FT61F13X

MSCK Application note

目录

FT61F13x MSCK 应用

1. 振荡器和系统时钟

系统时钟(SysClk) 可通过指令选择为内部高速振荡器 HIRC，内部低速振荡器 LIRC，或外部振荡器 (EC, LP, XT, 参阅 “SCS”, **表 1-2**)。如果选择外部振荡器，那么则由初始化配置寄存器 “FOSC” (**表 1-1**) 决定选用 3 种外部振荡器之一。系统时钟还可通过指令进一步选择为内部振荡器的分频 (参阅 IRCF，**表 1-2**)。系统时钟用于产生指令时钟(Instruction Clock)：

指令时钟 = SysClk / N; N = 2 for 2T, 4 for 4T.

外部时钟输入和内部指令时钟输出的引脚分配由初始化配置寄存器设置 (参阅 FOSC)。如果使能指令时钟输出，那么则由 ”CLKOS” (**表 1-3**) 选择输出端口为 PB0 或 PA2。

Timers 和 ADC 模块有独立的振荡器，因此可有多个振荡器同时运行。

当 Timers 使能时，其选用的振荡器将自动开启，且在 Timers 运行期间一直保持有效。SLEEP 模式可将振荡器配置为开启或关闭。当相应的振荡器在 SLEEP 模式下保持运行时，ADC, Timers 和 PWM 功能同样可在 SLEEP 时工作。

SLEEP 模式下指令停止运行，而指令时钟也将停止，因此选择指令时钟作为时钟源的外设模块也将在 SLEEP 模式下停止工作。

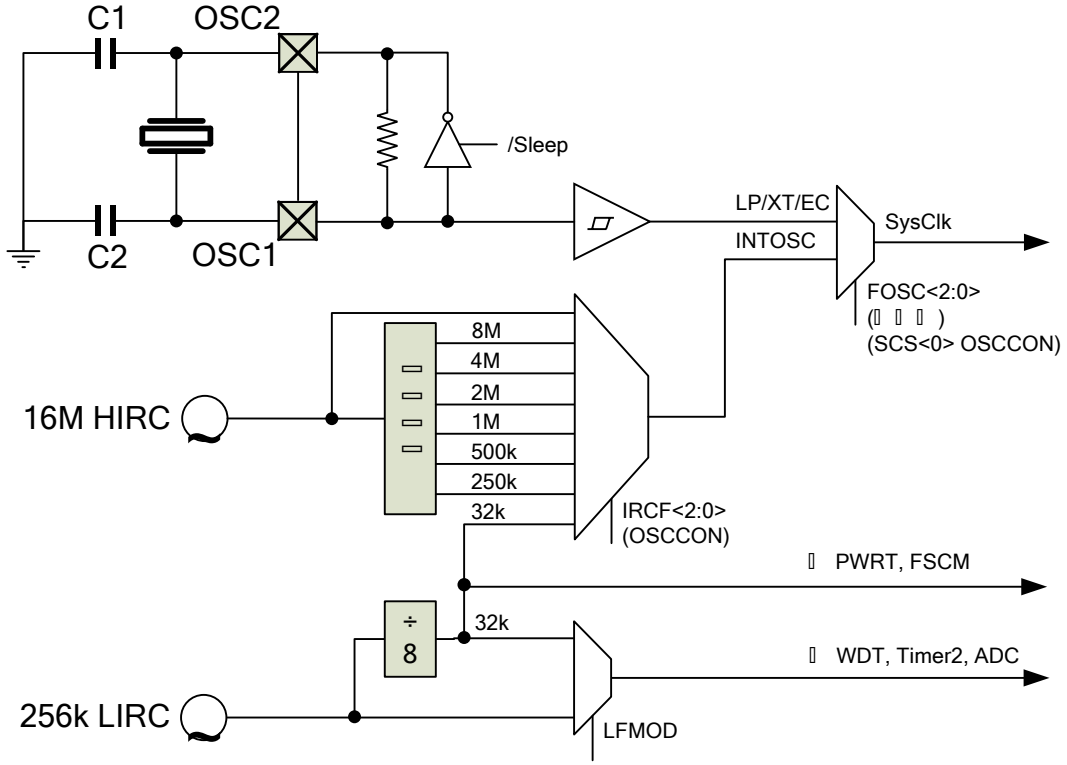

图 1-1 系统时钟 SysClk 的时钟源框图

1.1. 振荡器模块相关寄存器汇总

名称	功能	默认
FOSC	• LP：PC1 (+) 和 PC0 (−) 接外部低速晶振 • XT：PC1 (+) 和 PC0 (−) 接外部高速晶振 • EC：PC1 (+) 接外部时钟输入，PC0 为 I/O • INTOSC：PB0 或 PA2 输出“指令时钟”，PC0 和 PC1 为 I/O • INTOSCIO：PC0 和 PC1 为 I/O	INTOSCIO
IESO	XT / LP 双速时钟启动 • 使能 • 关闭	使能
FCMEN	故障保护时钟监控器 • 使能 • 关闭	使能
TSEL	指令时钟与系统时钟的对应关系 (2T or 4T) • 2 (指令时钟 = SysClk/2) • 4 (指令时钟 = SysClk/4)	2

表 1-1 FOSC 和双速启动初始化配置寄存器

SysClk 系统时钟源			配置			
			SCS	IRCF	LFMOD	OST
			OSCCON[0]	OSCCON[6:4]	OSCCON[7]	(固定值)
			0x8F			
			RW-0	RW-100	RW-0	
外部	EC		0	-	-	-
	XT		0	-	-	1,024
	LP		0	-	-	32,768
内部	HIRC	16 MHz	1	111	-	-
		8 MHz	1	110	-	-
		4 MHz	1	101	-	-
		2 MHz	1	100	-	-
		1 MHz	1	011	-	-
		500 kHz	1	010	-	-
		250 kHz	1	001	-	-
	LIRC	256 kHz [1]	1	000	1	-
		32 kHz [2]	1	000	0	-

表 1-2 SysClk 系统时钟源设置相关用户寄存器

[1] 256 kHz LIRC 只供 WDT 、Timer2 和 ADC 使用。

[2] 系统时钟源 (IRCF=000)、PWRT 和 FSCM 统一使用 LIRC 的 8 分频，即 32 kHz，而不管 LFMOD 为何值。

名称	状态	寄存器	地址	复位值
OSTS	振荡器启动超时状态位(锁存) 1 = 运行在外部振荡器下(启动成功) 0 = 运行在内部振荡器下	OSCCON[3]	0x8F	RO-x
HTS	HIRC ready (锁存) 1 = Yes 0 = No	OSCCON[2]		RO-0
LTS	LIRC ready (锁存) 1 = Yes 0 = No	OSCCON[1]		RO-0
CLKOS	内部时钟输出 CLKO 功能输出引脚控制位 (仅当 FOSC 选择 INTOSC 模式时有效) 1 = CLKO 映射到 PB0; 0 = CLKO 映射到 PA2;	MSCON0[4]	0x1B	RW-1
CKMAVG	LIRC 和 HIRC 交叉校准时 4 次平均测量模式 1 = 使能 0 = 关闭	MSCON0[2]		RW-0
CKCNTI	启动 LIRC 和 HIRC 的交叉校准功能 1 = 启动 0 = 完成(自动清零)	MSCON0[1]		RW-0
SOSCPR	校准 LIRC 周期所需的 HIRC 周期数	SOSCPR[11:0]	0x1D[3:0] 0x1C	RW-FFF

表 1-3 振荡器控制位/状态位

1.2. 内部时钟模式 (HIRC 和 LIRC)

内部高频时钟 (Internal high frequency clock, HIRC) 出厂时已校准到 16 MHz @ 2.5V/25°C。芯片之间的频率变化典型值 < ±1.5% @2.5 – 5.5V/25°C，温度变化典型值为 ±4% @$^{-}$40 – $^{+}$105 °C。

HIRC 精度在晶圆测试时已进行校准。封装过程可能会导致 HIRC 频率漂移。烧录器软件可选择是否需要对 HIRC 进行重新校准，此外，还可选择是否将校准后的 HIRC 频率误差存储到数据 EEPROM 的最后一个字节。每一个 step 代表 2% / 128 = 0.016%的误差。HIRC 出厂校准值已存储在“FOSCCAL”寄存器中，用户可以从默认的 16 MHz 来改变 HIRC 频率(微调)，微调 steps 是非线性的(~40 kHz)。粗略估计如下：

$$FOSCCAL[7:0] \pm N \approx 16000 \pm N * 40$$

内部低频时钟 (Internal low frequency clock, LIRC) 出厂时已校准到 256 kHz。芯片之间的频率变化典型值 < ±6% @2.5 – 5.5V/25°C，温度变化典型值 < ±2% @$^{-}$40 – $^{+}$105 °C。

同样可在烧录器软件选择是否需要测量 LIRC 精度，以及将 LIRC 频率误差存储到数据 EEPROM 的倒数第二个字节。每一个 step 代表 6% / 128 = 0.047%的误差。

LIRC 和 HIRC 可相互交叉校准 – 在一个 LIRC 周期内(值由“LFMOD”设置) 使用 Timer2 来测量指令时钟数(SysClk 选择 16MHz HIRC)，此为内置硬件功能。由于 LIRC 温度系数较低，因此当温度不稳定时，可通过用 LIRC 来校准 HIRC 的功能，以达到相同的±2%的温度系数。

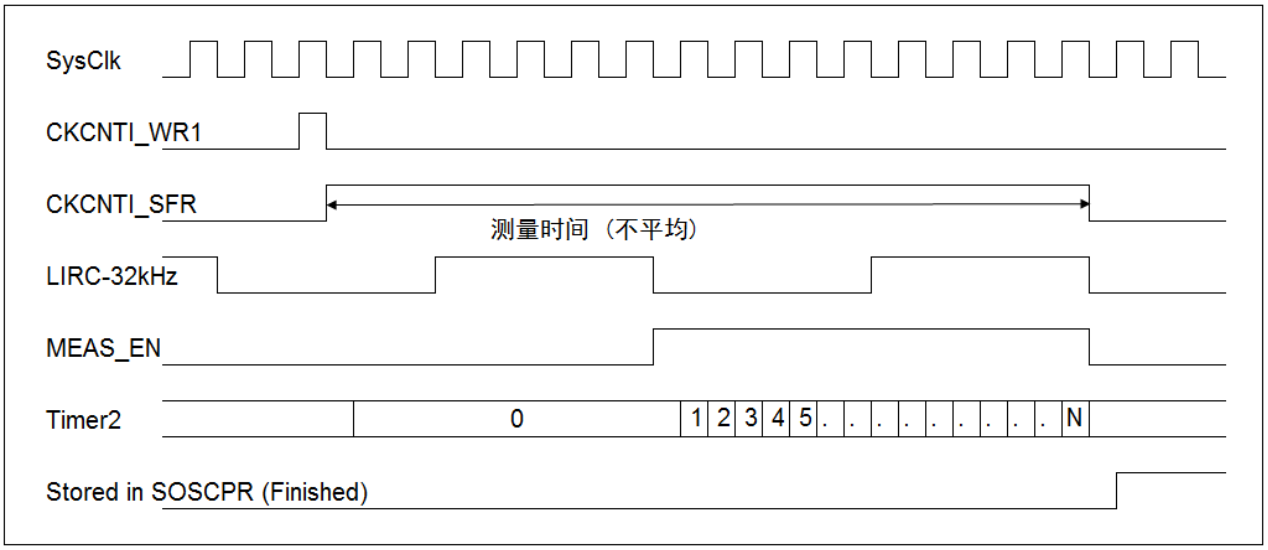

图 1-2 单次测量时序图

LIRC 和 HIRC 交叉校准步骤：

1. 设置 IRCF = 111, SCS = 1 ; SysClk 选择 16MHz HIRC (其他频率设置的精准度会降低)
2. 设置 CKMAVG = 1 ; 4 次测量平均，选择 0 表示不做平均
3. 设置 TMR2ON = 1 ; 使能 Timer2
4. 设置 CKCNTI = 1 ; 开始校准，默认 Timer2 预分频比 = 1，后分频比 = 1, T2CKSRC = SysClk for 2T; SysClk/2 for 4T
5. 校准完成时，CKCNTI 自动清零(“CKCNTI =0”), CKMIF 自动置位(“CKMIF = 1”)。
6. 测量值存储在 SOSCPR 寄存器中。
7. 如果 LIRC 为 32kHz，且 CPU 运行在 16MHz / 2T 下，则理想的匹配值为 500。

注：

- LIRC 和 HIRC 交叉校准时，不要对 SOSCPRH/L 寄存器进行写操作；
- LIRC 和 HIRC 交叉校准时，Timer2 不能被其他外设使用；

LIRC 和 HIRC 交叉校准功能与 IDE 的单步调试模式不兼容；

2. 应用范例

```
//*************************************************************
/* 文件名：test_61F13x_MSCK.c
* 功能：     FT61F13x MSCK 功能演示
*  IC:        FT61F135 SOP20
* 晶振：     16M/2T
* 说明：     程序中读取快时钟测量慢时钟数据
*
*                  FT61F135   SOP20
*                    ----------
*  VDD----------|1(VDD)   (VSS)20|----------VSS
*  NC-----------|2(PC1)    (PA0)19|----------NC
*  NC-----------|3(PC0)    (PA1)18|----------NC
*  NC-----------|4(PB7)    (PA2)17|----------NC
*  NC-----------|5(PB6)    (PA3)16|----------NC
*  NC-----------|6(PB5)    (PA4)15|----------NC
*  NC-----------|7(PB4)    (PA5)14|----------NC
*  NC-----------|8(PB3)    (PA6)13|----------NC
*  NC-----------|9(PB2)    (PA7)12|----------NC
*  NC-----------|10(PB1)   (PB0)11|----------NC
*                       ----------
*/
//*************************************************************************************
#include "SYSCFG.h"
//=========================================================
//DEFINE
//=========================================================
#define     unchar       unsigned char
#define     unint        unsigned int
volatile    unint        TestBuff;
/*-----------------------------------------------
 *  函数名：POWER_INITIAL
 *  功能：   上电系统初始化
 *  输入：   无
 *  输出：   无
 -----------------------------------------------*/
void POWER_INITIAL (void)
{
    OSCCON = 0B01110001;            //IRCF=111=16MHz/2T=8MHz，0.125μs
    OPTION = 0B00001000;            //Bit3=1 WDT，Bit[2:0]=000=WDT RATE 1:1
    INTCON = 0;                     //暂禁止所有中断
    PORTA = 0B00000000;
    TRISA = 0B00000000;             //PA 输入输出 0-输出 1-输入
    PORTC = 0B00000000;
```

```
    TRISC = 0B00000000;                 //PC 输入输出  0-输出  1-输入
    WPUA = 0B00000000;                  //PA 端口上拉控制  1-开上拉  0-关上拉
    WPUC = 0B00000000;                  //PC 端口上拉控制  1-开上拉  0-关上拉

    MSCON0 = 0B00000000;
    //Bit5: 低功耗模式。  0-关闭  1-使能
    //Bit4: 内部时钟输出 CLKO 功能输出引脚控制位。  0-映射到 PB0   1-映射到 PA2
    //Bit3: UCFG1<1:0>为 01 时此位有意义。  0-关闭 LVR 1-使能 LVR
    //Bit2: 快时钟测量慢周期的平均模式。  0-关闭  1-使能
    //Bit1：快时钟测量慢周期启动位。  0-关闭  1-启动
    //Bit0: 睡眠时 T2CK 保持工作。  0-NO 1-YES
}
/*-------------------------------------------------
 *   函数名：DelayUs
 *   功能：   短延时函数
 *   输入：   Time  延时时间长度  延时时长 Time μs
 *   输出：   无
 -------------------------------------------------*/
void DelayUs(unsigned char Time)
{
    unsigned char a;
    for(a=0;a<Time;a++)
    {
        NOP();
    }
}
/*-------------------------------------------------
 *   函数名：DelayMs
 *   功能：   短延时函数--16M-2T--大概快 1%左右.
 *   输入：   Time  延时时间长度  延时时长 Time ms
 *   输出：   无
 -------------------------------------------------*/
void DelayMs(unsigned char Time)
{
    unsigned char a,b;
    for(a=0;a<Time;a++)
    {
        for(b=0;b<5;b++)
        {
            DelayUs(197);                   //快 1%
        }
    }
}
/*-------------------------------------------------
```

```
 *   函数名：SlowTimeTest
 *   功能：   快时钟测量慢时钟
 *   输入：   无
 *   输出：   慢时钟时钟测量值 TestTime
              不开平均模式慢时钟频率=16M/TestTime(2T)
              开平均模式慢时钟频率=16M/TestTime/4(2T)
 ------------------------------------------------*/
unint SlowTimeTest()
{
    unint TestTime;
    TMR2ON = 1;                 //开定时器 2
    CKMIF = 0;                  //清零标志位
    CKMAVG = 0;                 //关闭平均模式
                                //打开平均模式输出数据为四个周期的时钟数(单周期*4)
    CKCNTI = 1;                 //使能快时钟测量位,开始测量
    while(!CKMIF);
    CKMIF = 0;                  //清零标志位
    TestTime = SOSCPRH << 8;
    TestTime = TestTime + SOSCPRL;
    return TestTime;
}
/*-----------------------------------------------
 *   函数名:   main
 *   功能：   主函数
 *   输入：   无
 *   输出：   无
 ------------------------------------------------*/
void main(void)
{
    POWER_INITIAL();            //系统初始化

    while(1)
    {

        TestBuff = SlowTimeTest();  //时钟测量值
                                    //32768 该数值≈488(不开平均模式-单周期)
                                    //慢时钟= TestBuff/16(KHz)
        NOP();
        DelayMs(200);               //延时 200ms
    }
}
```

联系信息

Fremont Micro Devices Corporation

#5−8, 10/F, Changhong Building
Ke-Ji Nan 12 Road, Nanshan District,
Shenzhen, Guangdong, PRC 518057

Tel: (+86 755) 8611 7811
Fax: (+86 755) 8611 7810

Fremont Micro Devices (HK) Limited

#16, 16/F, Block B, Veristrong Industrial Centre,
34−36 Au Pui Wan Street, Fotan, Shatin, Hong Kong SAR

Tel: (+852) 2781 1186
Fax: (+852) 2781 1144

http://www.fremontmicro.com