**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。

1. **实验任务**
2. **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。

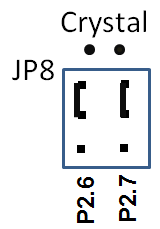
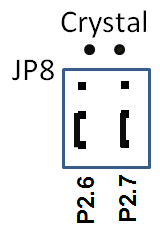
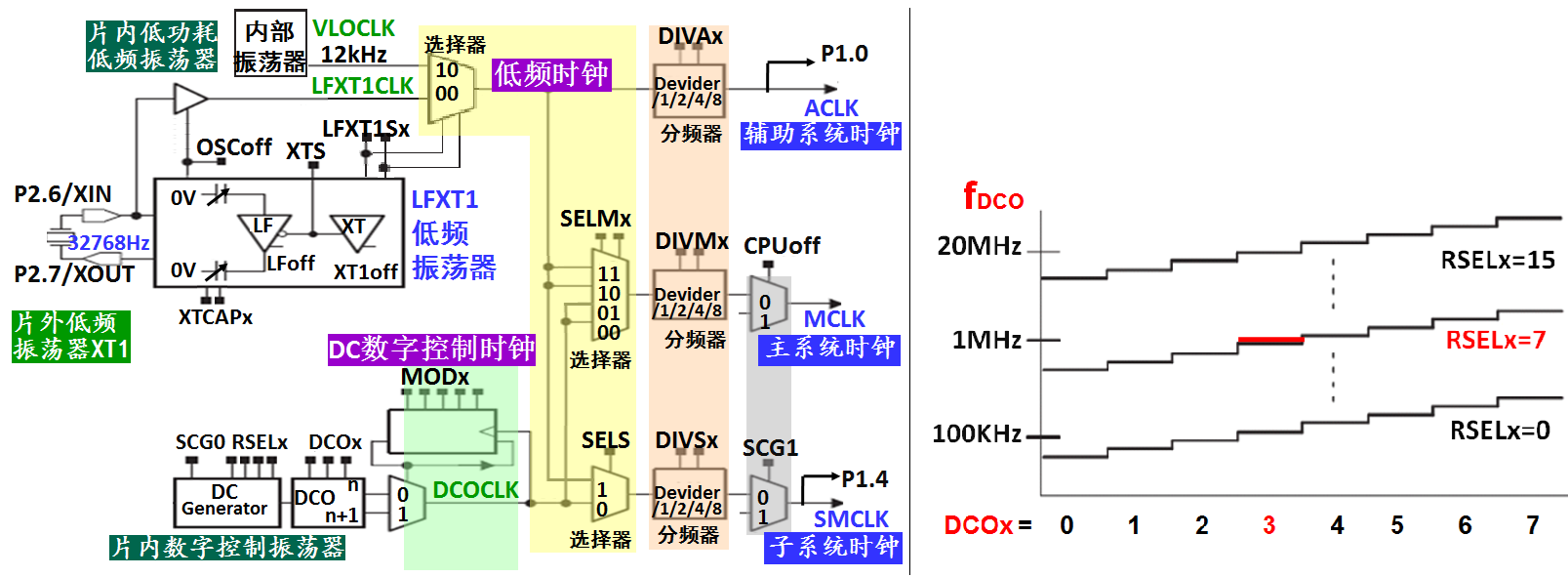


图5-1 图5-2 图5-3

参看图5-4，结合第6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，**线下同学**有示波器，可测量P1.0、P1.4分别输出的ACLK、SMCLK时钟频率值，并记录下来；**线上同学**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。
2. 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。因MCLK频率没有引脚直接输出，不能通过测量直接得到，但可通过图5-4利用ACLK或SMCLK的测量值，间接得到。根据图5-4中msp430g2553时钟示意图，根据MCLK与ACLK、SMCLK的时钟信号来源，**线下同学**借用示波器测量得到的ACLK或SMCLK频率值，**填写MCLK的频率值**；**线上同学**根据图5-4右侧的图，**填写MCLK的估计值**。
3. 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FDCO关系图粗略估值。
4. 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，否则若为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。

**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？

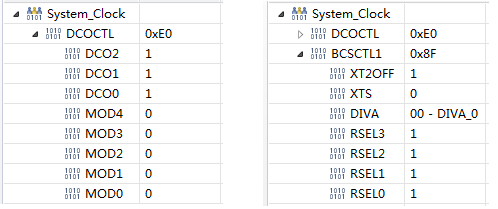


图5-5修改DCOx、RSELx的值

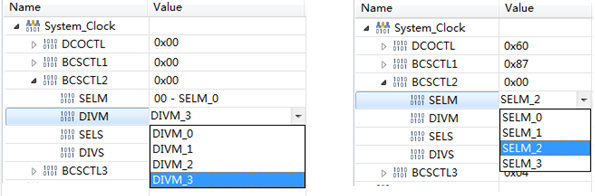


图5-6 设置MCLK的分频值DIVM 图5-7 设置SELMx改变MCLK来源低频时钟

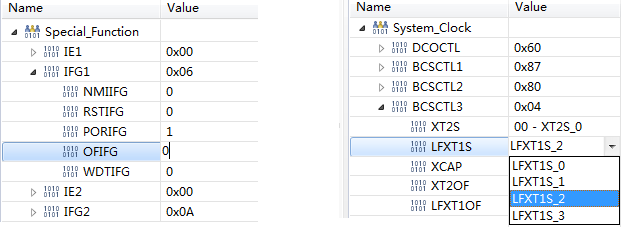


图5-8清除切换MCLK时钟来源引起的OFIFG 图5-9设置LFXT1Sx=10，选择VLOCLK

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 //实验板上如有引脚损坏，可以方便改用其他引脚

unsigned int i; //定义延时用变量

int main ( void )

{ WDTCTL = WDTPW + WDTHOLD; //关闭看门狗

//在单片机引脚上输出时钟信号，可利用示波器观察和测量，或给外部电路使用(如任务3)

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;

//设置用到的基本I/O引脚

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

//主循环

while (1)

{ P2OUT ^=(Led\_a+Led\_b); //将P2.0和P2.5的值取反后输出

for (i=0xFFF; i>0; i--); //延时

}

}

1. **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intinsics.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

理解了宏定义\_\_delay\_cycles()的原理，已知MCLK频率值，在需要延时的地方，可用它得到相对准确的延时。

1. 请根据任务1中上电复位的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)。**
2. 在1)的基础上，改写实验4中的延时函数delay(unsigned int i)，实现根据传递的形参i值的大小，结合宏定义\_\_delay\_cycles，提供可选择时长可知的延时函数。如，当i的值分别是1、2、3、4等不同值时，分别延时0.25s、0.5s、0.75s、1s。实际应用时，可根据需要再具体调整。请比较用改写后的delay(unsigned int i)函数延时，与直接只用宏定义\_\_delay\_cycles进行延时的不同。

任务2：L5\_DelayCycles.c（提供电子版）：

#include "msp430.h"

#define number 100 //可根据需要，改变该常数大小

void main( void )

{

WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer to prevent time out reset

P2DIR |=BIT0; //在上电复位基础上，设置P2.0为输出

while(1)

{ P2OUT|=BIT0; //置P2.0为1

\_\_delay\_cycles(number);

P2OUT&=~BIT0; //置P2.0为0

\_\_delay\_cycles(number);

}

}

1. **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P1.3与P1.0相连。编程控制基本时钟模块，使ACLK时钟频率为32768Hz/8=4096Hz，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P1.3的中断信号。如果引脚P1.3上的中断申请被响应，CPU将以每秒4096次的频率执行P1.3对应的中断函数。利用这一特点，在P1.3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。

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？为什么？

(2) 引脚P1.0、P1.3可否换成其他引脚？为什么？

(3) **(提高)**如果改用P1.4输出的上电复位SMCLK作为P1.3的中断信号, 完成任务1)功能，

如何修改程序? 改好后，请上机调试，解释看到的现象。

1. **(提高)** 如何在1）的编程基础上，增加一个**计数秒值**的变量，即该变量每半秒加1，实现每隔6秒蜂鸣器响5声？（注意不是将1）程序中的计数值2048，直接改为2048\*12=24576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？

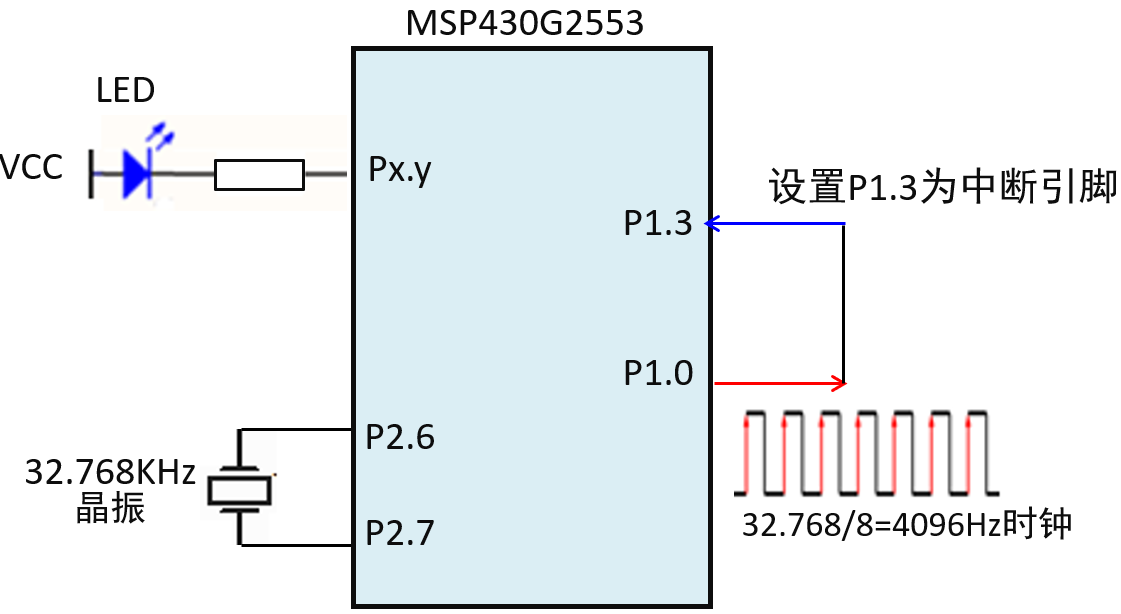


图5-10 定时功能接线示意图

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= //设置中断向量

\_\_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

}