青风带你学 stm32f030 系列教程

---------- 库函数操作版本

出品论坛： www.qfv8.com 青风电子社区

作者： 青风

QQ： 157736409

淘宝店：http://qfv5.taobao.com

邮箱： wanqin_002@126.com

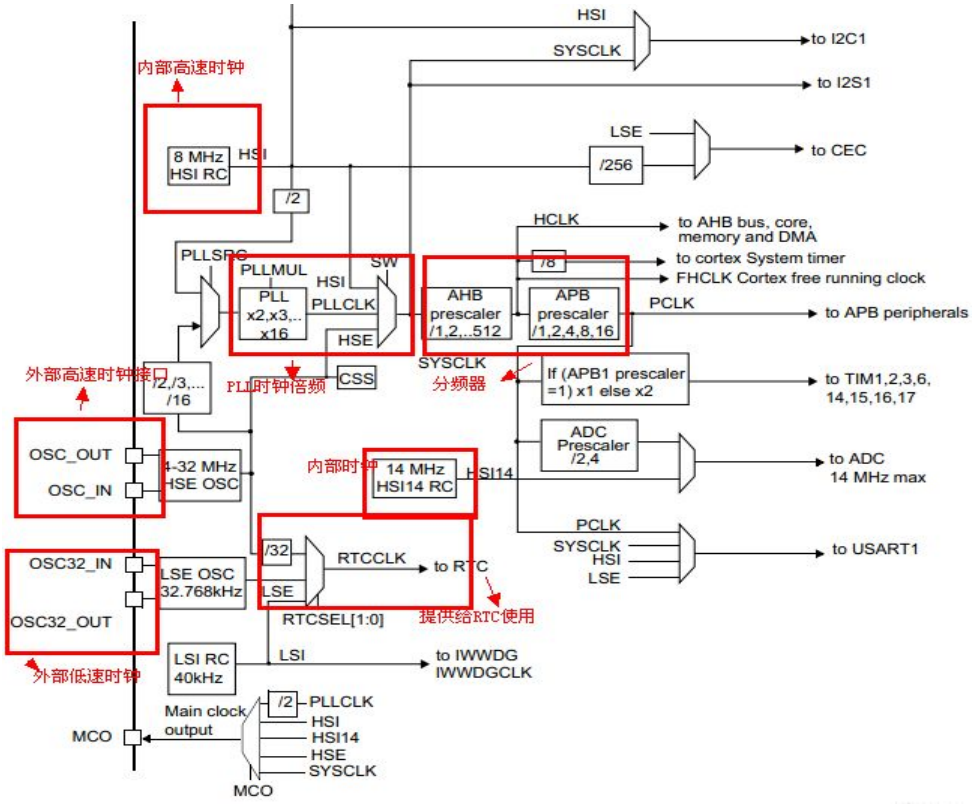
硬件平台：QF-STM32F030 开发板

2.2 系统时钟设置

2.2.1 原理分析：

系统时钟的设置在运行 MCU 时是十分关键的问题，你需要知道你的 CPU 跑在什么样的速度，使用什么样的时钟，如何设置。这些问题我们都在这一节一一讲述，并且同时将会通过硬件参数的系统滴答时钟 SysTick 来进行精确定时。

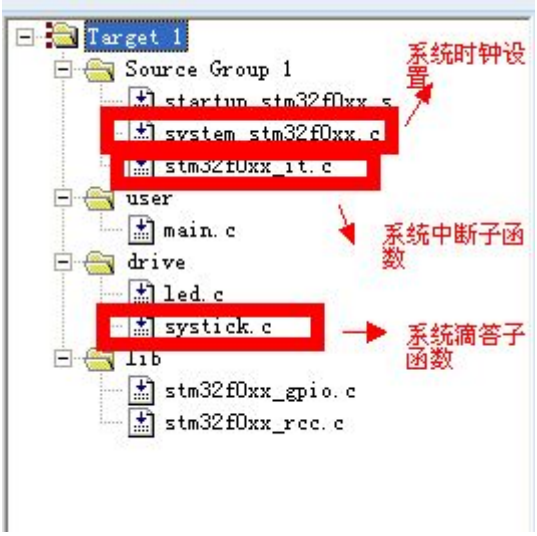
Stm32f030 系列属于 cortex m0 系列内核，时钟速率最快可以跑到接近 50MHZ 左右，那么其内核时钟如何产生的了？我们首先看看下面的时钟结构图：



图中的我们可以看到，M0 的时钟实际上是有多种选择途径的，你采用外部高速晶振时钟可以，也可以采用内部的高速时钟，最终都要通过一个 PLL 进行倍频的，倍频之后再分频提供给 AHB 总线，AHB 总线上挂了系统 CPU，AHB 分频后的时钟就是 CPU 的工作时钟了。之后还要通过 APB 时钟分频提供给其他的外设使用。上面所分析的就是 M0 内部时钟是如何产生的。知道这些了，我们就来通过 KEIL 编译环境对这个工作时钟状态进行设置。

2.2.2 软件准备：

我们首先结合软件程序进行讲解。首先看看通过 keil 建立工程项目，如下图所示：Keil 的工程树建立如下图所示：lib 中加入 gpio 和 rcc 两个库文件，分别用于设置 GPIO 和 rcc 时钟。用户需要编写 systick.c 文件用于设置时钟滴答。



如上图所示，其中多系统时钟，打开 stm32f0xx.h 文件，在 1928 行我们看见下面一个定义：

```
/******************  Bit definition for RCC_CR register  ******************/
#define  RCC_CR_HSION                        ((uint32_t)0x00000001)        /*!< Internal High Speed clock enable */
#define  RCC_CR_HSIRDY                       ((uint32_t)0x00000002)        /*!< Internal High Speed clock ready flag */
#define  RCC_CR_HSITRIM                      ((uint32_t)0x000000F8)        /*!< Internal High Speed clock trimming */
#define  RCC_CR_HSICAL                       ((uint32_t)0x0000FF00)        /*!< Internal High Speed clock Calibration */
#define  RCC_CR_HSEON                        ((uint32_t)0x00010000)        /*!< External High Speed clock enable */
#define  RCC_CR_HSERDY                       ((uint32_t)0x00020000)        /*!< External High Speed clock ready flag */
#define  RCC_CR_HSEBYP                       ((uint32_t)0x00040000)        /*!< External High Speed clock Bypass */
#define  RCC_CR_CSSON                        ((uint32_t)0x00080000)        /*!< Clock Security System enable */
#define  RCC_CR_PLLON                        ((uint32_t)0x01000000)        /*!< PLL enable */
#define  RCC_CR_PLLRDY                       ((uint32_t)0x02000000)        /*!< PLL clock ready flag */
```

红色方框标的就是外部高速时钟的定义，设置 RCC_CR 寄存器，在系统启动时，在 system_stm32f0xx.c 文件内定义好了系统时钟的配置函数，那么在启动项首先直接指向 SystemInit，启动项文件我们在 startup_stm32f0xx.s 中可以找到如下代码：

```
01.    Reset_Handler    PROC
02.                     EXPORT  Reset_Handler                 [WEAK]
03.            IMPORT  __main
04.            IMPORT  SystemInit
05.                     LDR     R0, =SystemInit//启动项首先进入系统初始化
06.                     BLX     R0
07.                     LDR     R0, =__main//之后才去跑主函数
08.                     BX      R0
```

```
09.                     ENDP
10.
```

我们点击查看 SystemInit 的原函数定义，这个函数在在 **system_stm32f0xx.c** 文件中进行了设置：

```
11. void SystemInit (void)
12. {
13.   /* Set HSION bit */
14.   RCC->CR |= (uint32_t)0x00000001;
15.
16. #if defined (STM32F0XX_MD) || defined (STM32F030X8)
17.   /* Reset SW[1:0], HPRE[3:0], PPRE[2:0], ADCPRE and MCOSEL[2:0] bits */
18.   RCC->CFGR &= (uint32_t)0xF8FFB80C;
19. #else
20.   /* Reset SW[1:0], HPRE[3:0], PPRE[2:0], ADCPRE, MCOSEL[2:0], MCOPRE[2:0] and
    PLLNODIV bits */
21.   RCC->CFGR &= (uint32_t)0x08FFB80C;
22. #endif /* STM32F0XX_MD or STM32F030X8 */
```

这里注意一点就是，在函数里区分了 **stm32f030x8** 没有设置 **PLLNODIV** 这个寄存器位,如参考手册说明：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLL NODIV	MCOPRE[2:0]			MCO[3:0]				Res.	Res.	PLLMUL[3:0]				PLL XTPRE	PLLS RC
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ADCP RE	Res.	Res.	Res.	PPRE[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
	rw				rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

Bit 31 **PLLNODIV:** PLL clock not divided for MCO (not available on STM32F030x8 devices)
This bit is set and cleared by software. It switches off divider by 2 for PLL connection to MCO.
0: PLL is divided by 2 for MCO
1: PLL is not divided for MCO

并且在这个函数末尾定义了 **SetSysClock()函数，这个 SetSysClock()函数里面正好使用了：**

/* Enable HSE */

RCC->CR |= ((uint32_t)RCC_CR_HSEON);

这句话使能了 **HSE。HSI 称为高速内部振荡时钟，HSE 称为外部振荡时钟。同时在文件系统中定义系统内核时钟为 48MHZ。根据初始化**

```
23.  RCC->CFGR &= (uint32_t)0xF8FFB80C;
```

```
/* Get SYSCLK source -------------------------------------------------------*/
tmp = RCC->CFGR & RCC_CFGR_SWS;

switch (tmp)
{
  case 0x00:  /* HSI used as system clock */
    SystemCoreClock = HSI_VALUE;
    break;
  case 0x04:  /* HSE used as system clock */
    SystemCoreClock = HSE_VALUE;
    break;
```

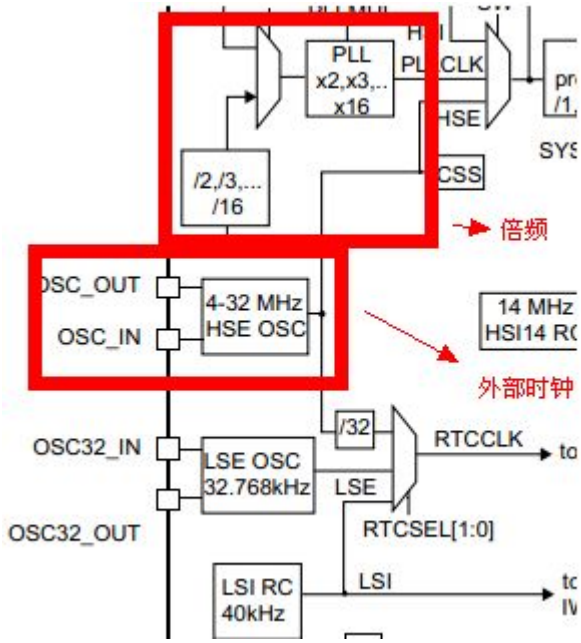
根据上面的代码，可以更新你所选择的时钟源。

```
24. uint32_t SystemCoreClock       = 48000000;
```

下面一段代码定义了 **HSE 通过 PLL 配置为 48MHZ：**

```
25.  /* PLL configuration = HSE * 6 = 48 MHz */
26.       RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_PLLSRC |
    RCC_CFGR_PLLXTPRE | RCC_CFGR_PLLMULL));
27.       RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLSRC_PREDIV1 |
    RCC_CFGR_PLLXTPRE_PREDIV1 | RCC_CFGR_PLLMULL6);
28.
```

系统外部时钟模型如下：



startup_stm32f0xx.s 是建立 stm32f030 keil 工程项目后 MDK 自动产生的。可以看到系统启动后首先确定系统时钟，然后才进入住函数。系统时钟配置好后，我们在来通过 SysTick 来进行精确定时，来驱动 LED 进行闪烁，这里面设置 ms 级的配置，然后设置 10s 闪烁，大家下载程序后可以对照时钟看是不是 10s 进行闪烁。

首先建立一个延迟函数，nTime 作为定时数。SysTick 配置通过设置为 1ms,为 SystemCoreClock / 1000：

```
29. void Delay_ms(__IO uint32_t nTime)//延迟函数，设置为 MS
30. {
31.    TimingDelay = nTime;//时钟滴答数
```

```
32.
33.     while(TimingDelay != 0);
34. }
35.
36.
37. void Systick_Init(void)
38. {
39.     if (SysTick_Config(SystemCoreClock / 1000))//设置为 1 毫秒
40.     {
41.       /* Capture error */
42.       while (1);
43.     }
44. }
```

并且通过 TimingDelay_Decrement 调动中断，代码如下：

```
45. void TimingDelay_Decrement(void)
46. {
47.     if (TimingDelay != 0x00)
48.     {
49.       TimingDelay--;
50.     }
51. }
```

在 stm32f0xx_it.c 中定义 SysTick_Handler 中断函数，函数调用 TimingDelay_Decrement：

```
52. void SysTick_Handler(void)
53. {
54.             TimingDelay_Decrement();//调动中断函数
55. }
```

最后我们调用上面编写的 systick.c 子函数，编写主函数如下：

```
56. #include "stm32f0xx.h"
57. #include "systick.h"
58. #include "led.h"
59.
60. int main(void)
61. {
62.     SystemInit();//系统初始化
63.     LED_Init();//led 灯初始化
64.     Systick_Init();//时钟滴答初始化
65.     while(1)
66.     {
67.       LED_Open();
68.       Delay_ms(10000); //延时 1s
69.       LED_Close();
70.       Delay_ms(10000);
71.       }
72. }
```

实验现象：

Led 灯按照 1s 等间隔的闪烁，精确定时。