FT62F08X

Application note

FMD Confidential

© 2018Fremont Micro Devices Inc.

目录

FMD Confidential

FT62F08X IR Send 应用

1 IR 介绍

一个通用的红外遥控系统由发射和接收两大部分组成，如图 1 所示：

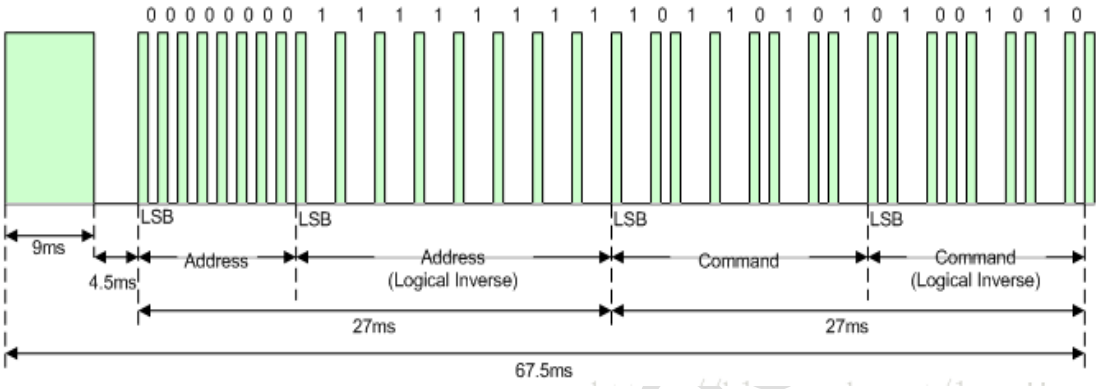

图 1

发射部分主要包括键盘矩阵、编码调制、红外发射管； 接收部分包括光、电信号的转换以及放大、解调、解码电路。

举例来说，通常我们家电遥控器信号的发射，就是将相应按键所对应的控制指令和系统码(由 0 和 1 组成的序列)，调制在 32~56kHz 范围内的载波上（目的为：抗干扰及低功率），然后经放大（接三极管）、驱动红外发射管（透明的头）将信号发射出去。

2 IR Send 相关寄存器的设置

本例使用两个定时器，一个是产生38KHz载波频率，另一个定时器是做时基，定时时长是560uS，红外信号的高低电平是560uS的整数倍。

定时器 2 为 16 位，配置成 13us 中断产生 38K 信号
定时器 4 为 8 位，配置成 560us 中断一次。

本讲解以IC FT62F08x LQFP32为示范，每2.5秒钟会发出一次信号，信号的码为IRData[4] = {0x00,0xff,0x40,0xBf}

FMD Confidential

3 应用范例

```
//************************************************************
/*  文件名： TEST_62F08x_IR_Send.c
*   功能：   FT62F08x_IR_Send 功能演示
*   IC:      FT62F088 LQFP32
*   内部：    16M
*   empno: 500
*   说明：  演示程序中,IR 红外是采用 6122 协议，起始信号是 9ms 低电平，到 4.5ms 高电平，再到低 8 位
*           用户识别码，到高 8 位的用户识别码，8 位数据码，8 位数据码的反码。SendIO（RB3）定时
*           （5 秒钟）发送一次，接收端收到遥控器发过来的数据后，校验数据互为补码，LED 会开关。
*
*   参考原理图 TEST_62F08x_sch.pdf
*/
//************************************************************
#include "SYSCFG.h"
// 宏定义
#define  uchar      unsigned char
#define  uint       unsigned int
#define  ulong      unsigned long

#define  IRSendIO          RB3                    // 红外的发送脚

#define IRSend_HIGH_1   1                         // 560uS
#define IRSend_LOW_1    3                         // 1680uS

#define IRSend_HIGH_0   1                         // 560uS
#define IRSend_LOW_0    1                         // 560uS

#define IRSend_PIN_1    T2UIE = 1                 //通过 PA4 输出载波
#define IRSend_PIN_0    T2UIE = 0                 //通过 PA4 输出载波

#define Status_NOSend 0                           // 不发送的状态
#define Status_Head       1                       // 发送引导码的状态
#define Status_Data       2                       // 发送数据的状态

uchar IRSendStatus;                               // 发送状态，是发送引导码还是数据
uchar IRSendData;                                 // 发送的数据中转变量
uchar  TxBit=0,TxTime=0;
uchar Sendbit = 0;
uchar level0,level1;                              // 一位数据里发送与关闭的时间值
bit SendLastBit = 0;
```

FMD Confidential

```
uchar SaveLastBit = 0;
uint SYSTime5S = 0;                                    // 系统时间，5S 发送一次

uchar IRData[4] = {0x00,0xff,0x40,0xBf};          // 需要发送的 4 个数据

//============================================================
//Funtion name：POWER_INITIAL
//parameters：无
//returned value：无
//说明：初始化单片机
//============================================================
void POWER_INITIAL(void)
{
    OSCCON = 0B01110001;     //WDT 32KHZ IRCF=111=16MHZ/2=8MHZ,0.125US/T
    PCKEN |=0B00001100;       //模块时钟使能
                              //Bit0=1,系统时钟为内部振荡器
                              //Bit0=0,时钟源由 FOSC<2：0>决定即编译选项时选择
    INTCON = 0;               //暂禁止所有中断

    PORTA = 0B00000000;
    TRISA = 0B11111111;       //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 = 0B11111111;       //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 = 0B00001000;        //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
```

FMD Confidential

```
    //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：TIMER4_INITIAL
 *  功能：初始化设置定时器 4
 *  相关寄存器：
 *  说明：560uS
 *  设置 TMR4 定时时长 560us=(1/16000000)*2*32*140(16M-2T-PSA 1:32- TMR0=255 溢出)
 -------------------------------------------------*/
void TIMER4_INITIAL (void)
{

    //CKOCON=0B00110000;
    //TCKSRC=0B00000011;

    TIM4CR1     =0B00000101;
    //BIT7: 0：TIM4_ARR 寄存器没有缓冲，它可以被直接写入; 1：TIM4_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：
```

FMD Confidential

//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=0B00000110;

//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。

FMD Confidential

```
    TIM4ARR     =140;
    //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。
}
/*---------------------------------------------
 * 函数名称：TIMER2_INITIAL
 * 功能：初始化设置定时器 2
 * 输入：无
 * 输出：无
 * 说明：用定时器 2 作为 38KHz 红外载波发生器，从 PA4 输出
 ---------------------------------------------*/
void TIMER2_INITIAL (void)
{
    CKOCON=0B00100000;
    TCKSRC=0B00110000;
    //BIT7 低频内振模式：1 = 256K  振荡频率模式,0 = 32K  振荡频率模式
    //BIT6~BIT4TIM2 时钟源选择位
        //值  时钟源
        //0   系统时钟/主时钟
        //1   HIRC
        //2   XT 时钟/外部时钟
        //3   HIRC 的 2 倍频
        //4   XT 时钟/外部时钟的 2 倍频
        //5   LIRC
        //6   LP 时钟/外部时钟
        //7   LP 时钟/外部时钟的 2 位频
    //BIT3:保留位
    //BIT2~BIT1:TIM2 时钟源选择位
        //值  时钟源
        //0   系统时钟/主时钟
        //1   HIRC
        //2   XT 时钟/外部时钟
        //3   HIRC 的 2 倍频
        //4   XT 时钟/外部时钟的 2 倍频
        //5   LIRC
        //6   LP 时钟/外部时钟
        //7   LP 时钟/外部时钟的 2 位频


    TIM2CR1 =0B10000101;  //预载允许，边沿对齐向上计数器，计数器使能
    //BIT7:自动预装载允许位
        //0：TIM2_ARR 寄存器没有缓冲，它可以被直接写入；
        //1：TIM2_ARR 寄存器由预装载缓冲器缓冲。
    //BIT6~BIT5:选择对齐模式
```

FMD Confidential

//00：边沿对齐模式。计数器依据方向位(DIR)向上或向下计数。

//01：中央对齐模式 1。计数器交替地向上和向下计数。配置为输出的通道(TIM2_CCMRx 寄存器中 CciS=00)的输出比较中断标志位，只在计数器向下计数时被置 1。

//10:中央对齐模式 2。计数器交替地向上和向下计数。配置为输出的通道(TIM2_CCMRx 寄存器中 CciS=00)的输出比较中断标志位，只在计数器向上计数时被置 1。

//11：中央对齐模式 3。计数器交替地向上和向下计数。配置为输出的通道(TIM2_CCMRx 寄存器中 CciS=00)的输出比较中断标志位，在计数器向上和向下计数时均被置 1。

//BIT4:方向

//0：计数器向上计数；

//1：计数器向下计数。

//BIT3: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。

//BIT2: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)

//BIT1: 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。

//BIT0 允许计数器

//0：禁止计数器；

//1：使能计数器。

TIM2IER =0B00000000;

//BIT7：允许刹车中断

//0：禁止刹车中断；

//1：允许刹车中断。

//BIT6：触发中断使能

//0：禁止触发中断；

//1：使能触发中断。

//BIT5：允许 COM 中断

//0：禁止 COM 中断；

//1：允许 COM 中断。

FMD Confidential

```
        //BIT4：允许捕获/比较 4 中断
            //0：禁止捕获/比较 4 中断；
            //1：允许捕获/比较 4 中断。
        //BIT3：允许捕获/比较 3 中断
            //0：禁止捕获/比较 3 中断；
            //1：允许捕获/比较 3 中断。
        //BIT2：允许捕获/比较 2 中断
            //0：禁止捕获/比较 2 中断；
            //1：允许捕获/比较 2 中断。
        //BIT1：允许捕获/比较 1 中断
            //0：禁止捕获/比较 1 中断；
            //1：允许捕获/比较 1 中断。
        //BIT0：允许更新中断
            //0：禁止更新中断；
            //1：允许更新中断。

    //TIM2ARRH =0x23;           //自动重载，周期
    //TIM2ARRL =0x00;


    TIM2ARRH =0x01;           //自动重载，周期
    TIM2ARRL =0xA0;

    INTCON = 0B11000000;      //开总中断和外设中断
}
/*---------------------------------------------
 *  函数名：  SendCtrl
 *  功能：    发送数据函数
 *  输入：    无
 *  输出：    无
 ---------------------------------------------*/
void SendCtrl(void)
{

    if (IRSendStatus == Status_NOSend)              // 不发送的状态
    {
        IRSend_PIN_0;
        Sendbit = 0;
        TxTime = 0;

    }
    else if (IRSendStatus == Status_Head)       // 发送引导码
    {
        TxTime++;
```

FMD Confidential

```
        if (TxTime < 17)                        // 发送 9mS 信号
        {
            IRSend_PIN_1;
        }
        else if (TxTime < 24)                   // 4.5mS 不发送
        {
            IRSend_PIN_0;
        }
        else
        {
            TxTime = 0;
            IRSendStatus = Status_Data;
        }
        IRSendData = IRData[0];
        TxBit = 0x01;
    }
    else if(IRSendStatus == Status_Data)        // 发送数据
    {
        if (IRSendData & TxBit)                 // 1，是 1:3 的时间
        {
            level1 = IRSend_HIGH_1;
            level0 = IRSend_LOW_1;
        }
        else                                    // 0，是 1:1 的时间
        {
            level1 = IRSend_HIGH_0;
            level0 = IRSend_LOW_0;
        }
        TxTime++;
        if (TxTime <= level1)                   // 发送信号
        {
            IRSend_PIN_1;
        }
        else if (TxTime <= (level0+level1))     // 不发送信号
        {
            IRSend_PIN_0;
        }
        else if (Sendbit < 4)                   // 发送 4 位数据未完成
        {
            TxTime = 1;
            IRSend_PIN_1;
            SaveLastBit = IRSendData & TxBit;
            TxBit <<= 1;
            if (TxBit == 0x00)                  // 发送完一个字节
```

FMD Confidential

```
            {
                TxBit = 0x01;
                Sendbit++;
                IRSendData = IRData[Sendbit];
                if (Sendbit > 3)                    // 最后一位要注意，因为发送完了还要有一
个脉冲
                {
                    SendLastBit = 1;
                }
            }
        }
        else                                        // 数据完成了，要补脉冲
        {
            if(SendLastBit)
            {
                TxTime++;
                if(SaveLastBit)
                {
                    if(TxTime < 3)
                    {
                        IRSend_PIN_0;
                    }
                    else if(TxTime < 4)
                    {
                        IRSend_PIN_1;
                    }
                    else
                    {
                        IRSend_PIN_0;
                        IRSendStatus = Status_NOSend;
                        IRSend_PIN_0;
                        SendLastBit = 0;
                        TxBit = 0;
                        TxTime = 0;
                    }
                }
                else
                {
                    if(TxTime < 5)
                    {
                        IRSend_PIN_0;
                    }
                    else if(TxTime < 6)
                    {
```

FMD Confidential

```
                    IRSend_PIN_1;
                }
                else
                {
                    IRSend_PIN_0;
                    IRSendStatus = Status_NOSend;
                    IRSend_PIN_0;
                    SendLastBit = 0;
                    TxBit = 0;
                    TxTime = 0;
                }
            }
        }
    }

}

/*-----------------------------------------------
 *  函数名：  interrupt ISR
 *  功能：    中断处理，包括定时器 0 中断和外部中断
 *  输入：    无
 *  输出：    无
 -----------------------------------------------*/
void interrupt ISR(void)
{
    //定时器 4 的中断处理**********************
    if(T4UIE  &&  T4UIF)                    //560US  中断一次  红外每一位都是  560uS  的倍数

    {
        T4UIF = 1;

        SendCtrl();
        SYSTime5S++;
    }
    //定时器 2 的中断处理**********************
    if(T2UIE && T2UIF)
    {
        T2UIF = 1;

        IRSendIO = ~IRSendIO;           //翻转电平    产生 38KHz
    }
}
```

FMD Confidential

```
/*-------------------------------------------------
 *   函数名：main
 *   功能：    主函数
 *   输入：    无
 *   输出：    无
 -------------------------------------------------*/
void main(void)
{
    POWER_INITIAL();
    TIMER4_INITIAL();
    TIMER2_INITIAL();
    GIE = 1;                                //开中断
    while(1)
    {
        if(SYSTime5S >5000)                 //定时 2.5S
        {
            SYSTime5S = 0;
            IRSendStatus = Status_Head;
        }

    }
}
//============================================================
```

FMD Confidential

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/

FMD Confidential

* Information furnished is believed to be accurate and reliable. However, Fremont Micro Devices, Incorporated (BVI) assumes no responsibility for the consequences of use of such information or for any infringement of patents of other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent rights of Fremont Micro Devices, Incorporated (BVI). Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. Fremont Micro Devices, Incorporated (BVI) products are not authorized for use as critical components in life support devices or systems without express written approval of Fremont Micro Devices, Incorporated (BVI). The FMD logo is a registered trademark of Fremont Micro Devices, Incorporated (BVI). All other names are the property of their respective own.