STM32初学者笔记（1） 一步步建立自己的STM32函数

STM32初学者笔记（1） 一步步建立自己的STM32函数
使用自己建立的 STM32F103.H 的 头文件。里面有大量中文注释。非常适合初学者。

STM32的库实在太庞大负责，对初学者来说实在是件头痛的事情！

下面是万利开发板 LCD DEMO 例子中的函数，改后初学者可以对照学习一下。

```
//======================================================================
//      原函数名称：  RCC_Configuration(void)
//======================================================================
void STM32_RCC_Configuration(void)
{

    STM32_Rcc_Regs->cr.bit.HSEON=0;         // HSE振荡器开启关闭
    STM32_Rcc_Regs->cr.bit.HSEBYP=0;    // 外部高速时钟没有旁路
    STM32_Rcc_Regs->cr.bit.HSERDY=0;    // 外部高速时钟就绪标志清零

    STM32_Rcc_Regs->cr.bit.HSEON=1;         // 外部高速时钟使能
1：HSE振荡器开启


     while( !(STM32_Rcc_Regs->cr.bit.HSERDY )
);//由硬件置1来指示外部时钟已经稳定。

     /* HCLK = SYSCLK＝72MHZ  :HLCK 提供给CPU, 内存和DMA 最大72MHZ  */
     STM32_Rcc_Regs->crfgr.bit.HPRE=RCC_AHB_SYSCLK_DIV1_B;    //AHB预分频

    /* PCLK2 = HCLK/1=SYSCLK=72   PCLK2 提供给 APB2外设，最大72MHZ */
     STM32_Rcc_Regs->crfgr.bit.PPRE2=RCC_APB2_HCLK_DIV1_B;
//高速APB预分频（APB2）

     /* PCLK1 = HCLK/2=SYSCLK/2=36   PCLK1提供给APB1外设，最大36MHZ */
     STM32_Rcc_Regs->crfgr.bit.PPRE1=RCC_APB1_HCLK_DIV2_B;
//低速APB预分频（APB1）必须保证APB1时钟频率不超过36MHz


      /* ADCCLK = PCLK2/6     ADC转换速率：72M/6 = 12MHZ  */
     STM32_Rcc_Regs->cfgr.bit.ADCPRE=RCC_ADCPRE_PCLK2_DIV6_B;

     /* Flash 2 wait state */
     STM32_Flash_Regs->ACR.all&=((u32)0x00000038);    //清零某些位
     STM32_Flash_Regs->ACR.bit.LATENCY=2;
    STM32_Flash_Regs->ACR.bit.PRFTBE=1;         //预取缓冲区使能

    /*  PLLCLK = 8MHz * 9 = 72MHZ ) */
     STM32_Rcc_Regs->cfgr.bit.PLLXTPRE=0;         //HSE分频器作为PLL输入
0：HSE不分频
     STM32_Rcc_Regs->cfgr.bit.PLLSRC=1;         //HSE时钟作为PLL输入时钟
     STM32_Rcc_Regs->cfgr.bit.PLLMUL=RCC_PLL_9_B;         //PLL倍频系数9

    /* Enable PLL */
    STM32_Rcc_Regs->cr.bit.PLLON=1;             // PLL使能
    while( !(STM32_Rcc_Regs->cr.bit.PLLRDY ) );    // PLL时钟就绪标志
PLL锁定后由硬件置1

     /* Select PLL as system clock source */
     STM32_Rcc_Regs->cfgr.bit.SW=0;
     STM32_Rcc_Regs->cfgr.bit.SW=2;         //RCC_SW_SYSCLK_PLL
```

```
    /* Wait till PLL is used as system clock source */
     while(STM32_Rcc_Regs->cfgr.bit.SWS!=2);        //RCC_SWS_SYSCLK_PLL

    /* Enable GPIOA, GPIOB, GPIOC, GPIOD, GPIOE and AFIO clocks */
    STM32_Rcc_Regs->apb2enr.all
|=(RCC_AFIOEN|RCC_IOPAEN|RCC_IOPBEN|RCC_IOPCEN|RCC_IOPDEN|RCC_IOPEEN);

    /* TIM2 AND CAN clocks enable */
    STM32_Rcc_Regs->apb1enr.all |=(RCC_TIM2EN| RCC_CANEN);

}
```