

实验 5 基本时钟和定时功能

一. 实验目的

1. 了解 MSP430Gxxx 基本时钟模块的作用和工作原理，掌握其控制方法；
2. 掌握利用时钟信号和中断技术实现定时功能的方法；
3. (提高)掌握低功耗模式控制方法。

二. 实验任务

1. 了解上电复位基本时钟模块各时钟频率和控制寄存器作用。

如图 5-1 将实验板 JP8 中间两个插针接到 32.768KHz 晶振侧；图 5-2 是板背面的晶振实物。若按图 5-3 接线，晶振则未连接到单片机。

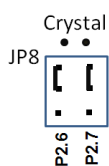


图 5-1



图 5-2

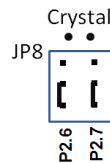


图 5-3

参看图 5-4，结合第 5 章基本时钟课件例 1 的分析，了解 msp430g2553 上电复位 ACLK、SMCLK 和 MCLK 时钟信号与片外低频振荡器 XT1、内部低频振荡器 VLOCLK、内部数字振荡器 DCO 三个时钟源关系，和相关控制寄存器起到的选择、分频等作用，控制位 DCOx、RSELx 与数字控制振荡器输出时钟频率关系。

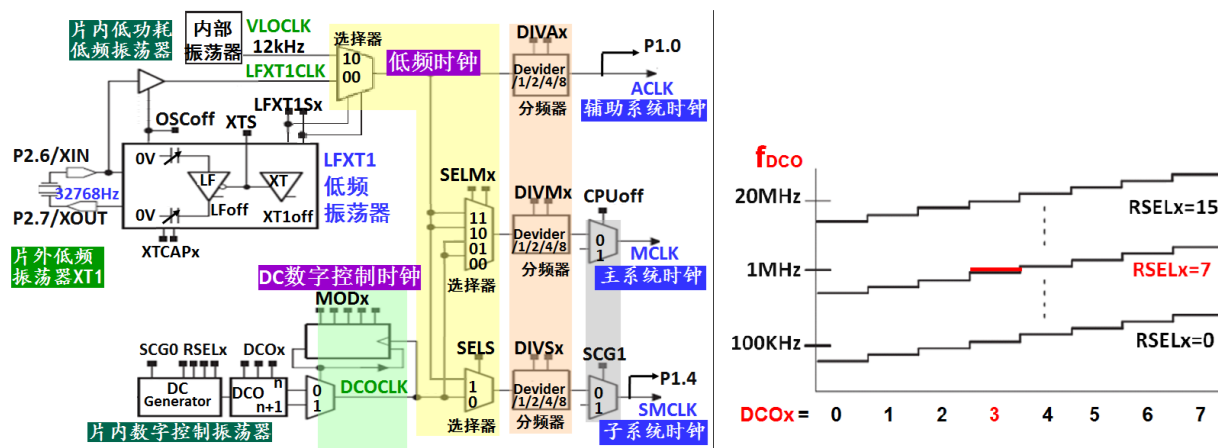



图 5-4 g2553 基本时钟示意图(左图)，控制位 DCOx 和 RSELx 与数字控制振荡器输出时钟频率关系(右图)

阅读 L5_testClk.C 程序，用单片机的引脚 P2.0 和 P2.5 分别连到发光二极管 L1、L6，如有示波器，可测量 P1.0、P1.4 分别输出的 ACLK、SMCLK 时钟频率值；如无示波器，可采用观察发光二极管的闪烁速度快慢，来判断 CPU 工作时钟 MCLK 频率的变化，并掌握基本时钟模块 4 个控制寄存器各控制位的作用。在 CCS 下进入 DEBUG，完成下面操作。

- 1) 对下面四种情况下，在表 5-1 中用慢、较慢、快、很快四种程度记录发光二极管闪烁的速度。
(1) 运行程序，用 View/Registers 查看上电复位时 System Clock 模块中 DCOCTL、BCSCTL1 的 DCOx、RSELx 的值，填入表 5-1 中，用二进制数表示即可；(2) 暂停运行，如图 5-5 通过更改 BCSCTL1，设置 DCOx=111，RSELx=1111，运行程序；(3) 暂停，再置 DCOx=000，RSELx=0000，运行程序；(4) 暂停程序，如图 5-6 置 DIVMx=11，即 DIVM_3。MCLK 频率可从图 5-4 的控制位 DCOx、RSELx 与数字控制振荡器输出时钟频率 f_{DCO} 关系图粗略估值。
- 2) Debug 下暂停程序运行，(1) 点击复位 ，回到上电复位状态，然后如图 5-7，设置 SELMx=10，即 SELM_2，MCLK 时钟来源为低频时钟，因此时 LFXT1Sx=LFXT1S_0，故 MCLK 时钟来源为外部 32.768KHz

晶振。如图 5-8 设置 Special Function 模块中寄存器 IFG1 中的振荡失效位 OFIFG 为 0，然后运行程序，观察时钟闪烁频率。注意要确保振荡失效标志 OFIFG 为 0，否则则为 1，表示此时 MCLK 尚未切换成功，仍来自 DCO 振荡器。(2) 如图 5-9，设置 LFXT1Sx=10，即 LFXT1S_2，改变低频时钟源为 VLOCLK，并如图 5-7 确保 OFIFG 为 0，运行程序。观察 LED 灯闪烁情况，完成表 5-2 的填写。有示波器的同学填写测量值，无示波器的同学填估计值。

思考：1) 选择使用 P2.6、P2.7 连接的外部 32.768KHz 晶振作为时钟源，程序 L5_testClk.c 没有对 P2SEL、P2SEL2 相应位的设置，可以吗？为什么？

2) 分析 msp430g2553 的 MCLK 最低工作频率大约是多少？

System_Clock	
DCOCTL	0xE0
DCO2	1
DCO1	1
DCO0	1
MOD4	0
MOD3	0
MOD2	0
MOD1	0
MOD0	0

System_Clock	
DCOCTL	0xE0
BCSCTL1	0x8F
XT2OFF	1
XTS	0
DIVA	00 - DIVA_0
RSEL3	1
RSEL2	1
RSEL1	1
RSEL0	1

图 5-5 修改 DCOx、RSELx 的值

Name	Value
System_Clock	
DCOCTL	0x00
BCSCTL1	0x00
BCSCTL2	0x00
SELM	00 - SELM_0
DIVM	DIVM_3
SELS	DIVM_0
DIVS	DIVM_1
BCSCTL3	DIVM_2
	DIVM_3

图 5-6 设置 MCLK 的分频值 DIVM

Name	Value
System_Clock	
DCOCTL	0x60
BCSCTL1	0x87
BCSCTL2	0x00
SELM	SELM_2
DIVM	SELM_0
SELS	SELM_1
DIVS	SELM_2
BCSCTL3	SELM_3
	0x04

图 5-7 设置 SELMx 改变 MCLK 来源低频时钟

Name	Value
Special_Function	
IE1	0x00
IFG1	0x06
NMIIFG	0
RSTIFG	0
PORIFG	1
OFIFG	0
WDTIFG	0
IE2	0x00
IFG2	0x0A

图 5-8 清除切换 MCLK 时钟来源引起的 OFIFG

Name	Value
System_Clock	
DCOCTL	0x60
BCSCTL1	0x87
BCSCTL2	0x80
BCSCTL3	0x04
XT2S	00 - XT2S_0
LFXT1S	LFXT1S_2
XCAP	LFXT1S_0
XT2OF	LFXT1S_1
LFXT1OF	LFXT1S_2
	LFXT1S_3

图 5-9 设置 LFXT1Sx=10，选择 VLOCLK

表 5-1 主系统时钟 MCLK 的频率控制（来自 DCO 时钟）

任务	SELMx	DCOx	RSELx	DIVMx	LED 闪烁速度	MCLK 频率测量值或估计值
(1)	00			00		
(2)	00	111	1111	00		
(3)	00	000	0000	00		
(4)	00	000	0000	11		

表 5-2 主系统时钟 MCLK 的频率控制（来自低频时钟情况）

任务	SELMx	LFXT1Sx	OFIFG	DIVMx	LED 闪烁速度	MCLK 频率测量值或估计值
(1)	10	00	0	00		
(2)	10	10	0	00		

任务 1: L5_TestClk.C（提供电子版）:

```
#include "msp430.h"
#define Led_a BIT0 //定义与 LED 连接的引脚
#define Led_b BIT5
unsigned int i; //定义延时变量
int main ( void )
{
    WDTCTL = WDTPW + WDTHOLD; //关闭看门狗

    //下面这段代码时钟信号输出代码，有示波器时用，无示波器时可不用
    P1SEL |=BIT0; //设置 P1.0 输出 ACLK 时钟
    P1SEL2 &= ~BIT0;
    P1DIR |=BIT0;
    P1SEL |=BIT4; //设置 P1.4 输出 SMCLK 时钟
    P1SEL2 &= ~BIT4;
    P1DIR |=BIT4;
    //

    P2SEL &=~(Led_a+Led_b); //设置引脚 P2.0 和 P2.5 为基本输入输出功能
    P2SEL2 &=~(Led_a+Led_b);
    P2OUT |=Led_a+Led_b; //设置引脚 P2.0 和 P2.5 输出的初值为 1
    P2DIR |=Led_a+Led_b; //设置端口 P2.0 和 P2.5 为输出方向
    while (1) //主循环
    {
        P2OUT ^=(Led_a+Led_b); //将 P2.0 和 P2.5 的值取反后输出
        for (i=0xFF; i>0; i--); //延时
    }
}
```

2. 内联延时宏定义 __delay_cycles()的应用

在 CCS 的库里，定义了一些内联函数（intrinsic functions），可在 intrinsics.h 文件中查看到它们的声明，比如 void __delay_cycles(unsigned long cycles) 是其中之一，只要包含了 msp430.h 文件即可使用，因为在 msp430g2553.h 中包含有 intrinsics.h。__delay_cycles(unsigned long cycles)实质上是一个宏定义，该定义起到根据常数 cycles 的大小，用相应的一些指令形成程序代码，这段代码的执行时间与 cycles 值的大小、及当前 CPU 运行的主频 MCLK 有关。可表示为：

__delay_cycles(unsigned long cycles)的执行时间

= cycles*CPU 的 T 周期

= cycles/CPU 的工作频率

= cycles/MCLK

请根据 CPU 运行的 MCLK 时钟频率值，利用上电复位的 MCLK 的实测值，在 L5_DelayCycles.C 基础上，设定 number 值的大小，使与 P2.0 连接的发光二极管，按 1 秒的速度闪烁，即亮半秒，灭半秒。(线上同学没有示波器，请用上电复位的 MCLK 约为 1MHz 计算和观察)。

任务 2: L5_DelayCycles.c (提供电子版):

```
#include "msp430.h"
#define number 15 //可根据实际需要，改变该常数大小
void main( void )
{
    WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer to prevent time out reset
    P2DIR |= BIT0;
    while(1)
    {
        P2OUT |= BIT0;
        __delay_cycles(number);
        P2OUT &= ~BIT0;
        __delay_cycles(number);
    }
}
```

3. 利用时钟信号做中断源，实现定时功能

如图 5-1 确认 JP8 中间插针信号用短线块接至晶振 32.768KHz 侧，并按下页图 5-10，用导线将 P1.0 与 P1.5 相连。编程控制基本时钟模块，使 ACLK 时钟频率为 $32768\text{Hz}/8=4096\text{Hz}$ ，即对低频时钟 32.768KHz 进行 8 分频，并通过 P1.0 输出 ACLK 时钟信号，用此时钟信号的上升沿作为引脚 P1.5 的中断信号。如果引脚 P1.5 上的中断申请被响应，CPU 将以每秒 4096 次的频率执行 P1.5 对应的中断函数。利用这一特点，在 P1.5 的中断函数中设置一个计数变量，计数中断函数被执行的次数，每被执行 2048 次表示半秒时间到，然后设置计数值回 0，重新计数。MCLK 为上电复位时钟。

1) 利用该定时功能，让一个发光二极管以 1 秒的频率闪烁，即亮半秒、灭半秒。请阅读给出的程序 L5_CountClk.c，了解程序思路，并在“.....”处填写相应代码，调试出该功能。

思考：(1) 中断函数中不清 num_clk 为 0 的话，结果会怎样？

(2) 引脚 P1.0、P1.5 可否换成其他引脚？为什么？

2) (提高) 如何在 1) 的编程基础上，增加一个计数秒值的变量，实现每隔 6 秒蜂鸣器响 4 声？（注意不是用 1) 中直接改变计数值 2048 的方法）。

思考：蜂鸣响的函数 Buzz() 可否放在中断函数中调用，为什么？

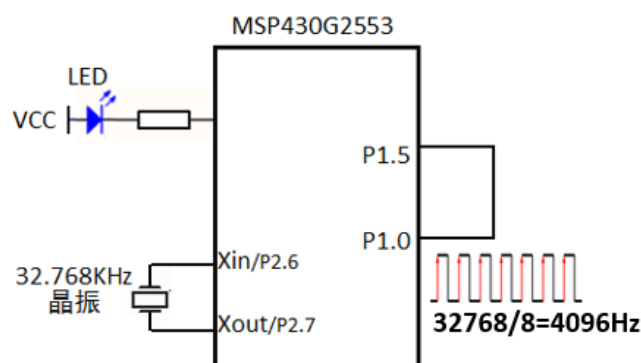


图 5-10 定时功能接线示意图

任务 3: L5_CountClk.C(提供电子版)

```
#include "msp430.h"
void main ( void )
{   WDTCTL = WDTPW + WDTHOLD;           //关闭看门狗

//初始化 LED 引脚，基本输出，初值 LED 灭
    .....

//设置基本时钟 ACLK =32768Hz/8=4096Hz(上电复位 ACLK 来自外部晶振时钟)
    .....

//设置 P1.0 输出时钟 ACLK
    .....

//中断引脚的相关设置
    .....

    _EINT();                             //总中断允许

    while (1) {   };                     //主循环
}

unsigned int num_clk=0;                  //计数时钟个数变量
#pragma vector=PORT1_VECTOR
__interrupt void count_clk( )
{   num_clk++;                           //中断次数加 1，即时钟个数加 1
    if (num_clk==2048)                   //半秒计数时钟个数到
    {   .....   ;                       //对 LED 状态取反，即半秒变化一次:亮半秒,灭半秒
        num_clk=0;                       //计数时钟个数清零
    }

    .....                               //清除中断标志
}
```