

Sugar的专栏

如果只得一星期



- ☰ 目录视图
- ☰ 摘要视图
- RSS 订阅

💡 博客专家福利

2015年4月微软MPP申请

10月推荐文章汇总

有奖征文-我亲历的京东发展史

参与迷你编程马拉松赢iPhone 6

原 第一个函数SystemInit()里面有些啥

🗨 技术问答

分类: Cortex-M3

2012-12-06 16:26

👁 3132人阅读

💬 评论(0)

🔖 收藏

✍ 快速回复

SystemInit()这个函数出现在main()函数的第一行，可以看出它的重要性。以前关于SystemInit()这个函数从来没有关心过，只知道这是进行STM32系统初始化的一个函数。今天决定仔细看看，重新开始STM32的学习。这个函数在system_stm32f10x.c中，此C文件主要就是干具体硬件配置相关的工作。

[cpp] view plain copy print ?

```
01.  /** @addtogroup STM32F10x_System_Private_Functions
02.      * @{
03.      */
04.
05.  /**
06.      * @brief Setup the microcontroller system
07.      * Initialize the Embedded Flash Interface, the PLL and update the
08.      * SystemCoreClock variable.
09.      * @note This function should be used only after reset.
10.      * @param None
11.      * @retval None
12.      */
13.  void SystemInit (void)
14.  {
15.      /* Reset the RCC clock configuration to the default reset state(for debug purpose) */
16.      /* Set HSION bit */
17.      RCC->CR |= (uint32_t)0x00000001;
18.
19.      /* Reset SW, HPRE, PPRE1, PPRE2, ADCPRE and MCO bits */
20.  #ifndef STM32F10X_CL
21.      RCC->CFGR &= (uint32_t)0xF8FF0000;
22.  #else
23.      RCC->CFGR &= (uint32_t)0xF0FF0000;
24.  #endif /* STM32F10X_CL */
25.
26.      /* Reset HSEON, CSSON and PLLON bits */
27.      RCC->CR &= (uint32_t)0xFEFFFFFF;
28.
29.      /* Reset HSEBYP bit */
30.      RCC->CR &= (uint32_t)0xFFBFFFFFF;
31.
32.      /* Reset PLLSRC, PLLXTPRE, PLLMUL and USBPRE/OTGFSPRE bits */
33.      RCC->CFGR &= (uint32_t)0xFF80FFFF;
34.
35.  #ifndef STM32F10X_CL
36.      /* Reset PLL2ON and PLL3ON bits */
37.      RCC->CR &= (uint32_t)0xEBFFFFFF;
38.
39.      /* Disable all interrupts and clear pending bits */
40.      RCC->CIR = 0x00FF0000;
41.
42.      /* Reset CFGR2 register */
43.      RCC->CFGR2 = 0x00000000;
44.  #elif defined (STM32F10X_LD_VL) || defined (STM32F10X_MD_VL) || (defined STM32F10X_HD_VL)
45.      /* Disable all interrupts and clear pending bits */
46.      RCC->CIR = 0x009F0000;
47.
48.      /* Reset CFGR2 register */
49.      RCC->CFGR2 = 0x00000000;
50.  #else
51.      /* Disable all interrupts and clear pending bits */
52.      RCC->CIR = 0x009F0000;
53.  #endif /* STM32F10X_CL */
54.
55.  #if defined (STM32F10X_HD) || (defined STM32F10X_XL) || (defined STM32F10X_HD_VL)
56.      #ifndef DATA_IN_ExtSRAM
```

个人资料



Suagr

- + 加关注
- 📧 发私信



访问: 94026次

积分: 2221

等级: BLOG > 5

排名: 第6566名

原创: 125篇 转载: 3篇

译文: 8篇 评论: 47条

文章搜索

🔍

博客专栏



python学习——
Learn Beautiful
Soup

文章: 9篇
阅读: 1495

文章分类

- ARM7 (17)
- 嵌入式积累 (11)
- 电脑鼠征程 (1)
- Cortex-M3 (6)
- Cortex-M0 (3)
- 51单片机 (3)
- TCP/IP (2)
- MSP430 (3)
- Live (2)
- Python (17)
- ucOS (2)
- Linux (29)
- eLua (10)
- 毕业设计之路 (9)
- CC2530/Zigbee (5)
- CodeCombat (12)
- 30天自制操作系统 (0)
- 射雕ZERO (0)

文章存档

- 2014年10月 (9)
- 2014年07月 (1)
- 2014年05月 (1)
- 2014年03月 (12)
- 2013年06月 (1)


```
02.     #ifdef DATA_IN_ExtSRAM
03.         SystemInit_ExtMemCtl();
04.     #endif /* DATA_IN_ExtSRAM */
05. #endif
```

这段跟设置外部**RAM**有关吧，我用到的STM32F103RBT与此无关。

```
[cpp] view plain copy print ?
01. SetSysClock();
```

这又是一个函数，主要是配置系统时钟频率。HCLK,PCLK2,PCLK1的分频值，分别代表AHB,APB2,和APB1。当然还干了其它的事情，配置**FLASH**延时周期和使能预取缓冲期。后面的这个配置具体还不了解。

```
[cpp] view plain copy print ?
01. #ifdef VECT_TAB_SRAM
02.     SCB->VTOR = SRAM_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM. */
03. #else
04.     SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal FLASH. */
05. #endif
```

这段代码主要是实现**向量表的重定位**。依据你想要将向量表定位在内部SRAM中还是内部FLASH中。这个SCB开始没在STM32参考手册中发现，原来它是跟Cortex-M3内核相关的东西。所以ST公司就没有把它包含进来吧。内核的东西后面再了解，这里给自己提个醒。

然后再看看SystemInit()中的那个函数SetClock()又做了什么吧。

```
[cpp] view plain copy print ?
01. static void SetSysClock(void)
02. {
03.     #ifdef SYSCLOCK_FREQ_HSE
04.         SetSysClockToHSE();
05.     #elif defined SYSCLOCK_FREQ_24MHz
06.         SetSysClockTo24();
07.     #elif defined SYSCLOCK_FREQ_36MHz
08.         SetSysClockTo36();
09.     #elif defined SYSCLOCK_FREQ_48MHz
10.         SetSysClockTo48();
11.     #elif defined SYSCLOCK_FREQ_56MHz
12.         SetSysClockTo56();
13.     #elif defined SYSCLOCK_FREQ_72MHz
14.         SetSysClockTo72();
15.     #endif
16.
17.     /* If none of the define above is enabled, the HSI is used as System clock
18.        source (default after reset) */
19. }
```

从中可以看出就是根据不同的宏设置不同的系统时钟，这些宏就在跟此函数在同一个源文件里。官方很是考虑周到，我们只需要选择相应宏就能达到快速配置系统时钟的目的。

```
[cpp] view plain copy print ?
01. #if defined (STM32F10X_LD_VL) || (defined STM32F10X_MD_VL) || (defined STM32F10X_HD_VL)
02.     /* #define SYSCLOCK_FREQ_HSE      HSE_VALUE */
03.     #define SYSCLOCK_FREQ_24MHz      24000000
04. #else
05.     /* #define SYSCLOCK_FREQ_HSE      HSE_VALUE */
06.     /* #define SYSCLOCK_FREQ_24MHz    24000000 */
07.     /* #define SYSCLOCK_FREQ_36MHz    36000000 */
08.     /* #define SYSCLOCK_FREQ_48MHz    48000000 */
09.     /* #define SYSCLOCK_FREQ_56MHz    56000000 */
10.     #define SYSCLOCK_FREQ_72MHz      72000000
11. #endif
```

比如这里我需要配置系统时钟为**72MHZ**，则只需要将**#define SYSCLOCK_FREQ_72MHz 72000000**两边的注释符去掉。

这个函数里面又有SetSysClockTo72()函数，这个函数就是具体操作寄存器进行配置了。

```
[cpp] view plain copy print ?
01. #elif defined SYSCLOCK_FREQ_72MHz
```

```

02.  /**
03.   * @brief Sets System clock frequency to 72MHz and configure HCLK, PCLK2
04.   *       and PCLK1 prescalers.
05.   * @note This function should be used only after reset.
06.   * @param None
07.   * @retval None
08.   */
09. static void SetSysClockTo72(void)
10. {
11.     __IO uint32_t StartUpCounter = 0, HSEStatus = 0;
12.
13.     /* SYSClk, HCLK, PCLK2 and PCLK1 configuration -----*/
14.     /* Enable HSE */
15.     RCC->CR |= (uint32_t)RCC_CR_HSEON;
16.
17.     /* Wait till HSE is ready and if Time out is reached exit */
18.     do
19.     {
20.         HSEStatus = RCC->CR & RCC_CR_HSERDY;
21.         StartUpCounter++;
22.     } while((HSEStatus == 0) && (StartUpCounter != HSE_STARTUP_TIMEOUT));
23.
24.     if ((RCC->CR & RCC_CR_HSERDY) != RESET)
25.     {
26.         HSEStatus = (uint32_t)0x01;
27.     }
28.     else
29.     {
30.         HSEStatus = (uint32_t)0x00;
31.     }
32.
33.     if (HSEStatus == (uint32_t)0x01)
34.     {
35.         /* Enable Prefetch Buffer */
36.         FLASH->ACR |= FLASH_ACR_PRFTBE;
37.
38.         /* Flash 2 wait state */
39.         FLASH->ACR &= (uint32_t)((uint32_t)~FLASH_ACR_LATENCY);
40.         FLASH->ACR |= (uint32_t)FLASH_ACR_LATENCY_2;
41.
42.
43.         /* HCLK = SYSCLK */
44.         RCC->CFGR |= (uint32_t)RCC_CFGR_HPRE_DIV1;
45.
46.         /* PCLK2 = HCLK */
47.         RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE2_DIV1;
48.
49.         /* PCLK1 = HCLK */
50.         RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE1_DIV2;
51.
52. #ifdef STM32F10X_CL
53.         /* Configure PLLs -----*/
54.         /* PLL2 configuration: PLL2CLK = (HSE / 5) * 8 = 40 MHz */
55.         /* PREDIV1 configuration: PREDIV1CLK = PLL2 / 5 = 8 MHz */
56.
57.         RCC->CFGR2 &= (uint32_t)~(RCC_CFGR2_PREDIV2 | RCC_CFGR2_PLL2MUL |
58.                                RCC_CFGR2_PREDIV1 | RCC_CFGR2_PREDIV1SRC);
59.         RCC->CFGR2 |= (uint32_t)(RCC_CFGR2_PREDIV2_DIV5 | RCC_CFGR2_PLL2MUL8 |
60.                                RCC_CFGR2_PREDIV1SRC_PLL2 | RCC_CFGR2_PREDIV1_DIV5);
61.
62.         /* Enable PLL2 */
63.         RCC->CR |= RCC_CR_PLL2ON;
64.         /* Wait till PLL2 is ready */
65.         while((RCC->CR & RCC_CR_PLL2RDY) == 0)
66.         {
67.         }
68.
69.
70.         /* PLL configuration: PLLCLK = PREDIV1 * 9 = 72 MHz */
71.         RCC->CFGR &= (uint32_t)~(RCC_CFGR_PLLXTPRE | RCC_CFGR_PLLSRC | RCC_CFGR_PLLMULL);
72.         RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLXTPRE_PREDIV1 | RCC_CFGR_PLLSRC_PREDIV1 |
73.                                RCC_CFGR_PLLMULL9);
74.     #else
75.         /* PLL configuration: PLLCLK = HSE * 9 = 72 MHz */
76.         RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_PLLSRC | RCC_CFGR_PLLXTPRE |
77.                                RCC_CFGR_PLLMULL));
78.         RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLSRC_HSE | RCC_CFGR_PLLMULL9);
79. #endif /* STM32F10X_CL */
80.
81.         /* Enable PLL */
82.         RCC->CR |= RCC_CR_PLLON;
83.
84.         /* Wait till PLL is ready */
85.         while((RCC->CR & RCC_CR_PLLRDY) == 0)
86.         {
87.         }
88.
89.         /* Select PLL as system clock source */
90.         RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_SW));
91.         RCC->CFGR |= (uint32_t)RCC_CFGR_SW_PLL;
92.

```

```
93.      /* Wait till PLL is used as system clock source */
94.      while ((RCC->CFGR & (uint32_t)RCC_CFGR_SW5) != (uint32_t)0x08)
95.      {
96.      }
97.  }
98.  else
99.  { /* If HSE fails to start-up, the application will have wrong clock
100.     configuration. User can add here some code to deal with this error */
101.  }
102. }
103. #endif
```

上面的代码需要细细看。SystemInit()差不多就是这样了。



- 上一篇

感慨下STM32的学习状况
- 下一篇

从第二个函数NVIC_PriorityGroupConfig()中了解Cortex-M3的中断

主题推荐

- application
- interface
- function
- buffer
- 工作

猜你在找

- uCOSII移植到友善之臂mini2440

stm32之keil开发环境搭建

Ucos II 移植之二

死循环在BEAB BKPT 0xAB汇编的解决办法
- 电阻触摸屏的校准算法

对通用输入输出GPIO的深入理解

I2S音频总线学习—数字音频技术

一起学mini2440裸机开发十一—mini2440外部中断实验

查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

- 全部主题
- Hadoop
- AWS
- 移动游戏
- Java
- Android
- iOS
- Swift
- 智能硬件
- Docker
- OpenStack
- VPN
- Spark
- ERP
- IE10
- Eclipse
- CRM
- JavaScript
- 数据库
- Ubuntu
- NFC
- WAP
- jQuery
- BI
- HTML5
- Spring
- Apache
- .NET
- API
- HTML
- SDK
- IIS
- Fedora
- XML
- LBS
- Unity
- Splashtop
- UML
- components
- Windows Mobile
- Rails
- QEMU
- KDE
- Cassandra
- CloudStack
- FTC
- coremail
- OPhone
- CouchBase
- 云计算
- iOS6
- Rackspace
- Web App
- SpringSide
- Maemo
- Compuware
- 大数据
- aptech
- Perl
- Tornado
- Ruby
- Hibernate
- ThinkPHP
- HBase
- Pure
- Solr
- Angular
- Cloud Foundry
- Redis
- Scala
- Django
- Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服
 杂志客服
 微博客服
 webmaster@csdn.net
 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持

京 ICP 证 070598 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved

