使用 SysTick 精准延时

SAM4N 学习笔记

版本号：1.00

日期：2013/10/25

一、准备工作:

将上一节搭建的 LED 工程复制一份，命名为“2.systick”。这一节主要讲如何使用系统的 SysTick 节拍定时器来进行精准延时程序。

二、程序编写:

SysTick 是 CM3/CM4 内核芯片提供的一个定时器，我们的板子使用的 SAM4N 芯片是 CM4 内核的，自然少不了这个定时器。

首先在 Drivers 文件夹中建立 delay.c 和 delay.h 文件并添加到工程中，在 delay.c 中写入如下代码：

```
#include "delay.h"
uint32_t timeout=0;

/*******************************************************************************
* 函数名：SysTick_Handler()
* 参数  ：void
* 返回值：void
* 描述  ：系统节拍中断服务函数
*******************************************************************************/
void SysTick_Handler(void)
{
    /*定时计数自减 1*/
    timeout--;
}
/*******************************************************************************
* 函数名：systick_hw_init()
* 参数  ：void
* 返回值：void
* 描述  ：系统节拍初始化函数
*******************************************************************************/
void systick_hw_init(void)
{
    //选择系统节拍定时器的时钟源为系统内核时钟
    SysTick->CTRL|=0x01<<2;
    //设置重装值，配置系统节拍定时器为 1ms 中断一次
    /*1ms <-->SystemCoreClock/1000
    * 100us <-->SystemCoreClock/10000
    * 10us <-->SystemCoreClock/100000
    */
    SysTick->LOAD = SystemCoreClock/1000 - 1;

    //使能系统节拍时钟中断
```

```
    SysTick->CTRL|=0x02;

    //使能系统节拍时钟
    SysTick->CTRL|=0x01;
}
/*****************************************************************************
* 函数名：delay_ms()
* 参数  ：uint32_t n  延时的毫秒数
* 返回值：void
* 描述  ：使用系统节拍进行精准延时函数
*****************************************************************************/
void delay_ms(uint32_t n)
{
    timeout = n;                        //设置系统节拍延时
    while(timeout!=0);          //等待延时时间到
}
/*****************************************************************************
* 函数名：delay()
* 参数  ：uint32_t n  延时的循环个数
* 返回值：void
* 描述  ：简单的循环延时函数
*****************************************************************************/
void delay(uint32_t n)
{
 while(n--);
}
```

虽然注释已经写得很清楚了，但还是讲一下吧，主要是 systick_hw_init 函数，这是初始化配置 SysTick 的代码，第一步先选择 SysTick 的时钟源，这里我选择使用系统内核时钟作为时钟源；第二步是设置 SysTick 的 Load 值，这里设置为 ystemCoreClock/1000 - 1;，这样 SysTick 就会每毫秒中断一次，最后是开启中断和使能 SysTick。

延时主要是通过设定 timeout 的值，然后等待 timeout 为 0，在 Systick 中断中，timeout 会自减 1，直到为 0。这样就实现了 ms 级的精准延时函数 delay_ms()。

在 delay.h 中主要写写函数声明，如下：

```
#ifndef __DELAY_H
#define __DELAY_H
#include "sam4n16c.h"

/*****************************************************************************
* 函数名：SysTick_Handler()
* 参数  ：void
* 返回值：void
* 描述  ：系统节拍中断服务函数
*****************************************************************************/
```

```
void SysTick_Handler(void);
/*****************************************************************************
* 函数名：systick_hw_init()
* 参数  ：void
* 返回值：void
* 描述  ：系统节拍初始化函数
*****************************************************************************/
void systick_hw_init(void);
/*****************************************************************************
* 函数名：delay_ms()
* 参数  ：uint32_t n  延时的毫秒数
* 返回值：void
* 描述  ：使用系统节拍进行精准延时函数
*****************************************************************************/
void delay_ms(uint32_t n);
/*****************************************************************************
* 函数名：delay()
* 参数  ：uint32_t n  延时的循环个数
* 返回值：void
* 描述  ：简单的循环延时函数
*****************************************************************************/
void delay(uint32_t n);
#endif
```

接下来把 main.c 中的 delay 改掉，如下：

```
int main(void)
{
 systick_hw_init();
 led_hw_init();
 while(1){
 led_hw_on();
 delay_ms(500);
 led_hw_off();
 delay_ms(500);
 }

}
```

好了，下载程序到板子，这下是不是延时很准？