Data：2019/1/6

1.想要选择的主时钟使能,等待稳定

2.选择主时钟并进行切换,等待切换完成

3.选择系统时钟分频数

4.选择相应的外设时钟

#include "clock.h"

void ClockInit(u8 clockSel)

{

if(clockSel == HSE\_CLOCK)

{

//切换到外部始终之前,因为我使用的是24M时钟,所以需要修改插入flash等待时间

//此处讲解选项字节再补上

//注意,下面这一段只能在使用16M一下时钟的时候有用

CLK->ECKR |= 0x01;//使能外部晶振

while(!(CLK->ECKR & 0x02));//等待时钟准备就绪

CLK->SWCR &= ~((1<<2)|(1<<3));//时钟切换中断禁止并清除中断标志

CLK->SWR = 0xB4;//时钟选择为HSE

while(CLK->SWCR & 0x01);//等待时钟切换慢的结束

CLK->SWCR |= 0x01;//使能时钟切换

while(CLK->CMSR != 0xB4);//等待系统时钟切换到0xb4

CLK->CKDIVR = 0x00;//设置时钟预分频,设置为fmaster = fclock

CLK->CCOR &= ~(1<<0);//禁止CCO输出

//此时,应当关闭内部始终

CLK->ICKR &= ~0x01;

}

else if(clockSel == HSI\_CLOCK)

{

CLK->ICKR |= 0x01;//使能内部高速时钟

while(!(CLK->ICKR | 0x02));//等待准备就绪

CLK->SWCR &= ~((1<<2)|(1<<3));//时钟切换中断禁止并清除中断标志

CLK->SWR = 0xE1;//时钟选择为HSI

while(CLK->SWCR & 0x01);//等待时钟切换慢的结束

CLK->SWCR |= 0x01;//使能时钟切换

while(CLK->CMSR != 0xE1);//等待系统时钟切换到0xb4

CLK->CKDIVR = 0x00;//设置时钟预分频,设置为fmaster = fclock

CLK->CCOR &= ~(1<<0);//禁止CCO输出

//此时,应当关闭外部时钟

CLK->ECKR &= ~0x01;//使能外部晶振

}

}

/\*定时时间 = (TIM1\_ARRL + 1) \* 10ms \*/

void Timer1\_init(void)

{

CLK\_PCKENR1 |= 0x80; //打开TIM1时钟 = Fmaster = 2MHz

asm("sim"); // 关全局中断

TIM1\_PSCRH = 0x4E;

TIM1\_PSCRL = 0x20; //时钟20000分频，2MHz/20000 = 100Hz = 10ms

TIM1\_ARRH=0; //自动重装载

TIM1\_ARRL=9;

TIM1\_IER |= 0X01; //允许更新中断

TIM1\_EGR |= 0x01; //产生更新事件

TIM1\_CR1 |= 0x01; //开计数器

asm("rim"); // 开全局中断 是不是像51里面的EA = 1；O(∩\_∩)O~

}

好了下面开始编写主函数和中断服务函数了。

int main( void )

{

System\_Init();

Gpio\_Init();

Timer1\_init(); //100ms中断一次

while (1);

}

#pragma vector=TIM1\_OVR\_UIF\_vector //0x0D

\_\_interrupt void TIM1\_OVF\_IRQHandler(void)

{

TIM1\_SR1=0x00; //清除中断标志位 该位由软件清零

PE\_ODR^=0x20; //PE5取反

}

Data：2019/1/7

void Time1\_Init()  
{    
      
  CLK\_PCKENR1 |= SETBIT7;//开TIM1时钟   
  TIM1\_EGR = 0x0;  //允许产生更新事件    
  TIM1\_PSCRH = 0x00;  
  TIM1\_PSCRL = 0xef;//240分频，100kHZ  
  //初始化计数器值,设定自动重载寄存器值，计数125，得到800HZ  
  TIM1\_ARRH = 0x00;   
  TIM1\_ARRL = 0x7c;   
  TIM1\_CNTRH = 0x00;   
  TIM1\_CNTRL = 0x00;  
  //使能计数器,边缘向上计数  
  TIM1\_CR1 = 0x81;  
  //允许更新中断  
  TIM1\_IER = 0x01;   
}  
  
但是我看有的程序里也给计数器如此赋值：  
  TIM1\_CNTRH = 0x00;   
  TIM1\_CNTRL = 0x7c;  
  
按照我对datasheet的理解，向上计数应该是CNTR从0开始计数，到ARR值时产生溢出，再从0开始计数，所以给CNTR赋值应该是0。

例；用TIM1中断定时，1ms进一次中断，LED灯1s翻转一次，向下计数模式

/\* 这个工程模板是别人的，但是这个头文件不影响使用 ，在自己的工程中需更改为自己的头文件 \*/

#include<iostm8s105k6.h>

unsigned int ms\_count=0;//毫秒计数

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

各种函数声明

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void InitLED();

void InitTIM1();

int main(void)

{

CLK\_CKDIVR = 0x00; //CPUDIV = 1 HSIDIV = 1 内部时钟 = 16Mhz

asm("sim"); //先关闭总中断

InitLED(); //各种初始化

InitTIM1();

asm("rim"); //打开总中断

while(1); //进入死循环，等待定时器周期中断

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

定时器1更新中断处理函数

中断向量也可以用十进制代替，这里使用16进制是为了方便而已，

因为头文件的宏定义就是用16进制的，

函数名可以随便写，开心就好

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#pragma vector = 0x0D //这个是中断的格式，直接调用就好，

\_\_interrupt void TIM1\_OVR\_UIF(void)

{

TIM1\_SR1 &= ~(1<<0);//清除中断标志

ms\_count++;

if(ms\_count>1000)//1ms\*1000=1s

{

ms\_count=0;

PE\_ODR ^= 1<<5;//LED灯1s翻转一次

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

定时器1初始化

tim1向下计数模式，

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void InitTIM1()

{

TIM1\_PSCRH=0;//一定要先写高八位

TIM1\_PSCRL=0;//1分频，定时器时钟等于系统时钟=16m

TIM1\_ARRH=0X3e;//一定要先装高八位，再装低八位

TIM1\_ARRL=0X80;//1ms重装值16000，

TIM1\_CNTRH=0;

TIM1\_CNTRL=0;//有必要清除下计数器

TIM1\_IER |= 1<<0;//使能tim1更新中断

TIM1\_SR1 |= 1<<0;//清除tim1更新中断标志

TIM1\_CR1 |= 1<<7;//允许重装，使能定时器

TIM1\_CR1 |= 1<<4;//选择向下计数模式

TIM1\_CR1 |= 1<<0;//使能计数器

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LED初始化

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void InitLED()

{

PE\_DDR |= 1<<5;//PE5方向为输出

PE\_CR1 |= 1<<5;//PE5为推挽输出

PE\_CR2 |= 1<<5;//最大输出速度为10M

PE\_ODR |= 1<<5;//PE5输出1

}

---------------------

作者：深蓝判官

来源：CSDN

原文：https://blog.csdn.net/LHChandsome/article/details/78102720

版权声明：本文为博主原创文章，转载请附上博文链接！