FT62F08X

IR_Receive Application note

目录

FT62F08x IR_Receive 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 1-1 所示：

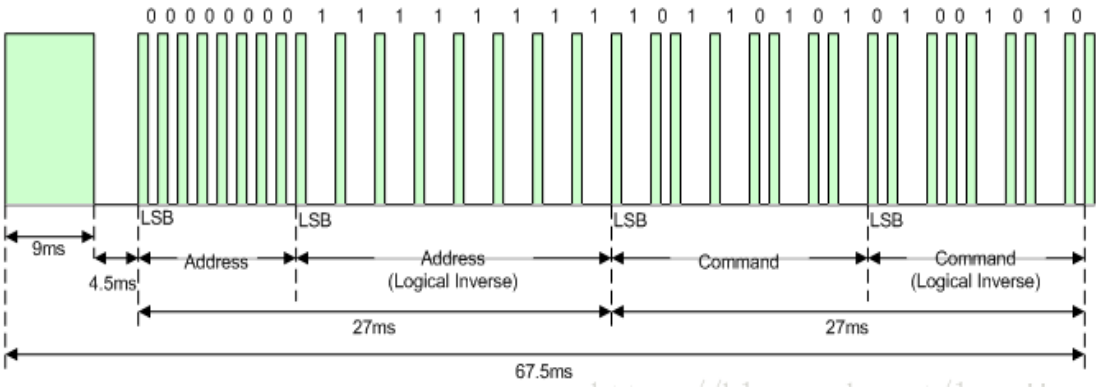

图 1-1

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。

本讲解以IC FT62F088 LQFP32为示范，采用一体的红外接收头，接收头输出脚连到MCU的IO口，IO口通过识别高低电平时间长短来解码，当收到的数据是合法的，指示LED的状态（开与关）会翻转一次。接收的IO口使用电平变化中断来识别信号，并使用定时器记录电平的时间长短。

2. 应用范例

```
//*************************************************************
/* 文件名：TEST_60F88x_IR_Receive.c
* 功能：    FT62F08X_IR_Receive 红外接收 功能演示
*  IC：     FT62F088 LQFP32
* 内部：    16M/2T
* 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。RXIO（RC3）每次收到遥控器发过来的数据后，
*           数据是合法（两对补码，不对内容判断）的话，LED(RB3)开关状态就改变一次。
*
*   参考原理图 TEST_62F08x_sch.pdf
*/
//*************************************************************
#include "SYSCFG.h"
//*********************宏定义*****************************
#define   uchar   unsigned char

#define   IRRIO   PC3             //IR 的接收脚
#define   LED     PB3             //LED 指示灯的 IO

uchar IRbitNum = 0;               //用于记录接收到第几位数据了
uchar IRbitTime = 0;              //用于计时一位的时间长短
volatile uchar IRDataTimer[4];    //存出来的 4 个数据
uchar bitdata = 0x01;             //用于按位或的位数据
uchar ReceiveFinish = 0;          //用于记录接收完成
uchar ReadAPin = 0;               //用于读取 IO 口状态，电平变化中断标志清除
volatile uchar rdata1,rdata2;
/*-----------------------------------------------
 * 函数名：POWER_INITIAL
 * 功能：   MCU 初始化函数
 * 输入：   无
 * 输出：   无
 -----------------------------------------------*/
void POWER_INITIAL(void)
{
    OSCCON = 0B01110001;        //IRCF=111=16MHz/2T=8MHz,0.125μs
    INTCON = 0;                 //暂禁止所有中断

    PORTA = 0B00000000;
    TRISA = 0B00000000;         //PA 输入输出  0-输出  1-输入
    PORTB = 0B00000000;
    TRISB = 0B00000000;         //PB 输入输出  0-输出  1-输入  PB3-输出
```

```
    PORTC = 0B00000000;
    TRISC = 0B00001000;          //PC 输入输出  0-输出  1-输入  PC3-输入
    PORTD = 0B00000000;
    TRISD = 0B00000000;          //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   PC3-上拉
    WPUD = 0B00000000;           //PD 端口上拉控制  1-开上拉  0-关上拉

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

    PSRC0 = 0B11111111;          //PORTA,PORTB 源电流设置最大
    PSRC1 = 0B11111111;          //PORTC,PORTD 源电流设置最大

    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：TIMER4_INITIAL
 * 功能：    初始化设置定时器
 * 设置 TMR4 定时时长 560μs
 ---------------------------------------------------*/
void TIMER4_INITIAL (void)
{
    PCKEN |= 0B00001000;         //TIME4 模块时钟使能

    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
    //Bit[5:4]: 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: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: 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。

TIM4SR = 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。

    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：Px_Level_Change_INITIAL
 * 功能：   端口电平变化中断初始化
 * 输入：   无
 * 输出：   无
---------------------------------------------*/
void Px_Level_Change_INITIAL(void)
{
    EPS0=0B10000000;            //选择 PC3 管脚中断
    //Px[0:3]中断管脚选择
    EPS1=0B00000000;
    //Px[4:7]中断管脚选择

    ITYPE0 = 0B11000000;        //双沿中断
    ITYPE1 = 0B00000000;

    EPIE0   = 0B00001000;       //使能中断 3

    INTCON = 0B01000000;
}

/*---------------------------------------------
 * 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)
    {
```

```
        T4UIF = 1;                  //写 1 清零标志位

        IRbitTime++;
        if(IRbitTime > 50)
        {
            T4UIE = 0;
            IRbitTime = 0;
        }
    }

    //Px 电平变化中断
    if(EPIF0 & 0x08)
    {
        EPIF0 |= 0x08;              //写 1 清零标志位

        if(IRRIO == 0)
        {
            T4UIE = 1;
            if(IRbitTime > 21)
            {
                IRDataTimer[0] = 0;
                IRDataTimer[1] = 0;
                IRDataTimer[2] = 0;
                IRDataTimer[3] = 0;
                IRbitNum = 0;
                bitdata = 0x00;
            }
            else if(IRbitTime > 3)
            {
                IRDataTimer[IRbitNum-1] |= bitdata;
            }
            IRbitTime = 0;
            bitdata<<=1;
            if(bitdata == 0)
            {
                bitdata = 0x01;
                IRbitNum++;
            }
            if(IRbitNum > 4)
            {
                IRbitNum = 0;
                T4UIE = 0;
                ReceiveFinish = 1;
            }
```

```
        }
    }
}
/*-------------------------------------------------
 * 函数名：main
 * 功能：   主函数
 * 输入：   无
 * 输出：   无
 -------------------------------------------------*/
void main(void)
{
    POWER_INITIAL();
    TIMER4_INITIAL();
    Px_Level_Change_INITIAL();
    GIE = 1;                        //开中断

    while(1)
    {
        if(ReceiveFinish==1)
        {
            ReceiveFinish = 0;
            rdata1 = 0xFF - IRDataTimer[0];
            rdata2 = 0xFF - IRDataTimer[2];
            if((rdata1 == IRDataTimer[1])&&(rdata2 == IRDataTimer[3]))
            {
                LED = ~LED;     //翻转电平
            }
        }
    }
}
```

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