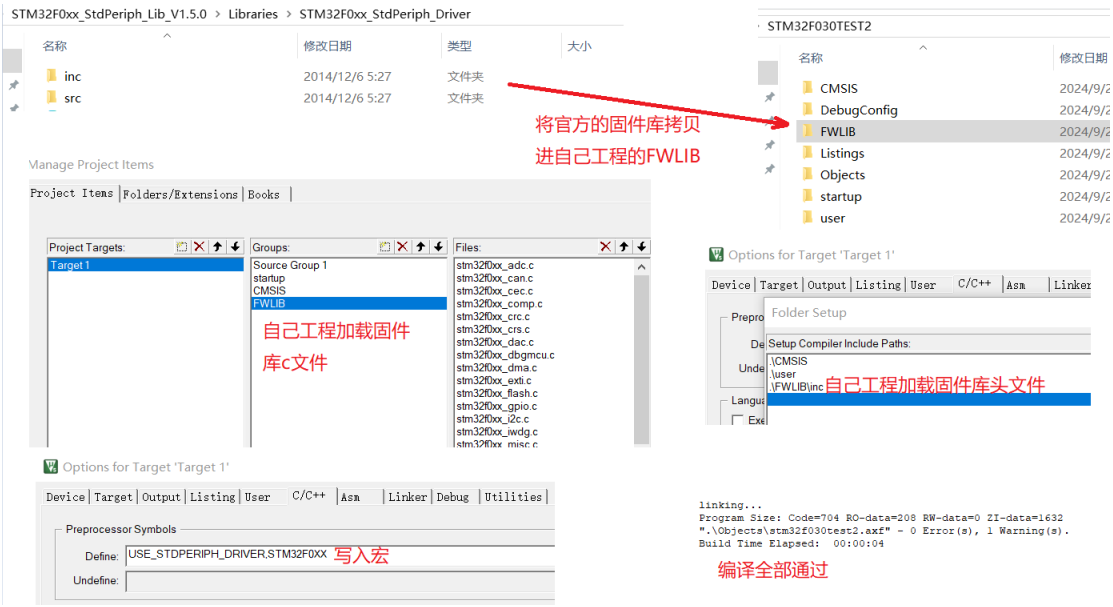


再次编译

error: #35: #error directive: "Please select first the target STM32F0xx device used in your application (in stm32f0xx.h file)"



我将 libraries 目录改名炜 FWLIB



USE_STDPERIPH_DRIVER,STM32F0XX

在 SystemInit 函数中， 系统默认使用的是外部时钟起振配置。

void SystemInit (void){

……

SetSysClock();

}

以下就是外部时钟起振代码

```
static void SetSysClock(void)
{
  __IO uint32_t StartUpCounter = 0, HSEStatus = 0;

  /* SYSCLK, HCLK, PCLK configuration ----------------------------------------*/
  /* Enable HSE */
  RCC->CR |= ((uint32_t)RCC_CR_HSEON);

  /* Wait till HSE is ready and if Time out is reached exit */
  do
  {
    HSEStatus = RCC->CR & RCC_CR_HSERDY;
    StartUpCounter++;
  } while((HSEStatus == 0) && (StartUpCounter != HSE_STARTUP_TIMEOUT));

  if ((RCC->CR & RCC_CR_HSERDY) != RESET)
  {
    HSEStatus = (uint32_t)0x01;
  }
  else
  {
    HSEStatus = (uint32_t)0x00;
  }

  if (HSEStatus == (uint32_t)0x01)
  {
    /* Enable Prefetch Buffer and set Flash Latency */
    FLASH->ACR = FLASH_ACR_PRFTBE | FLASH_ACR_LATENCY;

    /* HCLK = SYSCLK */
    RCC->CFGR |= (uint32_t)RCC_CFGR_HPRE_DIV1;

    /* PCLK = HCLK */
    RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE_DIV1;

    /* PLL configuration = HSE * 6 = 48 MHz */
    RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_PLLSRC | RCC_CFGR_PLLXTPRE | RCC_CFGR_PLLMULL));
    RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLSRC_PREDIV1 | RCC_CFGR_PLLXTPRE_PREDIV1 | RCC_CFGR_PLLMULL6);

    /* Enable PLL */
    RCC->CR |= RCC_CR_PLLON;

    /* Wait till PLL is ready */
    while((RCC->CR & RCC_CR_PLLRDY) == 0)
```

```
    {
    }

    /* Select PLL as system clock source */
    RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_SW));
    RCC->CFGR |= (uint32_t)RCC_CFGR_SW_PLL;

    /* Wait till PLL is used as system clock source */
    while ((RCC->CFGR & (uint32_t)RCC_CFGR_SWS) != (uint32_t)RCC_CFGR_SWS_PLL)
    {
    }
  }
  else
  { /* If HSE fails to start-up, the application will have wrong clock
         configuration. User can add here some code to deal with this error */
  }
}
```

现在改为内部时钟起振代码

```
static void SetSysClock2(void)
{
        __IO uint32_t StartUpCounter = 0, HSIStatus = 0;
        /* SYSCLK, HCLK, PCLK configuration ----------------------------------------*/
        /* Enable HSI*/ //使能内部时钟
  RCC->CR |= ((uint32_t)RCC_CR_HSION);

        //等待内部时钟起振
        do
  {
    HSIStatus = RCC->CR & RCC_CR_HSIRDY;
    StartUpCounter++;
  } while((HSIStatus== 0) && (StartUpCounter != HSI_STARTUP_TIMEOUT));

        if ((RCC->CR & RCC_CR_HSIRDY) != RESET)
  {
                HSIStatus = (uint32_t)0x01;
  }
  else
  {
                HSIStatus = (uint32_t)0x00;
  }
        if (HSIStatus == (uint32_t)0x01)
        {
                /* Enable Prefetch Buffer and set Flash Latency */ //flash 总线时钟使能
                FLASH->ACR = FLASH_ACR_PRFTBE | FLASH_ACR_LATENCY;
                /* HCLK = SYSCLK *///外设 AHB 总线时钟等于系统时钟
                RCC->CFGR |= (uint32_t)RCC_CFGR_HPRE_DIV1;
                /* PCLK = HCLK *///外设 APB 总线时钟等于系统时钟
                RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE_DIV1;
                /* PLL configuration = HSI/2 * 12= 48 MHz */
                RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_PLLSRC | RCC_CFGR_PLLSRC | RCC_CFGR_PLLMULL));
                RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLSRC_HSI_Div2 | RCC_CFGR_PLLMULL12); //RC 时钟 2 分频后  进行 12 倍频
                /* Enable PLL *///使能锁相环倍频开关
                RCC->CR |= RCC_CR_PLLON;
                /* Wait till PLL is ready *///等待锁相环就绪
                while((RCC->CR & RCC_CR_PLLRDY) == 0);
                /* Select PLL as system clock source *///选择锁相环输出时钟作为系统时钟
                RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_SW));
                RCC->CFGR |= (uint32_t)RCC_CFGR_SW_PLL;
                /* Wait till PLL is used as system clock source *///等待锁相环输出时钟已经成为系统时钟
                while ((RCC->CFGR & (uint32_t)RCC_CFGR_SWS) != (uint32_t)RCC_CFGR_SWS_PLL);
        }
        else
        {

        }
}
```

```
  /* Reset PREDIV1[3:0] bits */
  RCC->CFGR2 &= (uint32_t)0xFFFFFFF0;

  /* Reset USARTSW[1:0], I2CSW, CECSW and ADCSW bits */
  RCC->CFGR3 &= (uint32_t)0xFFFFFEAC;

  /* Reset HSI14 bit */
  RCC->CR2 &= (uint32_t)0xFFFFFFFE;

  /* Disable all interrupts */
  RCC->CIR = 0x00000000;

  /* Configure the System clock frequency, AHB/APBx prescalers and Flash settings */
//  SetSysClock();
  SetSysClock2();//内部时钟启动
}
```

KEIL 配置成 SWD 模式，下载测试下。

GPIO 输入输出功能

GPIO 输出功能

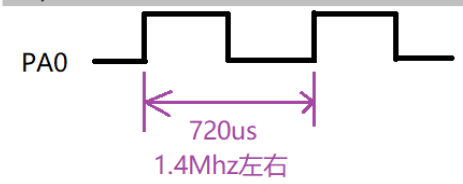
测试PA0输出电平

DSIO	6	
IO	7	PA0/USART1_CTS/ADC_IN0/RTC_TAMP2/WKUP1
	8	PA1/USART1_RTS/ADC_IN1

因为我使用的内部时钟HSI经过分频，倍频之后是48Mhz

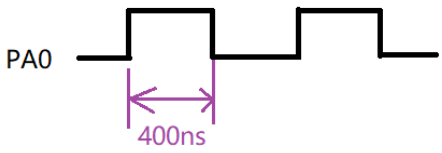
```
#include "stm32f0xx.h"

int main(void)
{
  GPIO_InitTypeDef        GPIO_InitStructure;
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);

  /* 配置PA0 */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 ;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT; //输出模式
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP; //全速输出
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; //输出速率50Mhz
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL; //无上下拉输出
  GPIO_Init(GPIOA, &GPIO_InitStructure);


  while(1)
  {
    GPIO_SetBits(GPIOA,GPIO_Pin_0); //PA0 = 1
    GPIO_ResetBits(GPIOA,GPIO_Pin_0); //PA0 = 0
  }
```

所以我测试IO口最大电平切换速度



1.4Mhz左右

PA0两次高电平时间间隔是720us/1.4Mhz左右，达不到50M输出的频率，这是因为硬件所限制。



2.5Mhz左右 高低电平变换

延时函数实现

```
/*****************************
*微秒延时实现
*usec:传入微秒
*****************************/
void delay_us(uint32_t usec)
{
  for(uint32_t i = 0; i < usec ; i++)
  {
    for(uint32_t i = 0; i < 4; i++);//实测1.08us
  }
}
```

内部晶振时钟实现的

```c
/*****************************
*毫秒延时实现
*msec:传入毫秒
*****************************/
void delay_ms(uint32_t msec)
{
  for(uint32_t i = 0; i < msec ; i++)
  {
    delay_us(1350);//1.02ms
  }
}
```

以上是 48Mhz 内部 HSI 时钟下实现的延时函数

GPIO 输入功能

```c
int main(void)
{
  GPIO_InitTypeDef        GPIO_InitStructure;
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);

  /* 配置PA0 */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 ;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN; //输入模式
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP; //上拉输入
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  USART_Config();
  while(1)
  {
    if(GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_0) == 1)
    {
      printf("PA0 = 1\r\n");
    }
    else
    {
      printf("PA0 = 0\r\n");
    }
    delay_ms(500);
  }
```

GPIO PA0上拉输入，默认就是输入高电平

```
PA0 = 1
PA0 = 1
PA0 = 1
PA0 = 1
PA0 = 1
PA0 = 1
PA0 = 1
PA0 = 1
PA0 = 1
PA0 = 1
PA0 = 0
PA0 = 0
PA0 = 0
PA0 = 0
PA0 = 0
PA0 = 0
```

PA0接VCC，或者悬空

PA0接GND

GPIO 外部中断触发功能

```c
/************************
PA0 引脚外部中断初始化
************************/
void EXIT_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;
  EXTI_InitTypeDef   EXTI_InitStructure;
  NVIC_InitTypeDef   NVIC_InitStructure;

  RCC_APB2PeriphClockCmd(RCC_AHBPeriph_GPIOA,ENABLE);
  /* Enable SYSCFG clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);
  /* EXTI0中断线连接PA0*/
  SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOA, EXTI_PinSource0);

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN; //输入模式
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;//输入上拉
  GPIO_Init(GPIOA, &GPIO_InitStructure);
```

PA0上拉电阻到高电平

```
  /* 配置 EXTI0 中断线 */
  EXTI_InitStructure.EXTI_Line = EXTI_Line0;
  EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
  EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling; //下降沿触发
  EXTI_InitStructure.EXTI_LineCmd = ENABLE;
  EXTI_Init(&EXTI_InitStructure);

  /* 中断线0中断优先级 */
  NVIC_InitStructure.NVIC_IRQChannel = EXTI0_1_IRQn; //中断线0中断服务函数
  NVIC_InitStructure.NVIC_IRQChannelPriority = 0x00;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);

}
```

之前用的上升沿模式，IO配置的下拉电阻，发现PA0也只有接触低电平才触发，所以干脆改成了下降沿模式

```
void EXTI0_1_IRQHandler(void)
{
  if(EXTI_GetITStatus(EXTI_Line0) != RESET)
  {
    printf("EXTI0 PA0 Interrupt....\r\n");

    /* 清除中断线0 中断标准位 */
    EXTI_ClearITPendingBit(EXTI_Line0);
  }
}
```

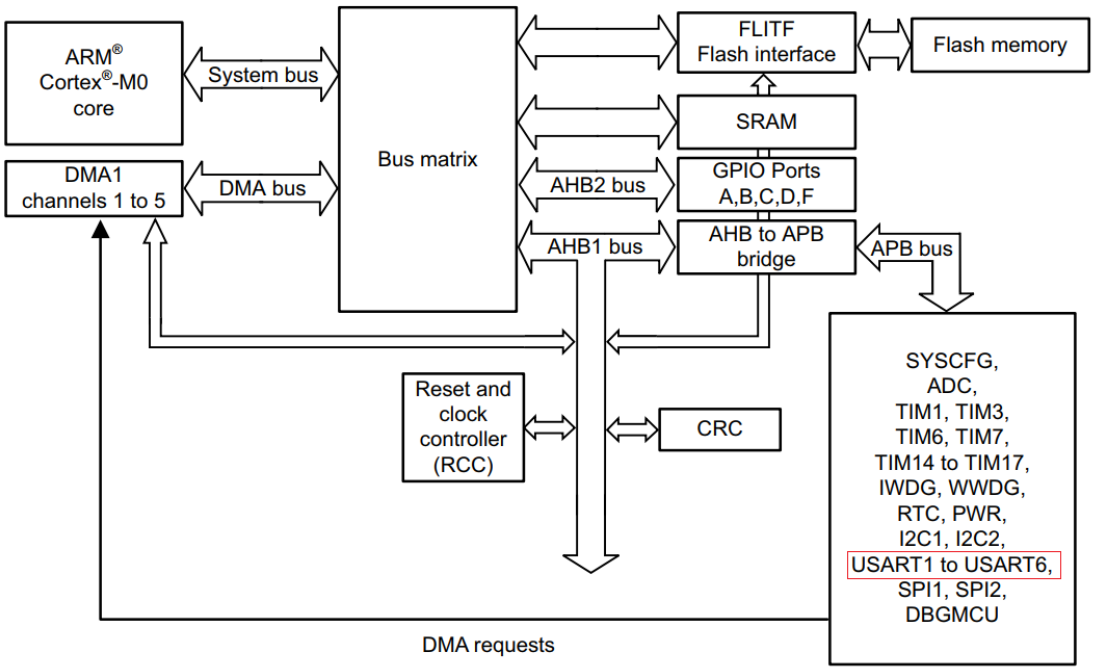
```
static void USART_Config(void);
void EXIT_Init(void);

int main(void)
{
  USART_Config();
  EXIT_Init();
  while(1)
  {
    printf("111111\r\n");
    delay_ms(500);
  }

  return 0;
}
```

```
111111
111111
111111
111111
111111
EXTI0 PA0 Interrupt....
111111
111111
111111
111111
EXTI0 PA0 Interrupt....
111111
111111
111111
EXTI0 PA0 Interrupt....
EXTI0 PA0 Interrupt....
111111
111111
```

中断被触发

串口 1 发送接收实现

串口 1 发送



串口外设都挂在APB总线上

Table 12. Alternate functi

Pin name	AF0	AF1
PA0	-	USART1_CTS(1)
		USART2_CTS(2)
PA1	EVENTOUT	USART1_RTS(1)
		USART2_RTS(2)
PA2	TIM15_CH1(2)	USART1_TX(1)
		USART2_TX(2)
PA3	TIM15_CH2(2)	USART1_RX(1)
		USART2_RX(2)
PA4	SPI1_NSS	USART1_CK(1)
		USART2_CK(2)
PA5	SPI1_SCK	-
PA6	SPI1_MISO	TIM3_CH1
PA7	SPI1_MOSI	TIM3_CH2
PA8	MCO	USART1_CK
PA9	TIM15_BKIN(2)	USART1_TX
PA10	TIM17_BKIN	USART1_RX

串口引脚复用功能

```c
static void USART_Config(void)
{
  GPIO_InitTypeDef  GPIO_InitStructure;
  USART_InitTypeDef USART_InitStructure;

  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA,ENABLE); //打开PA口时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE); //启动串口1时钟

  /* 配置PA10 RX, PA9 TX */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10 |  GPIO_Pin_9;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF; //串口是复用模式
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  GPIO_PinAFConfig(GPIOA,GPIO_PinSource9,GPIO_AF_1); //PA9串口1复用
  GPIO_PinAFConfig(GPIOA,GPIO_PinSource10,GPIO_AF_1);//PA10串口1复用
  USART_InitStructure.USART_BaudRate = 115200;
  USART_InitStructure.USART_WordLength = USART_WordLength_8b;
  USART_InitStructure.USART_StopBits = USART_StopBits_1;
  USART_InitStructure.USART_Parity = USART_Parity_No;
  USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowCont
  USART_InitStructure.USART_Mode =  USART_Mode_Tx;
  USART_Init(USART1,&USART_InitStructure);
  USART_Cmd(USART1,ENABLE);//启用串口1
}
```

```c
static void USART_Config(void);

int main(void)
{
  USART_Config();

  while(1)
  {
    USART_SendData(USART1, 0xF1);
    delay_ms(100);
  }

  return 0;
}
```

SSCOM V5.13.1 串口/网络数据调试器,作者:大虾丁丁,2618058@qq.com

通讯端口 串口设置 显示 发送 多字符串 小工具 帮助 联系作者 大虾

F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1

数据发送测试成功。

串口 1 打印输出实现

```c
#include<stdio.h>
#pragma import(__use_no_semihosting)
//标准库需要的支持函数
struct __FILE
{
  int handle;
};

FILE __stdout;
//定义_sys_exit()以避免使用半主机模式
void _sys_exit(int x)
{
  x = x;
}
//重定义fputc函数
int fputc(int ch, FILE *f)
{
  USART_SendData(USART1,(uint8_t)ch);
  while (USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET);//必须加入等待串口发送完成
  return ch;
}
```

```c
static void USART_Config(void);

int main(void)
{
  USART_Config();

  printf("XXXZZZZZZ\r\n");
  while(1)
  {
    printf("11111111\r\n");
    delay_ms(100);
  }

  return 0;
}
```

```
XXXZZZZZZ
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
```

实现完成

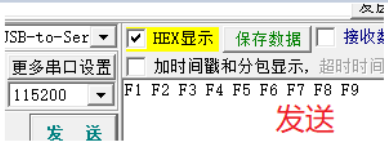
串口 1 接收中断实现

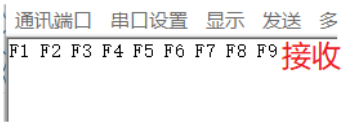
```c
static void USART_Config(void)
{
  GPIO_InitTypeDef  GPIO_InitStructure;
  USART_InitTypeDef USART_InitStructure;
  NVIC_InitTypeDef NVIC_InitStructure;

  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA,ENABLE); //打开PA口时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE); //启动串口1时钟

  /* 配置PA10 RX, PA9 TX */
  GPIO_InitStructure.GPIO_Pin =   GPIO_Pin_9;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF; //串口是复用模式
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP; //推挽输出
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  /* 配置PA10 RX, */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10 ;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF; //串口是复用模式
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
```

```
  GPIO_PinAFConfig(GPIOA,GPIO_PinSource9,GPIO_AF_1); //PA9串口1复用
  GPIO_PinAFConfig(GPIOA,GPIO_PinSource10,GPIO_AF_1);//PA10串口1复用

  /* USARTx configured as follow:
  - BaudRate = 115200 baud
  - Word Length = 8 Bits
  - Stop Bit = 1 Stop Bit
  - Parity = No Parity
  - Hardware flow control disabled (RTS and CTS signals)
  - Receive and transmit enabled
  */
  USART_InitStructure.USART_BaudRate = 115200;
  USART_InitStructure.USART_WordLength = USART_WordLength_8b;
  USART_InitStructure.USART_StopBits = USART_StopBits_1;
  USART_InitStructure.USART_Parity = USART_Parity_No;
  USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode =  USART_Mode_Tx | USART_Mode_Rx;
  USART_Init(USART1,&USART_InitStructure);


  NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPriority = 0; //中断优先级
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);

  USART_ITConfig(USART1, USART_IT_RXNE, ENABLE); // 接收中断
  USART_Cmd(USART1,ENABLE);//启用串口1
}
```

```
void USART1_IRQHandler(void)
{
  uint8_t Rx_Data;

  if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
  {
    Rx_Data = (uint8_t)USART_ReceiveData(USART1);
    USART_SendData(USART1,(uint8_t)Rx_Data);
    while (USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET);//必须加入等待串口发送完成
    //STM32F030中断接收之后，不需要写代码清除中断标准位
  }
}
```

```
int main(void)
{
  USART_Config();

  printf("XXXZZZZZZ\r\n");
  while(1)
  {

    delay_ms(100);
  }

  return 0;
}
```





串口 1 DMA 空闲中断接收数据

我是STM32F030F4
可以使用DMA

Table 26. DMA requests for each channel on STM32F030x4/x6/x8 and STM32F070x6/xB devices

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
ADC	ADC(1)	ADC(2)	-	-	-
SPI	-	SPI1_RX	SPI1_TX	SPI2_RX	SPI2_TX
USART	-	USART1_TX(1) USART3_TX(2)	USART1_RX(1) USART3_RX(2)	USART1_TX(2) USART2_TX	USART1_RX(2) USART2_RX
I2C	-	I2C1_TX	I2C1_RX	I2C2_TX	I2C2_RX

主要使用通道3实现串口1 接收数据

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4
ADC	ADC(1)	ADC(2)	-	-
SPI	-	SPI1_RX	SPI1_TX	SPI2_RX
USART	-	USART1_TX(1) USART3_TX(2)	USART1_RX(1) USART3_RX(2)	USART1_TX USART2_T
I2C	-	I2C1_TX	I2C1_RX	I2C2_TX

```
#define Temp_BufSize_LEN 512
uint8_t DMA_Rx[Temp_BufSize_LEN] = {0};

void DMA1_Config(void)
{
    DMA_InitTypeDef DMA_InitStructure;
    NVIC_InitTypeDef    NVIC_InitStructure;
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1,ENABLE);

    /*****DMA接收配置 Peripheral->Memory*****************/
    DMA_DeInit(DMA1_Channel3);
    DMA_InitStructure.DMA_BufferSize = Temp_BufSize_LEN;
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
    DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;  //外设到源
    DMA_InitStructure.DMA_Priority = DMA_Priority_High;
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable; //内存递增
    DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&USART1->RDR;  //外设串口1
    DMA_InitStructure.DMA_PeripheralInc =  DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
    DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)DMA_Rx;
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
    DMA_Init(DMA1_Channel3,&DMA_InitStructure);

    DMA_ClearITPendingBit(DMA1_IT_TC3); //DMA1_IT_TC3 传输通道3传输完成标志，串口1 RX在DMA channel3
    DMA_ITConfig(DMA1_Channel3,DMA_IT_TC,ENABLE); //配置DMA通道3中断
    USART_DMACmd(USART1,USART_DMAReq_Rx,ENABLE); //DMA接收使能
    DMA_Cmd(DMA1_Channel3,ENABLE);  //使能DMA通道3
    NVIC_InitStructure.NVIC_IRQChannel = DMA1_Channel2_3_IRQn; //DMA中断通道
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_InitStructure.NVIC_IRQChannelPriority = 1; //DMA中断优先级
    NVIC_Init(&NVIC_InitStructure);

}
void USART1_IRQHandler(void)
{
  uint32_t RXLEN = 0;

  //OverRun Error处理
   if(USART_GetFlagStatus(USART1, USART_FLAG_ORE) != RESET)
    {
      USART_ReceiveData(USART1);
      USART_ClearFlag(USART1, USART_FLAG_ORE);
    }
    //接收空闲中断处理
  if(USART_GetITStatus(USART1, USART_IT_IDLE) != RESET)
  {
    USART_ClearITPendingBit(USART1, USART_IT_IDLE); //清除中断标志位
    DMA_Cmd(DMA1_Channel3, DISABLE);
    RXLEN= Temp_BufSize_LEN - DMA_GetCurrDataCounter(DMA1_Channel3);//获得传输的数据个数
    DMA_SetCurrDataCounter(DMA1_Channel3, Temp_BufSize_LEN);
    DMA_Cmd(DMA1_Channel3, ENABLE);
    printf("DMA IRQ..\r\n");

    for(int i = 0; i < RXLEN; i++)
    {
      printf(" %x ", DMA_Rx[i]); //接收数据展示，项目开发要取消掉打印
    }
    printf("\r\n");
  }
}
```

```c
static void USART_Config(void)
{
  GPIO_InitTypeDef  GPIO_InitStructure;
  USART_InitTypeDef USART_InitStructure;
  NVIC_InitTypeDef NVIC_InitStructure;

  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA,ENABLE); //打开PA口时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE); //启动串口1时钟

  /* 配置PA10 RX, PA9 TX */
  GPIO_InitStructure.GPIO_Pin =   GPIO_Pin_9;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF; //串口是复用模式
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP; //推挽输出
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  /* 配置PA10 RX, */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10 ;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF; //串口是复用模式
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOA, &GPIO_InitStructure);


  GPIO_PinAFConfig(GPIOA,GPIO_PinSource9,GPIO_AF_1); //PA9串口1复用
  GPIO_PinAFConfig(GPIOA,GPIO_PinSource10,GPIO_AF_1);//PA10串口1复用
  USART_InitStructure.USART_BaudRate = 115200;
  USART_InitStructure.USART_WordLength = USART_WordLength_8b;
  USART_InitStructure.USART_StopBits = USART_StopBits_1;
  USART_InitStructure.USART_Parity = USART_Parity_No;
  USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode =  USART_Mode_Tx | USART_Mode_Rx;
  USART_Init(USART1,&USART_InitStructure);


  NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPriority = 2; //串口中断优先级
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);

//  USART_ITConfig(USART1, USART_IT_RXNE, ENABLE); // 接收中断
  USART_ITConfig(USART1, USART_IT_IDLE, ENABLE);//使能串口空闲中断
  USART_ClearITPendingBit(USART1, USART_IT_IDLE);//清除串口空闲中断标志位
  USART_Cmd(USART1,ENABLE);//启用串口1
  DMA1_Config(); //DMA配置
}
```

使用DMA空闲中断，就要取消掉串口普通中断

```c
static void USART_Config(void);
void DMA1_Config(void);

int main(void)
{
  USART_Config();

  printf("XXXZZZZZZ\r\n");
  while(1)
  {

    delay_ms(100);
  }

  return 0;
}
```

发送数据

```
XDMA IRQ..

XXZZZZZZ
DMA IRQ..
 f1  f2  f3  f4  f5  f6  f7  f8  f9
DMA IRQ..
 f1  f2  f3  f4  f5  f6  f7  f8  f9
DMA IRQ..
 f1  f2  f3  f4  f5  f6  f7  f8  f9
DMA IRQ..
 f1  f2  f3  f4  f5  f6  f7  f8  f9
DMA IRQ..
 f1  f2  f3  f4  f5  f6  f7  f8  f9
```

开机的时候莫名其妙执行了一次空闲中断

空闲中断接收数据成功

记得滤除开机莫名奇妙的空闲中断

定时器 TIME3 使用

```c
void TIM3_Init(void)//0.1ms
{

  TIM_TimeBaseInitTypeDef          TIM_TimeBaseInitStructure;
  NVIC_InitTypeDef                NVIC_InitStructure;

  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);  //tim3时钟使能,APB时钟48M

  TIM_TimeBaseInitStructure.TIM_ClockDivision = TIM_CKD_DIV1; //分频系数为1
  TIM_TimeBaseInitStructure.TIM_CounterMode = TIM_CounterMode_Up; //向上计数
  TIM_TimeBaseInitStructure.TIM_Period = 100 - 1;
  TIM_TimeBaseInitStructure.TIM_Prescaler = 48 - 1; //定时100us
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseInitStructure);
  TIM_SelectOutputTrigger(TIM3, TIM_TRGOSource_Update);

  TIM_ClearITPendingBit(TIM3, TIM_IT_Update);
  TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE); //允许定时器3更新中断

  NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn; //定时器3中断
  NVIC_InitStructure.NVIC_IRQChannelPriority = 0; //优先级0
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
  TIM_Cmd(TIM3, ENABLE); //使能定时器3
}
```

```c
void TIM3_IRQHandler(void)//100us
{
  if(TIM_GetITStatus(TIM3, TIM_IT_Update) != RESET) //溢出中断
  {
    printf("Timer3...\r\n");

    TIM_ClearITPendingBit(TIM3, TIM_IT_Update); //清除中断标志位
  }
}
```

```c
static void USART_Config(void);
void TIM3_Init(void);

int main(void)
{
  USART_Config();

  TIM3_Init();
  printf("XXXZZZZZZ\r\n");
  while(1)
  {

    delay_ms(100);
  }

  return 0;
}
```

```
Timer3...
Timer3...
Timer3...
Timer3...
Timer3...
Timer3...
Timer3...
Timer3...
Timer3...
Timer3...
Timer3...
Timer3...
Timer3...
Timer3...
Timer3...
Timer3...
```