MSP430(F5529)学习笔记——UCS 配置详解

UCS 简介

MSP430F5XX/MSP430F6XX 系列器件的 UCS 包含有五种时钟源，依次是：XT1CLK、VLOCLK、REFOCLK、DCOCLK 和 XT2CLK。这五种时钟的详细介绍请参考该系列芯片的指导手册，其中 XT1CLK、VLOCLK、REFOCLK 和 XT2CLK 跟 MSP430F1XX 系列没有太大区别，学习配置起来也比较简单。

UCS 上电默认状态

PUC 后，UCS 模块的默认状态如下：

1. (1)XT1 处于 LF 模式作为 XT1CLK 时钟源。ACLK 选通为 XT1CLK。
2. (2)MCLK 选通为 DCOCLKDIV
3. (3)SMCLK 选通为 DCOCLKDIV
4. (4)FLL 使能，且将 XT1CLK 作为 FLL 参考时钟。
5. (5)XIN 和 XOUT 脚设置为通用 IO，XIN 和 XOUT 配置为 XT1 功能前，XT1 保持禁用。
6. (6)如果可用的话，XT2IN 和 XT2OUT 被设置为通用 IO 且保持禁止状态。

清楚 UCS 上电默认状态是非常重要的，这对于理解后面的配置逻辑来说非常重要。

UCS 时钟源切换

由于 REFOCLK、VLOCLK、DCOCLK（这里暂时这么认为）默认状态下是可用的，所以，切换的时候只需要通过 UCSCTL4 来配置 ACLK、SMCLK 和 MCLK 的时钟源即可，而 XT1CLK 和 XT2CLK 需要根据硬件的具体配置情况确定，所以，这两者的配置比起前三者来讲，就有些不同了。下面，我们做三个实验：

（1）将 MCLK 和 SMCLK 配置 REFOCLK、VLOCLK

REFOCLK 和 VLOCLK 是芯片默认提供的，只要芯片正常工作，这两个时钟就会正常工作，因此，该时钟配置非常简单，只需要修改 UCSCTL4，将 SELS 和 SELM 配置为对应的选项 VLOCLK 或者 REFOCLK 即可，具体代码如下：

```
#include <msp430f5529.h>
void main(void) {
  WDTCTL = WDTPW+WDTHOLD;
  P1SEL |= BIT0;
  P1DIR |= BIT0;//测量 ACLK 用
  P2SEL |= BIT2;
  P2DIR |= BIT2;//测量 SMCLK 用
  P7SEL |= BIT7;
  P7DIR |= BIT7;//测量 MCLK 用
  //UCSCTL4 = UCSCTL4&(~(SELS_7|SELM_7))|SELS_1|SELM_1; //将 SMCLK 和 MCLK 配置为 VLOCLK
  UCSCTL4 = UCSCTL4&(~(SELS_7|SELM_7))|SELS_2|SELM_2; //将 SMCLK 和 MCLK 配置为 REFOCLK
```

```
12.  while(1);
13. }
```

上面的代码就实现了将 SMCLK 和 MCLK 切换为 VLOCLK 和 REFOCLK，ACLK 的操作也是同样的，不作过多解释。

（2）将 MCLK 和 SMCLK 配置 XT1CLK
我手头上的开发板 XT1 外接的是 32.768K 的手表时钟晶振，XT1CLK 的配置要分为以下几步：

```
1.  1.配置 IO 口 5.4 和 5.5 为 XT1 功能。
2.  2.配置 XCAP 为 XCAP_3，即 12PF 的电容。
3.  3.清除 XT1OFF 标志位。
4.  4.等待 XT1 起振。
```

具体的代码如下：

```
1.  #include <msp430f5529.h>
2.  void main(void) {
3.    WDTCTL = WDTPW+WDTHOLD;
4.    P1SEL |= BIT0;
5.    P1DIR |= BIT0;//测量 ACLK 用
6.    P2SEL |= BIT2;
7.    P2DIR |= BIT2;//测量 SMCLK 用
8.    P7SEL |= BIT7;
9.    P7DIR |= BIT7;//测量 MCLK 用
10.   P5SEL |= BIT4|BIT5; //将 IO 配置为 XT1 功能
11.   UCSCTL6 |= XCAP_3;  //配置电容为 12pF
12.   UCSCTL6 &= ~XT1OFF; //使能 XT1
13.   while (SFRIFG1 & OFIFG){
14.     UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + DCOFFG);          // 清除三类时钟
    标志位
15.                                  // 这里需要清除三种标志位，因为任何一种
16.                                  // 标志位都会将 OFIFG 置位
17.     SFRIFG1 &= ~OFIFG;                                   // 清除时钟错误
    标志位
18.   }
19.   UCSCTL4 = UCSCTL4&(~(SELS_7|SELM_7))|SELS_0|SELM_0;      //将 SMCLK 和
    MCLK 时钟源配置为 XT1
20.   while(1);
21. }
```

（3）将 SMCLK 和 MCLK 配置 XT2

将 SMCLK 和 MCLK 配置为 XT2 跟配置为 XT1 的过程基本相同，唯一不同的是，在配置 SMCLK 和 MCLK 为 XT2 之前，需要将 ACLK 和 REFCLK 的时钟源，因为 ACLK 和 REFCLK 的默认时钟源是 XT1，而我们这里并没有配置启动 XT1CLK，所以会产生 XT1 时钟错误，即 XT1LFFG，因此，我们先将 ACLK 和 REFCLK 配置为芯片自带的时钟（REFOCLK 或 VLOCLK）或者即将启动的时钟（XT2），此外，XT2 配置时不需要配置电容，故将 SMCLK 和 MCLK 配置为 XT2 的代码如下：

```
#include <msp430f5529.h>
void main(void) {
  WDTCTL = WDTPW+WDTHOLD;
  P1SEL |= BIT0;
  P1DIR |= BIT0;//测量 ACLK 用
  P2SEL |= BIT2;
  P2DIR |= BIT2;//测量 SMCLK 用
  P7SEL |= BIT7;
  P7DIR |= BIT7;//测量 MCLK 用
  P5SEL |= BIT2|BIT3; //将 IO 配置为 XT2 功能
  UCSCTL6 &= ~XT2OFF; //使能 XT2
  UCSCTL4 = UCSCTL4&(~(SELA_7))|SELA_1; //先将 ACLK 配置为 VLOCLK
  UCSCTL3 |= SELREF_2;                  //将 REFCLK 配置为 REFCLK
  while (SFRIFG1 & OFIFG){
    UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + DCOFFG);          // 清除三类时钟标志位
                                  // 这里需要清除三种标志位，因为任何一种
                                  // 标志位都会将 OFIFG 置位
    SFRIFG1 &= ~OFIFG;                                   // 清除时钟错误标志位
  }
  UCSCTL4 = UCSCTL4&(~(SELS_7|SELM_7))|SELS_5|SELM_5;     //将 SMCLK 和 MCLK 时钟源配置为 XT2
  while(1);
    }
```

DCO 模块详解

DCO 模块在 MSP430F5XX 系列芯片中非常重要，因为从 MSP430F4XX 开始，MSP430 引用了 FLL 模块，FLL 即锁相环，可以通过倍频的方式提高系统时钟频率，进而提高系统的运行速度。

DCO 模块运行需要参考时钟 REFCLK,REFCLK 可以来自 REFOCLK、XT1CLK 和 XT2CLK，通过 UCSCTL3 的 SELREF 选择，默认使用的 XT1CLK，但如果 XT1CLK 不可用则使用 REFOCLK。

DCO 模块有两个输出时钟信号，级 DCOCLK 和 DCOCLKDIV，其中，倍频计算公式如下：

```
DCOCLK = D*(N+1)*(REFCLK/n)
DCOCLKDIV = (N+1)*(REFCLK/n)
```

其中：
n 即 REFCLK 输入时钟分频，可以通过 UCSCTL3 中的 FLLCLKDIV 设定，默认为 0，也就是不分频；
D 可以通过 UCSCTL2 中的 FLLD 来设定，默认为 1，也就是 2 分频；
N 可以通过 UCSCTL2 中的 FLLN 来设定，默认值为 32。
所以，系统上电后如果不做任何设置，DCOCLK 的实际值为 2097152，DCOCLKDIV 的实际值为 1048576。
另外，配置芯片工作频率还需要配置 DCORSEL 和 DCOx，DCORSEL 和 DCOx 的具体作用如下：
DCORSEL 位于 UCSCTL1 控制寄存器中的 4 到 6 位，共 3 位，将 DCO 分为 8 个频率段。
DCOx 位于 UCSCTL0 中的 8 到 12 位，共 5 位，将 DCORSEL 选择的频率段分为 32 个频率阶，每阶比前一阶高出约 8%，该寄存器系统可以自动调整，通常配置为 0。
DCORSEL 和 DCOx 值的具体作用可以参考 MSP430F5529 的数据手册，阅读该手册相关部分可以找到如下表格：

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$f_{DCO(0,0)}$	DCO frequency (0, 0)	DCORSELx = 0, DCOx = 0, MODx = 0	0.07		0.20	MHz
$f_{DCO(0,31)}$	DCO frequency (0, 31)	DCORSELx = 0, DCOx = 31, MODx = 0	0.70		1.70	MHz
$f_{DCO(1,0)}$	DCO frequency (1, 0)	DCORSELx = 1, DCOx = 0, MODx = 0	0.15		0.36	MHz
$f_{DCO(1,31)}$	DCO frequency (1, 31)	DCORSELx = 1, DCOx = 31, MODx = 0	1.47		3.45	MHz
$f_{DCO(2,0)}$	DCO frequency (2, 0)	DCORSELx = 2, DCOx = 0, MODx = 0	0.32		0.75	MHz
$f_{DCO(2,31)}$	DCO frequency (2, 31)	DCORSELx = 2, DCOx = 31, MODx = 0	3.17		7.38	MHz
$f_{DCO(3,0)}$	DCO frequency (3, 0)	DCORSELx = 3, DCOx = 0, MODx = 0	0.64		1.51	MHz
$f_{DCO(3,31)}$	DCO frequency (3, 31)	DCORSELx = 3, DCOx = 31, MODx = 0	6.07		14.0	MHz
$f_{DCO(4,0)}$	DCO frequency (4, 0)	DCORSELx = 4, DCOx = 0, MODx = 0	1.3		3.2	MHz
$f_{DCO(4,31)}$	DCO frequency (4, 31)	DCORSELx = 4, DCOx = 31, MODx = 0	12.3		28.2	MHz
$f_{DCO(5,0)}$	DCO frequency (5, 0)	DCORSELx = 5, DCOx = 0, MODx = 0	2.5		6.0	MHz
$f_{DCO(5,31)}$	DCO frequency (5, 31)	DCORSELx = 5, DCOx = 31, MODx = 0	23.7		54.1	MHz
$f_{DCO(6,0)}$	DCO frequency (6, 0)	DCORSELx = 6, DCOx = 0, MODx = 0	4.6		10.7	MHz
$f_{DCO(6,31)}$	DCO frequency (6, 31)	DCORSELx = 6, DCOx = 31, MODx = 0	39.0		88.0	MHz
$f_{DCO(7,0)}$	DCO frequency (7, 0)	DCORSELx = 7, DCOx = 0, MODx = 0	8.5		19.6	MHz
$f_{DCO(7,31)}$	DCO frequency (7, 31)	DCORSELx = 7, DCOx = 31, MODx = 0	60		135	MHz
$S_{DCORSEL}$	Frequency step between range DCORSEL and DCORSEL + 1	$S_{RSEL} = f_{DCO(DCORSEL+1,DCO)}/f_{DCO(DCORSEL,DCO)}$	1.2		2.3	ratio
S_{DCO}	Frequency step between tap DCO and DCO + 1	$S_{DCO} = f_{DCO(DCORSEL,DCO+1)}/f_{DCO(DCORSEL,DCO)}$	1.02		1.12	ratio
	Duty cycle	Measured at SMCLK	40	50	60	%
df_{DCO}/dT	DCO frequency temperature drift[1]	f_{DCO} = 1 MHz,		0.1		%/°C
df_{DCO}/dV_{CC}	DCO frequency voltage drift[2]	f_{DCO} = 1 MHz		1.9		

可以见，DCORESL 的频率调节范围大致如下：

```
1. DCORSEL = 0 的调节范围约为 0.20~0.70MHZ；
2. DCORSEL= 1 的调节范围约为 0.36~1.47MHZ；
3. DCORSEL = 2 的调节范围约为 0.75~3.17MHZ；
4. DCORSEL = 3 的调节范围约为 1.51~6.07MHZ；
5. DCORSEL = 4 的调节范围约为 3.2~12.3MHZ；
6. DCORSEL = 5 的调节范围约为 6.0~23.7MHZ；
7. DCORSEL = 6 的调节范围约为 10.7~39.7MHZ；
8. DCORSEL = 7 的调节范围约为 19.6~60MHZ。
```

理解了上面这些，可以理解 TI 官方例子中的代码了，官方代码中的相关部分如下：

```
1. if (fsystem <= 630)            //           fsystem < 0.63MHz
2.   UCSCTL1 = DCORSEL_0;
3. else if (fsystem <  1250)      // 0.63MHz < fsystem < 1.25MHz
4.   UCSCTL1 = DCORSEL_1;
5. else if (fsystem <  2500)      // 1.25MHz < fsystem <  2.5MHz
6.   UCSCTL1 = DCORSEL_2;
7. else if (fsystem <  5000)      // 2.5MHz  < fsystem <    5MHz
8.   UCSCTL1 = DCORSEL_3;
9. else if (fsystem <  10000)     // 5MHz    < fsystem <   10MHz
10.  UCSCTL1 = DCORSEL_4;
11.else if (fsystem <  20000)     // 10MHz   < fsystem <   20MHz
12.  UCSCTL1 = DCORSEL_5;
13.else if (fsystem <  40000)     // 20MHz   < fsystem <   40MHz
14.  UCSCTL1 = DCORSEL_6;
15.else
16.  UCSCTL1 = DCORSEL_7;
```

都在前面讲到的范围内，由于前面的范围有重叠部分，例子代码中的值是 TI 的工程师根据上面这些参数选取的比较合理的值。
到这里，我相信大家配合芯片手册和本文，都能明白 DCO 配置相关部分的原理了，下面是将 DCO 参考时钟选为 XT1，并将 DCOCLK 倍频到 25M 的详细代码：

```
1. #include <msp430f5529.h>
2.
3. void delay(){
4.   volatile unsigned int i;
5.   for(i = 0; i != 5000; ++i){
6.     _NOP();
7.   }
8. }
9.
10.void SetVcoreUp (unsigned int level)
```

```
{
  // Open PMM registers for write
  PMMCTL0_H = PMMPW_H;
  // Set SVS/SVM high side new level
  SVSMHCTL = SVSHE + SVSHRVL0 * level + SVMHE + SVSMHRRL0 * level;
  // Set SVM low side to new level
  SVSMLCTL = SVSLE + SVMLE + SVSMLRRL0 * level;
  // Wait till SVM is settled
  while ((PMMIFG & SVSMLDLYIFG) == 0);
  // Clear already set flags
  PMMIFG &= ~(SVMLVLRIFG + SVMLIFG);
  // Set VCore to new level
  PMMCTL0_L = PMMCOREV0 * level;
  // Wait till new level reached
  if ((PMMIFG & SVMLIFG))
    while ((PMMIFG & SVMLVLRIFG) == 0);
  // Set SVS/SVM low side to new level
  SVSMLCTL = SVSLE + SVSLRVL0 * level + SVMLE + SVSMLRRL0 * level;
  // Lock PMM registers for write access
  PMMCTL0_H = 0x00;
}

void main(void) {
  WDTCTL = WDTPW+WDTHOLD;
  P1SEL &= ~BIT1;
  P1DIR |= BIT1;

  P1SEL |= BIT0; //ACLK
  P1DIR |= BIT0;
  P2SEL |= BIT2; //SMCLK
  P2DIR |= BIT2;
  P7SEL |= BIT7; //MCLK
  P7DIR |= BIT7;

  P5SEL |= BIT4|BIT5;
  UCSCTL6 |= XCAP_3;
  UCSCTL6 &= ~XT1OFF;

  SetVcoreUp(1); //一次提高 Vcore 电压等级，具体请参考手册
  SetVcoreUp(2);
  SetVcoreUp(3);

  __bis_SR_register(SCG0);
  UCSCTL0 = 0;
```

```
  UCSCTL1 = DCORSEL_6;
  UCSCTL2 = FLLD_1 | 380;
  __bic_SR_register(SCG0);
  __delay_cycles(782000);

  /*
   * 默认状态下：ACLK=FLLREFCLK=XT1 SMCLK=MCLK=DCOCLKDIV XT2 关闭
   * 为了不产生 XT1LFOFFG，将 ACLK 和 FLLREFCLK 设置为 REFOCLK
   * 并打开 XT2OFF，否则 XT2 将处于无法使用状态
   * */
  //UCSCTL6 &= ~(XT2DRIVE0|XT2DRIVE1|XT2OFF);

  while (SFRIFG1 & OFIFG) {                           // Check OFIFG fault flag
    UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + DCOFFG);       // Clear OSC flaut Flags
    SFRIFG1 &= ~OFIFG;                                // Clear OFIFG fault flag
  }

  UCSCTL4 = UCSCTL4&(~(SELS_7|SELM_7))|SELS_3|SELM_3;

  while(1){
    P1OUT ^= BIT1;
    delay();
  }
}
```