Инфракрасный приемопередатчик с USB интерфейсом на микроконтроллере фирмы SiLabs

Олег Николайчук
onic@ch.moldpac.md

Схемотехника

Целью настоящей статьи является ознакомление читателей с устройством контроллера инфракрасного приемопередатчика с USB интерфейсом, предназначенного для получения диагностической информации с носимого контроллера и передачи ему команд управления от персонального компьютера.

В одной из опубликованных в нашем журнале статей уже рассказывалось о возможности передачи команд управления изолированному (носимому) контроллеру по инфракрасному каналу связи, например, от пульта дистанционного управления [1]. Инфракрасный канал позволяет передавать различную информацию между несколькими устройствами, например, между ведущим устройством управления и ведомым(и) изолированным(и) гальванически не связанным(и) контроллером (или несколькими контроллерами). Очевидно, что такая передача возможна на достаточно небольшом расстоянии – всего несколько метров. Однако этого вполне достаточно для осуществления некоторых задач, например, диагностики или контроля деятельности. При организации двухстороннего канала инфракрасной связи, от ведущего контроллера в сторону ведомого могут передаваться, например, различные команды, а от ведомого контроллера к ведущему – сообщения об исполнении тех или иных команд или о состоянии контроллера. Для организации двухстороннего канала инфракрасной связи в состав обоих контроллеров должны быть введены одинаковые инфракрасный приемник (например, TSOP-1436) и инфракрасный передатчик (например, инфракрасный светодиод TSAL-5100).

В рамках настоящей статьи мы рассмотрим только ведущий контроллер.

Принципиальная схема многофункционального контроллера M570 показана на рис.1. Контроллер M570 содержит собственно микроконтроллер D1 - C8051F320 фирмы Silicon Laboratories, к которому подключены инфракрасный интегрированный приемник A1 - TSOP-1436, инфракрасный передатчик VD1 - светодиод TSAL-5100, двухцветный индикаторный светодиод VD2 – L59-SRSG-C, слот для подключения к контроллеру мультимедиа карты (MMC – MultimediaCard) SL1 и разъем X1 – MiniUSB для подключения USB интерфейса. Кроме того, микроконтроллер имеет штырьковый разъем J1 для подключения адаптера программирования – отладки EC2 с интерфейсом JTAG/C2. Естественно, что также установлены необходимые для работы интерфейса JTAG/C2 резисторы и конденсаторы. Контроллер имеет небольшие размеры, определяемые, главным образом размерами слота для подключения MMC карты – 32x44 мм. Следует еще раз подчеркнуть, что приведена схема многофункционального контроллера, а описываемый в данной статье инфракрасный приемопередатчик является частным случаем, поэтому некоторые элементы принципиальной схемы могут не устанавливаться. В частности, в описываемом устройстве не будет использоваться мультимедиа карта MMC и слот для ее установки SL1.

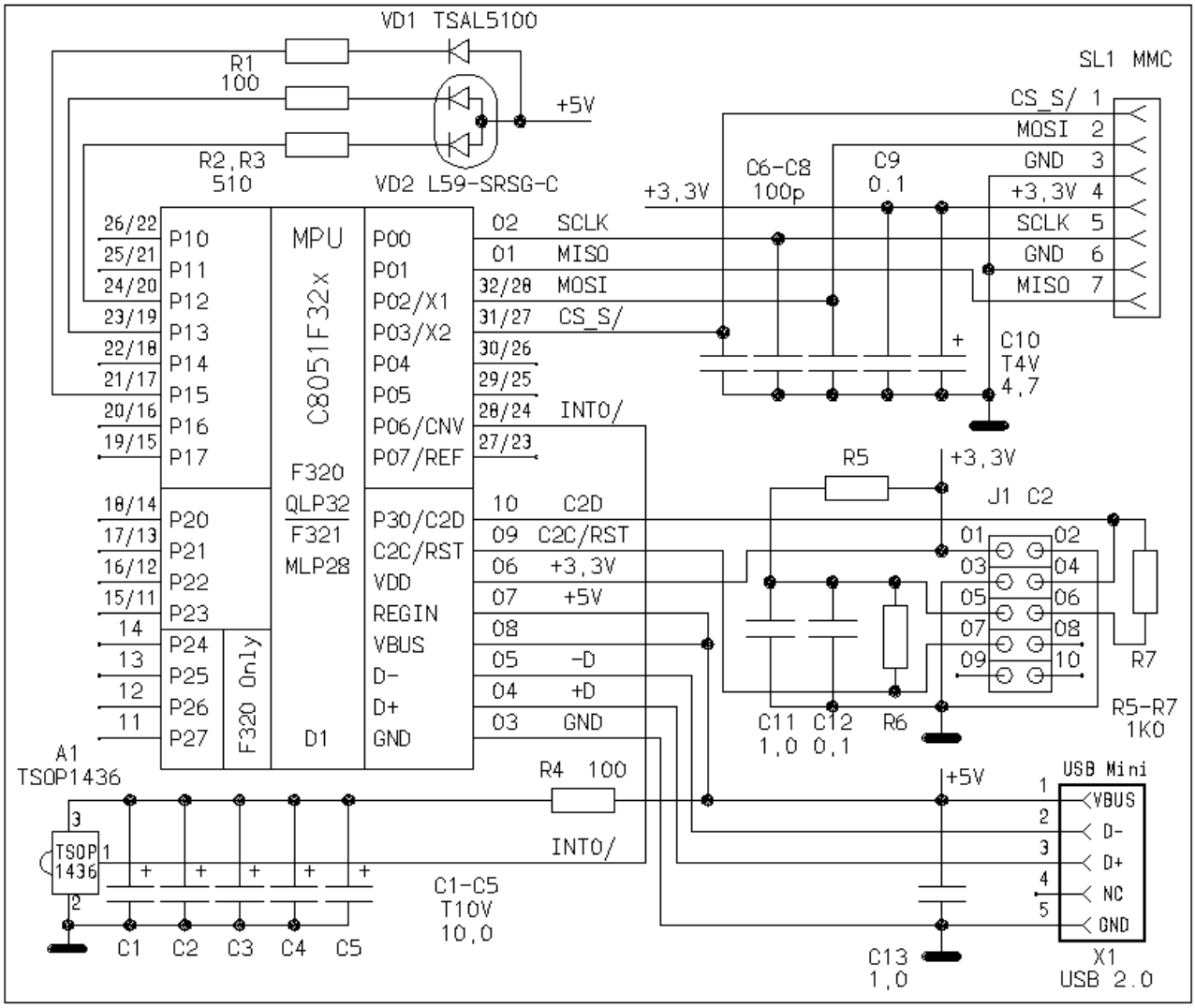

Рис.1. Многофункциональный USB контроллер M570

Приведенная принципиальная схема довольно проста и сборке. Однако для ее эксплуатации совместно с персональным компьютером, необходимо установить USB драйверы для данного устройства. Набор необходимых драйверов для работы с микроконтроллером C8051F320 поставляется в составе примеров совместно со средой программирования SiLabs IDE. При этом в описании драйвера(ов) - файле с расширением *.inf указаны данные производителя микросхемы, т.е. заполнены следующие информационные строки, определяющие параметры:

```
[DeviceList]
%DESCRIPTION%=DriverInstall,USB\VID_10C4&PID_0002
```

Параметр DeviceList содержит код производителя VID_10C4 и код продукта PID_0002 (код приводится после символа подчеркивания). Такие же величины должен выдавать контроллер в процессе его идентификации. Код производителя VID присваивается ассоциацией USB пользователей [2] всем производителям устройств, содержащих USB интерфейс. Список обладателей кодов производителей приведен, например, в [3]. Именно эти два номера, приводимые в строке файла описания DeviceList определяют, какие драйверы будут работать совместно с подключенным USB устройством.

Кроме этого, в файле описания драйвера(ов) приводятся текстовые строки Strings с описанием имени производителя MFGNAME, названия установочного источника INSTDISK, описанием прибора DESCRIPTION и короткого имени прибора FRIENDLYNAME, а также с названием группы приборов, к которым относится устройство S_DeviceClassDisplayName.

```
[Strings]
MFGNAME="Silabs Software"
INSTDISK="Silabs USBXpress Software Installation Disc"
DESCRIPTION="F32x USBXpress Device"
FRIENDLYNAME="USBXpress"

; device class display name, shown in Device Manager
S_DeviceClassDisplayName="USB I/O controlled devices"
```

Довольно часто разработчик стремиться заменить атрибуты производителя на свои личные. Для этого, во избежание конфликтов с оборудованием других фирм, необходимо выбрать свободный VID код и заменить его и в описании драйверов, и в тексте программы микроконтроллера. Также необходимо поступить и с параметрами MFGNAME, INSTDISK, DESCRIPTION и FRIENDLYNAME. Следует отметить, что совсем недавно на сайте производителя [4] появилась утилита CustomUSBDriverWizard.exe, позволяющая генерировать файлы описаний драйверов, а также производить их установку, причем пользователю предоставляется возможность самостоятельного задания кодов VID, PID и строковых описаний.

Теперь рассмотрим собственно программное обеспечение контроллера. Прежде всего отметим, что USB интерфейс мы будем использовать в режиме прерывания. В этом режиме по USB интерфейсу передаются пакеты по 64 байта 1 раз за миллисекунду.

Отметим, что для разработки программного обеспечения микроконтроллеров семейства C8051F32x (а также ряда их производных CP1201/1202/1203) производитель предоставляет очень удобный инструмент – библиотеку usb_api.lib, естественно с файлом описаний usb_api.h. Библиотека содержит всего восемь функций, которые позволяют осуществлять все возможные действия с интерфейсом USB [5,6]. Рассмотрим коротко функции этой библиотеки.

1. Функция void USB_Clock_Start (void) - разрешает использование встроенного тактового генератора и устанавливает все необходимые регистры для работы на частоте 48 МГц, необходимой для работы Full Speed USB.
2. Функция void USB_Init (UINT VendorID, UINT ProductID, BYTE *ManufacturerStr, BYTE *ProductStr, BYTE *SerialNumberStr, BYTE MaxPower, BYTE PwAttributes, UINT bcdDevice) – разрешает работу USB интерфейса, а также устанавливает все необходимые атрибуты устройства, которые передаются в функцию в качестве параметров:
   - *VendorID* — шестнадцатибитный код производителя VID (Vendor ID), о котором мы говорили несколько выше.
   - *ProductID* — шестнадцатибитный код продукта PID (Product ID).
   - *ManufacturerStr* — Строка, описывающая производителя – аналогична MFGNAME. Формат записи этой и остальных текстовых строк имеет следующею особенность – каждый символ строки описания должен быть обязательно дополнен кодом 0x00.
   - *ProductStr* — Строка, описывающая продукта – аналогична DESCRIPTION.
   - *SerialNumberStr* — строка, описывающая серийный номер.
   - *MaxPower* — Код, описывающий потребление прибора. Например, при потреблении в 100 мА следует указать код 0x32, а при максимальном потреблении до 500 мА – код 0xFA. Остальные величины описаны в [5].
   - *PwAttributes* — Код, описывающий способ питания устройства. Если устройство имеет автономное питание – необходимо установить биты от 1 до 6, а если устройство питается от USB шины - необходимо установить бит 0. См. Приложения с [5].
   - *BcdDevice* — Код версии устройства.
3. Функция UINT Block_Write (BYTE *Buffer, UINT NumBytes) – передает через USB интерфейс данные в буфер Buffer. Максимальное количество байтов задается величиной NumBytes.
4. Функция BYTE Block_Read (BYTE *Buffer, BYTE NumBytes) - читает из USB интерфейса в буфер Buffer до 64 байт, определяемых величиной NumBytes.
5. Функция BYTE Get_Interrupt_Source (void) – возвращает код причины API прерывания (код состояния интерфейса):
   - 0x00 - No USB API Interrupts – прерывания не произошли.
   - 0x01 - USB_RESET – Прерывание по сбросу USB.
   - 0x02 - TX_COMPLETE – Передача завершена.

- 0x04 - RX_COMPLETE – Прием завершен.
- 0x08 - FIFO_PURGE - USB FIFO буфер пуст (очищен).
- 0x10 - DEVICE_OPEN – Устройство открыто со стороны Host (ведущего, персонального компьютера).
- 0x20 - DEVICE_CLOSE – Устройство закрыто со стороны Host
- 0x40 - DEV_CONFIGURED – Устройство сконфигурировано.
- 0x80 - DEV_SUSPEND – Шина USB заторможена (приостановлена).

6. Функция void USB_Int_Enable (void) – разрешает API прерывания.
7. Функция void USB_Int_Disable (void) - запрещает API прерывания.
8. Функция void USB_Disable (void) – запрещает USB интерфейс.
9. Функция void USB_Suspend (void) – затормаживает USB прерывания.

Текст программы состоит из трех программных модулей, написанных на языке «С» фирмы Keil [7] и общего файла описаний. Рассмотрение начнем с главного модуля M570.C. В первой части приведены все необходимые определения переменных и атрибутов устройства:

```
// ************************************
// * M570     USB Interrupts Panel      *
// * with use lib USB_API ver.1.20      *
// * Create   by   O.Nicolaiciuc        *
// ************************************
#include "M570P.h"

code BYTE VER = 30; // Код версии программы
// ************************************
// Определения атрибутов устройства
// ************************************
#define MN_LEN sizeof("MY PRODUCTS")*2
code const BYTE MN_String[MN_LEN] =
{MN_LEN,0x03,'M',0,'Y',0,' ',0,'P',0,'R',0,'O',0,'D',0,'U',0,'C',0,'T',0,'S',0};

#define PR_LEN sizeof("M570")*2
code const BYTE PR_String[PR_LEN] =
{PR_LEN,0x03,'M',0,'5',0,'7',0,'0',0};

#define VR_LEN sizeof("M570_PANEL_V.30")*2
code const BYTE VR_String[VR_LEN] =
{VR_LEN,0x03,'M',0,'5',0,'7',0,'0',0,'_',0,'P',0,'A',0,'N',0,'E',0,'L',0,'_',0,'V',0,'.',0,'3',0,'0',0};
// ************************************
code const BYTE MaxPower = 0x10;     // Максимальный ток = 32 mA (16*2)
code const BYTE PwAttributes = 0x80;  // Питание от USB шины
code const UINT bcdDevice = 0x100;     // Номер release версии 1.00
// ************************************
#define OFRAME_SIZE          19 // Длина строки, передаваемой по ИК каналу
#define IFRAME_SIZE          19 // Длина строки, принимаемой по ИК каналу
// ************************************
xdata BYTE IFRAME   [IFRAME_SIZE+1]; // Последний фрейм от Host
xdata BYTE OFRAME [OFRAME_SIZE+1]; // Следующий фрейм для Host
// ************************************
extern xdata byte        IRI_Ready;       // Готовность ИК канала
extern xdata byte        IRI_Frame[20];  // Входной буфер ИК канала
extern xdata BYTE        IRO_Frame[20]; // Выходной буфер ИК канала
```

Прежде всего, отметим, что применяемые в настоящей программе типы данных UINT и BYTE определены в файле определений usb_api.h следующим образом:

```
#ifndef _UINT_DEF_              // UINT type definition
#define _UINT_DEF_
typedef unsigned int UINT;
#endif                          // _UINT_DEF_

#ifndef _BYTE_DEF_              // BYTE type definition
#define _BYTE_DEF_
typedef unsigned char BYTE;
#endif                          // _BYTE_DEF_
```

Далее рассмотрим главную функцию программы main(). Именно эта главная функция и реализует главный алгоритм работы программы.

В ее начале – этапе инициализации запрещается работа охранного таймера WDT, затем инициализируется USB интерфейс, включая и тактовый генератор, затем инициализируются Crossbar и порты, а затем устанавливается тактовая частота ядра, равная 24 МГц, после этого запускается охранный таймер. Далее включается последовательно через примерно 250 мс красный, зеленый и опять красный светодиоды и все выключаются. Далее разрешаются USB прерывания и с задержкой примерно в 3 секунды включается красный светодиод, и по USB интерфейсу передаются в Host (персональный компьютер) текстовые строки начальной инициализации, содержащие название устройства и версию программного обеспечения. Эти данные в Host заносятся в протокол обмена с устройством. В завершение этапа инициализации включается зеленый светодиод.

Далее следует основной цикл. В рамках этого цикла постоянно анализируется, имеется ли принятый ИК фрейм, а затем, имеется ли принятый USB фрейм. Если имеется принятый BR фрейм, то содержание принятой по ИК строки перезаписывается в исходящий буфер USB интерфейса и строка сразу же передается. Если имеется принятый USB фрейм, он анализируется в рамках функции Monitor() и при соблюдении всех требований передается в буфер ИК передатчика.

```
// ************************************
// Main Routine
// ************************************
void main (void)
{
int     i;

        PCA0MD &= ~0x40;    // Запрет охранного таймера WDT
        USB_Init (0x7777, 0x1516, MN_String, PR_String, VR_String,
                MaxPower, PwAttributes, bcdDevice);
        Port_Init();            // Инициализация Crossbar
        RM_Init();              // Инициализация ИК приемника
        RSTSRC |= 0x02;
        CLKSEL |= 0x02;         // Установить SYSCLK = 24 MHz
        WDT_Init();             // Запуск охранного таймера
BEGIN:
        Red();  Delay(250);     // Включить красный светодиод
        Green(); Delay(250);    // Включить зеленый светодиод
        Red();  Delay(120);     // Включить красный светодиод
        Black();                // Выключить светодилды
        USB_Int_Enable();       // Разрешить USB  прерывания
        Delay(3000);            // Пауза примерно 3 секунды
        Red();                  // Включить красный светодиод
                                // Строки инициализации в host
        Frame_Prep      (0," ");
        Frame_Prep      (0,"=*=");
        Frame_Prep      (0,"USB IR  Hardware");
        Frame_Prep      (0,"Board M570-PANEL");
        Frame_Prep      (0,"Firmware Ver. %02d",(unsigned) VER);
        Frame_Prep      (0,"==>");
```

```
        RST_Detect();           // Определяется источник сброса
        Green();                // Включается зеленый светодиод
        while (1)               // Основной цикл
        {
                if (IRI_Ready)  // Если принят ИК фрейм, т.е.
                {               // установлен флаг готовности
                        Red();  // Включить красный светодиод
                        // Копирование входного фрейма
                        for(i=0;i<19;i++) OFRAME[i]=IRI_Frame[i];
                        // Передача фрейма по USB
                        Block_Write (OFRAME, OFRAME_SIZE);
                        IRI_Ready=0;  // Очистить флаг готовности
                        Green(); // Включить зеленый светодиод
                }
                // Если функция Monitor() вернула «0» - сброс USB
                if (!Monitor())   goto BEGIN;
                Delay(100);
        }
}
```

Далее рассмотрим функции инициализации и функции общего назначения

```
void Port_Init (void)     // Инициализация портов
{
  P1MDIN   = 0x3F;    // Порт 1 выводы 7 и 6 устанавливаются как аналоговые входы
  P2MDIN   = 0x00;    // Порт 2 выводы 0:7 устанавливаются как аналоговые входы
  P0MDOUT= 0x0F;      // Порт 0 выводы 0:3 в ключевом режиме – интерфейс SPI
                      // Порт 0 выводы 4:5 – интерфейс UART в режиме открытого истока
                      // Порт 0 выводы 6:7 – прерывания 0&1 в режиме открытого истока
  P1MDOUT  = 0x00;  // Порт 1 выводы 0:5 в режиме открытого истока
  P0SKIP   = 0xC0;    // Порт 0 выводы 7:6 пропущены в crossbar для прерываний
  P1SKIP   = 0xC0;    // Порт 1 выводы 7:6 пропущены в crossbar для аналоговых входов
  P2SKIP   = 0xFF;    // Порт 2 пропущен в crossbar для аналоговых входов
  IT01CF  = 0x76;     // Порт 0 вывод 6 - прерывание 0, активный низкий уровень
                      // Порт 0 вывод 7 - прерывание 1, активный низкий уровень
  XBR0     = 0x03;    // Разрешить интерфейсы SPI(P00:P03) & UART(P04:P05)
  XBR1     = 0x40;    // Разрешить Crossbar
}
// ************************************
void WDT_Init (void)  // Инициализация охранного таймера
{
        PCA0MD  &= ~0x40;   // Запретить охранный таймер WDT
        PCA0MD   =  0x00;   // Источник SYSCLK/12
        PCA0CPL4 =  0xFF;   // Период WDT = 31 мс
        PCA0CPM4 =  0x40;   // Разрешить сравнение
        PCA0MD  |=  0x40;   // Разрешить охранный таймер WDT
}
// ************************************
void Time (unsigned mkS)        // Задержка mkS в микросекундах
{                               // Примерно соответствует при mkS > 10
        mkS *=2;
        while(mkS --)   WDT();
}
// ************************************
void WDT (void)                 // Перезапуск охранного таймера
{PCA0CPH4=0xFF;}
void Delay (unsigned mS)        // Задержка в миллисекундах mS
```

```
{while(mS --) {Time(1000);}}
void Green (void)                  // Включить зеленый светодиод
{RED=1; GREEN=0;}
void Red (void)                    // Включить красный светодиод
{RED=0; GREEN=1;}
void Black (void)                  // Выключить светодиоды
{RED=1; GREEN=1;}
```

Следующая функция вызывается при получении прерывания DEV_SUSPEND.

```
void Suspend_Device (void)
{
  // Здесь можно запретить все необходимые периферийные узлы
  USB_Suspend();                   // Вызвать функцию библиотеки API
  // Реинициализировать периферийного узлы
}
```

Функция USB прерывания

```
void USB_API_ISR (void) interrupt 16  // Обработка прерываний USB_API
{
        BYTE INTVAL = Get_Interrupt_Source ();
        if (INTVAL & RX_COMPLETE)
                {Block_Read(IFRAME,IFRAME_SIZE);}
        if (INTVAL & DEV_SUSPEND)
                {Suspend_Device();}
        if (INTVAL & DEV_CONFIGURED)
        { Port_Init();}
}
```

Далее приводятся функции обмена фреймами по USB интерфейсу

```
byte Input_CS (void) // Контрольная сумма входного фрейма
{
register byte CS=0, i;
        for (i=0;i<IFRAME_SIZE-1;i++) CS+=IFRAME[i]; return CS;
}
// ************************************
byte Output_CS (void) // Контрольная сумма выходного фрейма
{
register byte CS=0, i;
        for (i=0;i<OFRAME_SIZE-1;i++) CS+=OFRAME[i]; return CS;
}
// ************************************
// Подготовка и передача по USB выходного фрейма
void Frame_Prep (byte COM, char *fmt,...)
{
register byte i, LEN;

        GetFMT ();      // Макроопределение форматной строки
        WDT();          // Перезапустить охранный таймер
        memset (OFRAME,0,sizeof(OFRAME)); // Очистить выходной буфер
        OFRAME[0]=0xAA; // Записать преамбулу в 1й байт
        OFRAME[1]=COM; // Записать код команды во 2й байт
```

```
        LEN=strlen(fmtbuf);  // Определить длину форматной строки
        if  (LEN>16) LEN=16; // Если больше 16, ограничить
        for (i=0;i<LEN;i++) OFRAME[2+i]=fmtbuf[i]; // Скопировать
        OFRAME[OFRAME_SIZE-1]=Output_CS(); // Вычислить контрольную сумму
        WDT(); // Перезапустить охранный таймер
        Block_Write (OFRAME, OFRAME_SIZE); // Передать по USB
        Delay(600); // Задержка

}
// ************************************
// Получение и обработка USB фрейма
byte Monitor (void)
{
register byte i;
        if (IFRAME[0]!=0xAA) return 1; // Если нет преамбулы
        // Если не верна контрольная сумма
        if (IFRAME[IFRAME_SIZE-1]!=Input_CS())   return 2;
        // Если получен код сброса
        if (IFRAME[1]==0xFE) return 0;
        Red(); // Включить красный светодиод - передача
        // Копирование фрейма в ИК буфер
        for (i=0;i<IFRAME_SIZE;i++) IRO_Frame[i]=IFRAME[i];
        // Очистить USB буфер
        memset(IFRAME,0,sizeof(IFRAME));
        // Передать данные но ИК каналу
        RM_Frame_Send();
        // Включить красный светодиод
        Green();
        return 0xFF;
}
// ************************************
// Функция передает по USB причину последнего сброса
void RST_Detect (void)
{
BYTE  CH;

        CH=RSTSRC;
        switch (CH)
        {
                case 0x80:      Frame_Prep (0,"USB Reset      "); break;
                case 0x40:      Frame_Prep (0,"Flash Reset    "); break;
                case 0x20:      Frame_Prep (0,"Comp.0 Reset   "); break;
                case 0x10:      Frame_Prep (0,"Soft Reset     "); break;
                case 0x08:      Frame_Prep (0,"WDT Reset      "); break;
                case 0x04:      Frame_Prep (0,"Missing Reset  "); break;
                default:                                        break;
        }
}
```

Второй программный модуль RM_INP.C содержит набор функций, предназначенных для приема информации по инфракрасному интерфейсу. Более подробно приводимый ниже алгоритм описан в [1].

```
// ************************************
// RM_INP.C                 Version 02.00    *
// ************************************
```

```
#include "M570P.h"
// Формат входного фрейма по ИК интерфейсу
xdata byte IRI_Frame[20];
//IRx_Frame[0]=0xAA - преамбула
//IRx_Frame[1] - код команды
//IRx_Frame[2]-IR_Frame[17] - данные (16 байт)
//IRx_Frame[18] – контрольная сумма
xdata byte IRI_Ready; // Если!=0, входной фрейм готов
xdata unsigned int IRI_Bits[160]; // Массив входных битоы
// ************************************
// Функция инициализации приемника
void RM_Init (void)
{
   memset (IRI_Bits,0,sizeof(IRI_Bits)); // Очистка буфера
   EX0=1; // Разрешение входа
   IRI_Ready=0; // Очистка флага
}
// ************************************
// Подпрограмма прерывания 0 – осуществляет
// прием входной последовательности битов
void Ext0_INT (void) interrupt 0
{
register byte     i;
register int      S;

        if (!IRI_Ready)          // Если флаг очищен
        {
                IRI_Bits[0]=0;  // Текущий счетчик бита очищен
                S=30000;        // Установить счетчик таймаута
                while (!INT0)   // Если на входе 0
                {
                        WDT();          // Запустит WDT
                        IRI_Bits[0]++; // Увеличить текущий счетчик бита
                        S--;            // Уменьшить счетчик таймаута
                        if (S<=0) return; // Если таймаут – выход из функции
                }
        }
        // Если буфер не переполнен – выход из функции
        if ((IRI_Bits[0]<500)|(IRI_Bits[0]==0)) return;
        else // В противном случае
        {
                IRI_Bits[0]=0;          // Текущий счетчик бита очищен
                for(i=0;i<8*19;i++)     // До максимального числа битов
                                        // из посылки в 19 байтов
                {
                        S=30000;        // Установить счетчик таймаута
                        while (INT0!=0) // Если на входе «1»
                        {
                                WDT();          // Запустить охранный таймер
                                S--;            // Уменьшить счетчик таймаута
                                if (S<=0) return; // Если таймаут – выход из функции
                        }
                        S=30000;
                        while   (!INT0) // Если на входе «0»
                        {
                                WDT();          // Запустить охранный таймер
                                IRI_Bits[i]++;  // Увеличить текущий счетчик бита
```

```
                                S--;                    // Уменьшить счетчик таймаута
                                if (S<=0) return; // Если таймаут – выход из функции
                        }
                }
        }
        IRI_Ready=1; // Установить флаг готовности
        RM_Frame ();  // Обработать входной массив битов
}
// ************************************
// Подпрограмма преобразования входного массива битов
// во входной фрейм из 19 байтов
void RM_BitsToBytes (void)
{
register byte i, j;
register int M=1000;

        for(i=0;i<160;i++) // Начальная фильтрация массива битов
        {
                if ((IRI_Bits[i]<M)&(IRI_Bits[i]>0)) M=IRI_Bits[i];
        }
        M*=2;
        for(j=0;j<19;j++)  // Преобразование битов в байты
        {
                WDT();
                IRI_Frame[j]=0;
                for(i=0;i<8;i++)
                {if (IRI_Bits[i+j*8]>M) IRI_Frame[j]|=0x80>>i;}
        }
}
// ************************************
// Подпрограмма обработки фрейма
void RM_Frame (void)
{
register byte CS, i;

        // Преобразование входного массива битов
        // во входной фрейм из 19 байтов
        RM_BitsToBytes ();
        if ((IRI_Frame[0]!=0xAA))       IRI_Ready=0;
        else                            IRI_Ready=1;
        // Если начинается с преамбулы – установить
        // флаг готовности и обнулить входной массив
        memset (IRI_Bits,0,sizeof(IRI_Bits));
}
```

Третий программный модуль RM_OUT.C содержит набор функций, предназначенных для передачи информации по инфракрасному интерфейсу.

```
// ************************************
// RM_OUT.C                        Version 02.00  *
// ************************************
#include "M570P.h"
// Формат выходного фрейма по ИК интерфейсу
xdata BYTE IRO_Frame[20];
//IRx_Frame[0]=0xAA - преамбула
//IRx_Frame[1] - код команды
//IRx_Frame[2]-IR_Frame[17] - данные (16 байт)
```

```
//IRx_Frame[18] – контрольная сумма

#define COEFF 4 // Коэффициент учета частоты
// ************************************
// Эта функция используется для передачи бита
// с длительностью LEN
void RM_Floor (unsigned LEN)
{
register byte i;
        for(i=0;i<LEN;i++)
        {IRO=0; Time(1*COEFF);
        IRO=1; Time(2*COEFF);}
}
// ************************************
// Эта функция генерирует 1 бит
void RM_Send_Bit (bit BIT)
{
        if (BIT) RM_Floor (64); // Бит «1»
        else     RM_Floor (16); // Бит «0»
        Time (100);
}
// ************************************
// Функция генерирует сихроимпульс
void RM_Send_Syn (void)
{
        WDT(); // Запустить охранный таймер
        RM_Floor (128); // Длинный синхроимпульс
        WDT(); // Запустить охранный таймер
        Time (400);
}
// ************************************
// Функция передачи байта
void RM_Send_Byte (byte CH)
{
register byte i;
register byte MM;

        for(i=0;i<8;i++) // Передать 8 битов
        {
                MM=0x80>>i; // Сдвиг битов
                if (!(CH&MM)) RM_Send_Bit (0);
                else          RM_Send_Bit (1);
                WDT();
        }
        Time (100);
}
// ************************************
// Функция передачи фрейма = 19 байтов
void RM_Frame_Send (void)
{
register byte i;

        EA=0;           // Запретить прерывания
        WDT();          // Запустить охранный таймер
        RM_Send_Syn();// Функция передачи байта
        WDT();          // Запустить охранный таймер
        for (i=0;i<19;i++) // Передать байты
```

```
            RM_Send_Byte (IRO_Frame[i]);
        WDT();          // Запустить охранный таймер
        EA=1;           // Разрешить прерывания
}
```

Естественно также, что в проект программного обеспечения устройства входит файл описаний, содержащий описания всех функций, используемых в проекте (содержащихся в трех приведенных модулях). Важно отметить, что кроме описаний функций в файле описаний находится макроопределение форматной строки, которое мы приводим ниже. Оно позволяет конвертировать форматный ввод в строку символов:

```
#ifndef         __STDARG_H__
#include        "stdarg.h"

#define GetFMT()        xdata char fmtbuf[64];\
                        va_list argptr;\
                        va_start(argptr,fmt);\
                        vsprintf(fmtbuf,fmt,argptr);\
                        va_end(argptr);
#endif
```

Кроме этого, в файле описаний находятся определения линий ввода вывода микроконтроллера:

```
sbit RED        = P1^2;     // Красный светодиод
sbit GREEN      = P1^3;     // Зеленый светодиод
sbit IRO        = P1^5;     // Инфракрасный светодиод
sbit INT0       = P0^6;     // Выход инфракрасного фотоприемника
sbit INT1       = P0^7;     // Не используется
```

В заключение отметим, что в описанном устройстве длина посылок (пакета, фрейма) по инфракрасному каналу составляет 19 байтов. Эта длина определяется наличием кода преамбулы (0-й байт), кода команды (1-й байт), 16 байт текстовых данных (2-й – 17-й байты) и кода контрольной суммы (19-й байт). Длина текстовой строки – 16 байт, определяется количеством символов LCD, используемых в некоторых изделиях, связанных с описанным устройством. Естественно, что можно произвольно изменять формат передаваемых данных. Без проблем это можно делать до длины пакета в 64 байта, т.к. это – максимальная длина пакета в используемой USB библиотеке usb_api.lib. При этом следует помнить, что чем больше длина пакета, передаваемого по ИК каналу, тем больше вероятность ошибок передачи. Поэтому рекомендуется выбирать размер пакета минимально возможной длины. Очевидно также, что минимальная длина пакета будет составлять 3 байта – преамбула, команда, контрольная сумма. Исключать преамбулу и контрольную сумму из состава пакета нельзя, т.к. инфракрасный канал имеет очень низкую помехозащищенность из-за возможных паразитных засветок различной природы. Еще больше ухудшается картина, если один из контроллеров находится в движении, т.е. нельзя гарантировать его правильную постоянную ориентацию относительно другого контроллера. В этом случае приходится для повышения надежности канала связи многократно повторять посылки и вводить в фрейм дополнительные поля диагностики и(или) восстановления кодов посылки.

Литература:

1. О. Николайчук Подсистема управления микроконтроллером с помощью пульта дистанционного управления // Схемотехника, 2004, №6, 37-40.
2. http://www.ubs.org
3. http://www.linux-usb.org/usb.ids
4. http://www.silabs.com
5. http://www.silabs.com/public/documents/tpub_doc/anote/Microcontrollers/USB/en/an169.pdf
6. http://www.silabs.com/public/documents/tpub_doc/anote/Microcontrollers/USB/en/an169sw.zip
7. http://www.keil.com