FT62F08X

Application note

目录

1. 基本定时器 TIM4 3
1.1. 特性 3
1.2. 原理框图 3
1.3. Timer4 时钟源 3
1.4. 预分频器 4
1.5. TIM4 中断 4
1.6. Timer4 寄存器表 4
1.6.1. TIM4CR1，地址 0x112 5
1.6.2. TIM4IER，地址 0x113 5
1.6.3. TIM4SR，地址 0x114 6
1.6.4. TIM4EGR，地址 0x115 6
1.6.5. TIM4CNTR，地址 0x116 6
1.6.6. TIM4PSCR，地址 0x117 7
1.6.7. TIM4ARR，地址 0x118 7
2 应用范例 8

FT62F08x Timer4 的应用

1. 基本定时器 TIM4

1.1. 特性

- 8bit 自动重载向上计数器
- 计数时钟可编程预分频
- 中断
  - 计数器溢出

1.2. 原理框图

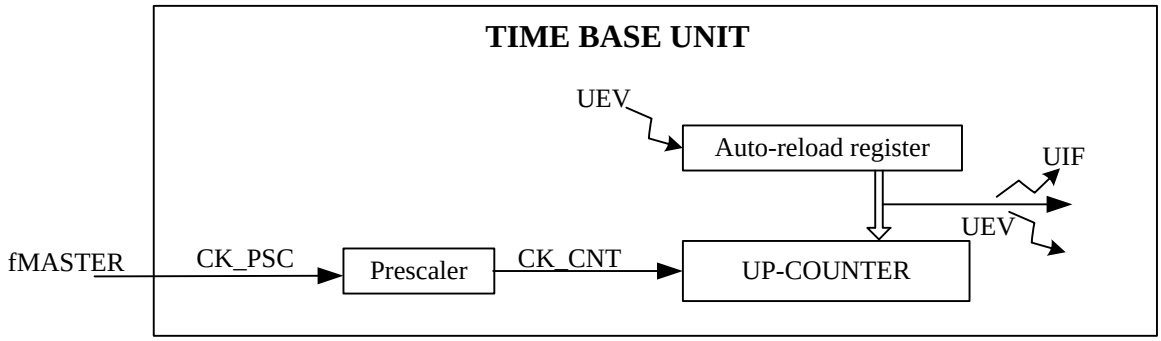

图 12.1 Timer4 原理框图

1.3. Timer4 时钟源

Timer4 有 4 种时钟源可选，由寄存器位 T4CKS 设置。在 Timer4 的被使能（PCKEN.TIM4EN=1）的情况下，所选择的时钟源被自动使能。

注意：

1. 如果要选择 LP 晶体时钟，系统时钟配置寄存器位 FOSC 必须选择 LP 模式，否则对应的时钟源将不被使能；

2. 同理，如果要选择 XT 晶体时钟，系统时钟配置寄存器位 FOSC 必须选择 XT 模式，否则对应的时钟源将不被使能；

SLEEP 模式下，如果 SYSON 为 1，且 TIM4EN=1，则所选择的时钟源将保持振荡，Timer4 将继续工作；否则，所选的时钟源取决于其他模块的设置情况。

1.4. 预分频器

计数时钟可以进行 3bit 的时钟预分频：

$f_{CK\_CNT} = f_{CK\_PSC}/2^{(PSCR[2:0])}$

预分频支持分频自动更新，即在更新事件发生后，能够自动改变预分频值。当 T4CEN 为 0 时，写入预分频寄存器的值也能直接加载实际应用的预分频寄存器中。

1.5. TIM4 中断

Timer4 只有一个中断请求源：

- 更新中断(计数器上溢或计数器初始化)

在用这些中断之前需要提前打开 TIM4IER 寄存器中的中断使能位(T4UIE)。

不同的中断源还可以配置通过 TIM4EGR 寄存器来产生(软件产生中断 T4UG)。

1.6. Timer4 寄存器表

名称	地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	复位值
TIM4CR1	0x112	T4ARPE	—	T4CKS[1:0]		T4OPM	T4URS	T4UDIS	T4CEN	0-00 0000
TIM4IER	0x113	—	—	—	—	—	—	—	T4UIE	---- ---0
TIM4SR	0x114	—	—	—	—	—	—	—	T4UIF	---- ---0
TIM4EGR	0x115	—	—	—	—	—	—	—	T4UG	---- ---0
TIM4CNTR	0x116	T4CNT[7:0]								0000 0000
TIM4PSCR	0x117	—	—	—	—	—	T4PSC[2:0]			---- -000
TIM4ARR	0x118	T4ARR[7:0]								0000 0000

1.6.1. TIM4CR1，地址 0x112

Bit	7	6	5	4	3	2	1	0
Name	T4ARPE	reversed	T4CKS[1:0]		T4OPM	T4URS	T4UDIS	T4CEN
Reset	0	—	0	0	0	0	0	0
Type	RW	RO-0	RO	RO	RW	RW	RW	RW

7	**T4ARPE**：自动预装载允许位 0：TIM1_ARR寄存器没有缓冲，它可以被直接写入； 1：TIM1_ARR 寄存器由预装载缓冲器缓冲。
6	保留位，读0
5:4	**T4CKS**：Timer4时钟选择位 00：系统时钟/主时钟 01：内部快时钟HIRC 10：LP时钟，只有当FOSC选择LP模式时才有意义 11：XT时钟，只有当FOSC选择XT模式时才有意义
3	**T4OPM**：单脉冲模式 0：在发生更新事件时，计数器不停止； 1：在发生下一次更新事件(清除CEN位)时，计数器停止。
2	**T4URS**：更新请求源 0：如果UDIS允许产生更新事件，则下述任一事件产生一个更新中断： 寄存器被更新(计数器上溢/下溢) 软件设置UG位 时钟/触发控制器产生的更新 1：如果UDIS允许产生更新事件，则只有当下列事件发生时才产生更新中断，并UIF置1： 寄存器被更新(计数器上溢/下溢)
1	**T4UDIS**：禁止更新 0：一旦下列事件发生，产生更新(UEV)事件： 计数器溢出/下溢 产生软件更新事件 时钟/触发模式控制器产生的硬件复位被缓存的寄存器被装入它们的预装载值。 1：不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了UG位或时钟/触发控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
0	**T4CEN**：允许计数器 0：禁止计数器； 1：使能计数器。 注：在软件设置了CEN位后，外部时钟、门控模式和编码器模式才能工作。然而触发模式可以自动地通过硬件设置CEN位。

1.6.2. TIM4IER，地址 0x113

Bit	7	6	5	4	3	2	1	0
Name	reversed							T4UIE
Reset	—	—	—	—	—	—	—	0

Type	RO-0	RO-0	RO-0	RO-0	RO-0	RO-0	RO-0	RW
0	**T4UIE**：允许更新中断 0：禁止更新中断； 1：允许更新中断。							

1.6.3. TIM4SR，地址 0x114

Bit	7	6	5	4	3	2	1	0
Name	reversed							T4UIF
Reset	—	—	—	—	—	—	—	0
Type	RO-0	RO-0	RO-0	RO-0	RO-0	RO-0	RO-0	RW-0
0	**T4UIF**：更新中断标记 当产生更新事件时该位由硬件置1。它由软件清0。 0：无更新事件产生； 1：更新事件等待响应。当寄存器被更新时该位由硬件置1： 若TIM1_CR1寄存器的UDIS=0，当计数器上溢或下溢时； 若TIM1_CR1寄存器的UDIS=0、URS=0，当设置TIM1_EGR寄存器的UG位软件对计数器CNT重新初始化时； 若TIM1_CR1寄存器的UDIS=0、URS=0，当计数器CNT被触发事件重新初始化时 (参考0从模式控制寄存器TIM1_SMCR)。							

1.6.4. TIM4EGR，地址 0x115

Bit	7	6	5	4	3	2	1	0
Name	reversed							T4UG
Reset	—	—	—	—	—	—	—	0
Type	RO-0	RO-0	RO-0	RO-0	RO-0	RO-0	RO-0	RW
0	**T4UG**：产生更新事件 该位由软件置1，由硬件自动清0。 0：无动作； 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清0(但是预分频系数不变)。若在中心对称模式下或DIR=0(向上计数)则计数器被清0；若DIR=1(向下计数)则计数器取TIM1_ARR的值。							

1.6.5. TIM4CNTR，地址 0x116

Bit	7	6	5	4	3	2	1	0
Name	T4CNT[7:0]							
Reset	0	0	0	0	0	0	0	0

Type	RW	RW	RW	RW	RW	RW	RW	RW
7:0	**T4CNT[7:0]：**计数器的8位值							

1.6.6. TIM4PSCR，地址 0x117

Bit	7	6	5	4	3	2	1	0
Name	reversed				T4PSC[3:0]			
Reset	—	—	—	—	0	0	0	0
Type	RO-0	RO-0	RO-0	RO-0	RW	RW	RW	RW
3:0	**T4PSC[3:0]：**预分频器的值 预分频器对输入的CK_PSC时钟进行分频。 计数器的时钟频率f_{CK_CNT}等于$f_{CK_PSC}/2^{(PSC[3:0])}$。PSC[7:4]由硬件清0。 PSCR包含了当更新事件产生时装入当前预分频器寄存器的值(包括由于清除TIMx_EGR寄存器的UG位产生的计数器清除事件)。这意味着如要新的预分频值生效，必须产生更新事件或者CEN=0。							

1.6.7. TIM4ARR，地址 0x118

Bit	7	6	5	4	3	2	1	0
Name	T4ARR[7:0]							
Reset	0	0	0	0	0	0	0	0
Type	RW	RW	RW	RW	RW	RW	RW	RW
7:0	**T4ARR[7：0]:** 自动重装载的8位值 ARR包含了将要装载入实际的自动重装载寄存器的值。 当自动重装载的值为空时，计数器不工作。							

2 应用范例

```
//************************************************************
/*   文件名: TEST_62F08x_TIM4.c
*    功能：    FT62F08x-TIM4 功能演示
*    IC:      FT62F088 LQFP32
*    内部：     16M
*    empno: 500
*    说明：    程序通过 TIM4 中断在 RB3 输出频率为 4K 的方波
*
*
*
*
*    参考原理图  TEST_62F08x_sch.pdf
*/
//************************************************************
#include "SYSCFG.h"
//************************************************************
//************************宏定义*********************************
#define   unchar        unsigned char
#define   unint         unsigned int
#define   unlong        unsigned long

#define   DemoPortOut  RB3
#define   DemoPortIn          RC3

//volatile    unchar mydata; //全局查看变量定义
/*---------------------------------------------
 *   函数名：中断
 *   功能：
 *   输入：    无
 *   输出：    无
 ---------------------------------------------*/
void interrupt ISR(void)                    //PIC_HI-TECH 使用
{
    //定时器 4 的中断处理**********************
    if(T4UIE && T4UIF)
    {
        T4UIF = 1;                                  //写 1 清零标志位
        DemoPortOut = ~DemoPortOut;   //翻转电平
    }
}
/*---------------------------------------------
```

```
 *  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;    //16MHZ 1:1
    //BIT7~BIT4: 主时钟（系统时钟）分频比选择。0111(1:1),0110(1:2),0101(1:4),0100(1:8),0011(1:16),0010,(1:32),0001(1:64),1xxx(1:128),0000(32kHz LIRC)
    //BIT3:振荡器起振超时状态位。1：器件运行在 FOSC<2:0>指定的外部时钟之下；0：器件运行在内部振荡器之下
    //BIT2:高速内部时钟状态。 1：HIRC is ready；0：HIRC is not ready
    //Bit1：低速内部时钟状态。1：LIRC is ready；0：LIRC is not ready
    //Bit0:系统时钟选择位。1：系统时钟选择为内部振荡器；0：时钟源由 FOSC<2:0>决定


    INTCON = 0;                 //暂禁止所有中断

    PORTA = 0B00000000;
    TRISA = 0B11111110;         //PA 输入输出  0-输出  1-输入
    PORTB = 0B00000000;
    TRISB = 0B11110111;         //PB 输入输出  0-输出  1-输入
    PORTC = 0B00000000;
    TRISC = 0B11111110;         //PC 输入输出  0-输出  1-输入
    PORTD = 0B00000000;
    TRISD = 0B11111111;         //PD 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
    WPUB = 0B00000000;          //PB 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
    WPUD = 0B00000000;          //PD 端口上拉控制  1-开上拉  0-关上拉

    WPDA = 0B00000000;          //PA 端口上拉控制  1-开下拉  0-关下拉
    WPDB = 0B00000000;          //PB 端口上拉控制  1-开下拉  0-关下拉
    WPDC = 0B00000000;          //PC 端口上拉控制  1-开下拉  0-关下拉
    WPDD = 0B00000000;          //PD 端口上拉控制  1-开下拉  0-关下拉

    PSRC0   = 0B11111111;       //PORTA,PORTB 源电流设置最大
    //BIT7~BIT6:PORTB[7:4]源电流能力控制,BIT5~BIT4:PORTB[3:0]源电流能力控制
    //BIT3~BIT2:PORTA[7:4]源电流能力控制,BIT1~BIT0:PORTA[3:0]源电流能力控制
```

```
    PSRC1   = 0B11111111;       //PORTC,PORTD 源电流设置最大
    //BIT7~BIT6:PORTD[7:4]源电流能力控制,BIT5~BIT4:PORTD[3:0]源电流能力控制
    //BIT3~BIT2:PORTC[7:4]源电流能力控制,BIT1~BIT0:PORTC[3:0]源电流能力控制

    PSINK0 = 0B11111111;        //PORTA 灌电流设置最大 0:最小，1:最大
    PSINK1 = 0B11111111;        //PORTB 灌电流设置最大 0:最小，1:最大
    PSINK2 = 0B11111111; //PORTC 灌电流设置最大 0:最小，1:最大
    PSINK3 = 0B11111111; //PORTD 灌电流设置最大 0:最小，1:最大

    ANSELA = 0B00000000;       //全为数字管脚

}
/*------------------------------------------------
 *  函数名称：DelayUs
 *  功能：      短延时函数 --16M-2T--大概快 1%左右.
 *  输入参数：Time 延时时间长度 延时时长 Time Us
 *  返回参数：无
 ------------------------------------------------*/
void DelayUs(unsigned char Time)
{
    unsigned char a;
    for(a=0;a<Time;a++)
    {
        NOP();
    }
}
/*------------------------------------------------
 *  函数名称：DelayMs
 *  功能：      短延时函数
 *  输入参数：Time 延时时间长度 延时时长 Time ms
 *  返回参数：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%
        }
    }
}
```

```
/*---------------------------------------------
 *   函数名称：DelayS
 *   功能：      短延时函数
 *   输入参数：Time 延时时间长度  延时时长 Time S
 *   返回参数：无
 ---------------------------------------------*/
void DelayS(unsigned char Time)
{
    unsigned char a,b;
    for(a=0;a<Time;a++)
    {
        for(b=0;b<10;b++)
        {
            DelayMs(100);
        }
    }
}

/*---------------------------------------------
 *   函数名称：Time4Initial
 *   功能：
 *   输入参数：
 *   返回参数：无
 ---------------------------------------------*/
void Time4Initial(void)
{
    PCKEN |=0B00001000;         //TIME4 模块时钟使能
    //CKOCON=0B00110000;
    //TCKSRC=0B00000011;

    TIM4CR1     =0B00000101;
    //BIT7: 0：TIM1_ARR 寄存器没有缓冲，它可以被直接写入; 1：TIM1_ARR 寄存器由预装载缓冲器缓冲。
    //BIT6:保留
    //BIT5~BIT4:timer4 时钟选择位。
                //00：系统时钟/主时钟
                //01：内部快时钟 HIRC
                //10：LP 时钟，只有当 FOSC 选择 LP 模式时才有意义
                //11：XT 时钟，只有当 FOSC 选择 XT 模式时才有意义

    //BIT3:
    //            0：在发生更新事件时，计数器不停止；
    //            1：在发生下一次更新事件(清除 CEN 位)时，计数器停止。
```

//BIT2:
// 0：如果 UDIS 允许产生更新事件，则下述任一事件产生一个更新中断：
//寄存器被更新(计数器上溢/下溢)
//软件设置 UG 位
//时钟/触发控制器产生的更新
// 1：如果 UDIS 允许产生更新事件，则只有当下列事件发生时才产生更新中断，并 UIF 置 1：
//寄存器被更新(计数器上溢/下溢)

//BIT1:
// 0：一旦下列事件发生，产生更新(UEV)事件：
//计数器溢出/下溢
//产生软件更新事件
//时钟/触发模式控制器产生的硬件复位被缓存的寄存器被装入它们的预装载值。
// 1：不产生更新事件，影子寄存器(ARR、PSC、CCRx)保持它们的值。如果设置了 UG 位或时钟/触发控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。

// BIT0: 0：禁止计数器；1：使能计数器。

TIM4IER=0B00000001;
//BIT0: 0：禁止更新中断；1：允许更新中断。

TIM4ISR=0B00000000;
//BIT0:当产生更新事件时该位由硬件置 1。它由软件写 1 清 0
//0：无更新事件产生；
//1：更新事件等待响应。当寄存器被更新时该位由硬件置 1：
//若 TIM4_CR1 寄存器的 UDIS=0，当计数器上溢或下溢时；
//若 TIM4_CR1 寄存器的 UDIS=0、URS=0，当设置 TIM4_EGR 寄存器的 UG 位软件对计数器
//CNT 重新初始化时；
//若 TIM4_CR1 寄存器的 UDIS=0、URS=0，当计数器 CNT 被触发事件重新初始化时。

TIM4EGR =0B00000000;
//BIT0:该位由软件置 1，由硬件自动清 0。
//0：无动作；
//1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清 0(但是预分频系数不变)。若在中心对称模式下或 DIR=0(向上计数)则计数器被清 0；若 DIR=1(向下计数)则计数器取 TIM1_ARR 的值。

```
    TIM4CNTR=0; //TIM4 8 位计数器

    TIM4PSCR=0B00000100;
    //预分频器对输入的 CK_PSC 时钟进行分频。
    //计数器的时钟频率 fCK_CNT 等于 fCK_PSC/2(PSC[2:0])。PSC[7:3]由硬件清 0。
    //PSCR 包含了当更新事件产生时装入当前预分频器寄存器的值(包括由于清除 TIMx_EGR 寄存器的 UG 位产生的计数器清除事件)。这意味着如要新的预分频值生效，必须产生更新事件或者 CEN=0。

    TIM4ARR     =124;
    //ARR 包含了将要装载入实际的自动重装载寄存器的值。
    //当自动重装载的值为空时，计数器不工作。

    INTCON |= 0B11000000;      //开总中断和外设中断
  }
/*------------------------------------------------
 *  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
    Time4Initial();

    while(1)
    {
        NOP();
    }
}
```

Fremont Micro Devices (SZ) Limited

#5-8, 10/F, Changhong Building, Ke-Ji Nan 12 Road, Nanshan District, Shenzhen, Guangdong 518057
Tel: (86 755) 86117811
Fax: (86 755) 86117810

Fremont Micro Devices (Hong Kong) Limited

#16, 16/F, Blk B, Veristrong Industrial Centre, 34-36 Au Pui Wan Street, Fotan, Shatin, Hong Kong
Tel: (852) 27811186
Fax: (852) 27811144

Fremont Micro Devices (USA), Inc.

42982 Osgood Road Fremont, CA 94539
Tel: (1-510) 668-1321
Fax: (1-510) 226-9918

Web Site: http://www.fremontmicro.com/