昨天搞定了ucos在stm32上的移植，今天继续移植msp430，我用的是IAR6.3编译环境。相比stm32来讲我觉得这个移植更简单一点，官方代码给的已经很好了，所需要的修改很少很少。开始正题。

## 准备材料

在TI官网上下载430的固件库，我用的是msp430f5528的板子，下载的是F5xx\_F6xx\_Core\_Lib

地址<http://www.ti.com/cn/litv/zip/slaa448b>

下载后解压得到



在ucos官网上下载移植430版本的ucos，地址

<http://micrium.com/download/getfile/uCOS-II-MSP-EXP430F5438.exe>

下载安装后得到



## 建立和配置工程

建立工程目录，并在工程目录下建立F5xx\_F6xx\_Core\_Lib和ucosii两个目录

将msp430固件库\slaa448b\F5xx\_F6xx\_Core\_Lib目录下的所有文件拷贝到工程目录下的F5xx\_F6xx\_Core\_Lib文件夹中，这个是msp430cpu操作的一些函数。我们要初始化cpu时钟用。

在在工程目录的ucosii下建立ports和source两个目录用来存放ucos系统文件

将ucos安装目录\Micrium\Software\uCOS-II\Ports\MSP430X\IAR\MSP430x5xx下的所有文件复制到工程目录下的ports文件夹中

将ucos安装目录\Micrium\Software\uCOS-II\Source下的所有文件复制到工程目录下的source文件夹中

建立main函数文件，建一个文本文档取名为test.c

我们还要借鉴ucos安装的例程给的两个系统配置文件

\Micrium\Software\EvalBoards\TI\MSP-EXP430F5438\IAR\uCOS-II目录里的app\_cfg.h和os\_cfg.h。将这两个文件拷贝的工程根目录下。app\_cfg.h是用户进程的一些配置，比如进程的优先级，进程堆栈的大小等等。os\_cfg.h是系统的一些配置，可以开关系统的一些功能，尽量裁剪掉不用的功能可以减少cpu的开销。

下面建立一个msp430的空工程并向工程中添加刚才的文件，得到目录如下



配置这个工程

选取相应的芯片型号



添加工程包含目录

$PROJ\_DIR$\F5xx\_F6xx\_Core\_Lib

$PROJ\_DIR$\

$PROJ\_DIR$\ucosii\ports

$PROJ\_DIR$\ucosii\source



仿真模式改成debug



保存

然后开始建立任务

建立4个任务每个任务闪一个灯和一个启动任务，首先删掉app\_cfg.h中的所有内容，写如下代码：

#ifndef \_\_APP\_CFG\_H\_\_

#define \_\_APP\_CFG\_H\_\_

#define OS\_TASK\_TMR\_PRIO (OS\_LOWEST\_PRIO - 4)

/\* task stack size \*/

#define STARTUP\_TASK\_STK\_SIZE 128

#define TASK1\_STK\_SIZE 128

#define TASK2\_STK\_SIZE 128

#define TASK3\_STK\_SIZE 128

#define TASK4\_STK\_SIZE 128

#define STARTUP\_TASK\_PRIO 4

#define TASK1\_PRIO 5

#define TASK2\_PRIO 6

#define TASK3\_PRIO 7

#define TASK4\_PRIO 8

#endif

这个就是建立的四个任务和一个启动任务的优先级和堆栈大小

下面是任务代码写在test.c中。

OS\_STK startup\_stk[STARTUP\_TASK\_STK\_SIZE];

OS\_STK task1\_stk[TASK1\_STK\_SIZE];

OS\_STK task2\_stk[TASK2\_STK\_SIZE];

OS\_STK task3\_stk[TASK3\_STK\_SIZE];

OS\_STK task4\_stk[TASK4\_STK\_SIZE];

unsigned char os\_err;

static void task1(void \*p\_arg)

{

 for (;;)

 {

 P1OUT |= BIT1;

 OSTimeDly(500);

 P1OUT &= ~BIT1;

 OSTimeDly(500);

 }

}

static void task2(void \*p\_arg)

{

 for (;;)

 {

 P1OUT |= BIT2;

 OSTimeDly(500);

 P1OUT &= ~BIT2;

 OSTimeDly(500);

 }

}

static void task3(void \*p\_arg)

{

 for (;;)

 {

 P1OUT |= BIT3;

 OSTimeDly(500);

 P1OUT &= ~BIT3;

 OSTimeDly(500);

 }

}

static void task4(void \*p\_arg)

{

 for (;;)

 {

 P1OUT |= BIT4;

 OSTimeDly(500);

 P1OUT &= ~BIT4 ;

 OSTimeDly(500);

 }

}

static void startup(void \*p\_arg)

{

 P1DIR |= BIT1+BIT2+BIT3+BIT4; // P1.0 set as output

 OS\_CPU\_SysTickInit();

 #if (OS\_TASK\_STAT\_EN > 0)

 //----统计任务初始化函数

 OSStatInit();

#endif

 os\_err = OSTaskCreate(task1, //任务1

 (void \*)0, //不带参数

 (OS\_STK \*)&task1\_stk[TASK1\_STK\_SIZE - 1],//堆栈指针

 (INT8U)5); //优先级

 os\_err = OSTaskCreate(task2, //任务2

 (void \*)0, //不带参数

 (OS\_STK \*)&task2\_stk[TASK2\_STK\_SIZE - 1],//堆栈指针

 (INT8U)6); //优先级

 os\_err = OSTaskCreate(task3, //任务3

 (void \*)0, //不带参数

 (OS\_STK \*)&task3\_stk[TASK3\_STK\_SIZE - 1],//堆栈指针

 (INT8U)7); //优先级

 //

 os\_err = OSTaskCreate(task4, //任务4

 (void \*)0, //不带参数

 (OS\_STK \*)&task4\_stk[TASK4\_STK\_SIZE - 1],//堆栈指针

 (INT8U)8);

 OSTaskDel(OS\_PRIO\_SELF);

}

我们用的ucos系统时钟是看门狗做的，我们要初始化看门狗滴答如下函数

void OS\_CPU\_SysTickInit()

{

 WDTCTL = (WDTPW+WDTTMSEL+WDTCNTCL+WDTIS\_4); /\*20M/32768=1.6384ms \*/

 SFRIE1 |= 1; /\* Enable WDT interrupts \*/

}

看门狗的中断函数已经写好了在os\_cpu\_a.s43中，可以自己看一下。

下面是初始化cpu时钟 我们吧系统时钟初始化到20M用的是内部PLL，（ucos时钟滴答跟这个有直接的关系，最好能保证看门狗可以设置到1ms的时间，当我们用OSTimeDly函数时时间比较直观）函数如下：

#include <msp430f5528.h>

#include "HAL\_PMM.h"

#include "HAL\_UCS.h"

#include "ucos\_ii.h"

#define SYS\_MCLK\_FREQ 20000000

void Init\_Clock(void)

{

 SELECT\_FLLREF(SELREF\_\_REFOCLK); // Set DCO FLL reference = REFO

 SELECT\_ACLK(SELA\_\_REFOCLK); // Select REFO as ACLK source

 Init\_FLL(SYS\_MCLK\_FREQ/1000, SYS\_MCLK\_FREQ/32768);

}

void InitSys()

{

/\*PMM全部关闭\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 DISABLE\_SVSL(); // SVS Low side is turned off

 DISABLE\_SVML(); // Monitor low side is turned off

 DISABLE\_SVMH(); // Monitor high side is turned off

 DISABLE\_SVSH(); // SVS High side is turned on

 //DISABLE\_SVSH\_RESET(); // Enable POR on SVS Event

 while (((PMMIFG & SVSMLDLYIFG) == 0)&&((PMMIFG & SVSMHDLYIFG) == 0));

 // SVSH turns off in LPM3,power-save

 CLEAR\_PMM\_IFGS();

 \_\_disable\_interrupt(); // Disable global interrupts

 SetVCore(PMMCOREV\_3); // USB core requires the VCore set to 1.8 volt, independ of CPU clock frequency

 Init\_Clock();

 \_\_enable\_interrupt(); // enable global interrupts

}

下面是main函数：直接建立启动任务运行

 void main(void)

{

 volatile unsigned int i;

 WDTCTL = WDTPW+WDTHOLD; // Stop WDT

 InitSys();

 OSInit();

 OSTaskCreate(startup, (void \*)0, &startup\_stk[STARTUP\_TASK\_STK\_SIZE-1], 4);

 OSStart();

}

这样我们的所有工作完成了，直接编译下载仿真，成功。哈哈。