ПОДСИСТЕМА УПРАВЛЕНИЯ МИКРОКОНТРОЛЛЕРОМ С ПОМОЩЬЮ ПУЛЬТА ДИСТАНЦИОННОГО УПРАВЛЕНИЯ

Олег Николайчук
onic@ch.moldpac.md

Статья опубликована: Схемотехника, 2004, №6, с. 37-40

Настоящая статья знакомит читателей с вариантом реализации подсистемы управления микроконтроллером с помощью пульта дистанционного управления. В статье приведено описание аппаратной части инфракрасного приемника и универсального «скользящего» алгоритма анализа входной последовательности. Приведен пример программной реализации «скользящего» алгоритма на языке «С» (Keil).

Большинство современных микроконтроллерных систем различного назначения использует одну или несколько кнопок управления. В некоторых случаях, например, в сверхминиатюрных контроллерах, от кнопок желательно избавляться с целью уменьшения площади печатной платы. В других случаях желательно обеспечить удаленное бесконтактное управление контроллером. Изящным решением этой проблемы является использование в микроконтроллерной системе интегрированного инфракрасного (ИК) фотоприемника. В настоящее время многие фирмы выпускают такие интегрированные приемники. Мы выбрали для реализации этой задачи рядовой малогабаритный интегрированный ИК фотоприемник TSOP4136 фирмы Vishay Telefunken[1], имеющий настройку на стандартную частоту передачи 36 кГц. Этот приемник имеет интегрированный PIN фотодиод, предусилитель с фильтром и выходной транзистор. Корпус ИК фотоприемника выполнен из специальной эпоксидной смолы, прозрачной для ИК излучения и не прозрачной для видимого диапазона. Непосредственное перед PIN фотодиодом корпус имеет собирающую линзу. Фотоприемник имеет малые размеры корпуса 7 x 6 мм при толщине 4 мм. Внешний вид ИК фотоприемника TSOP4136 показан на рис.1, а его функциональная схема – на рис.2.

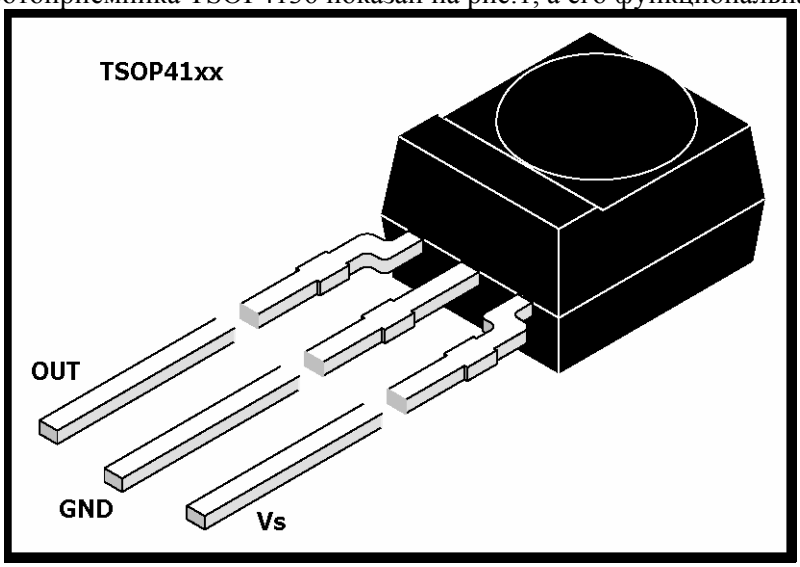

Рис.1. Внешний вид ИК фотоприемника TSOP4136

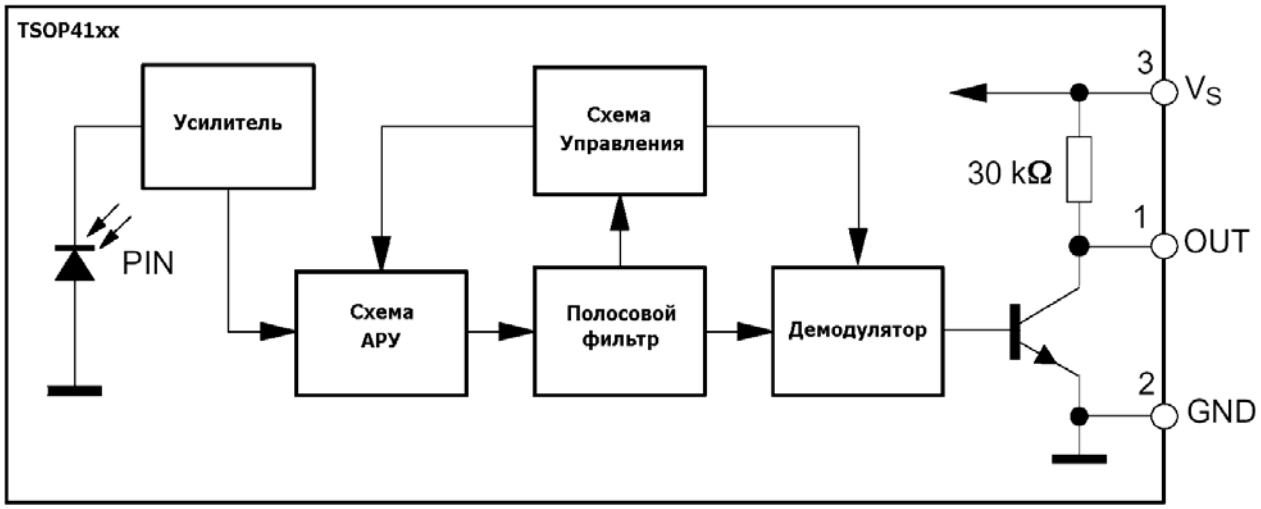

Рис.2. Функциональная схема семейства ИК фотоприемников TSOP41xx

Особенностью этого фотоприемника является то, что он реагирует только на серии световых импульсов, частота которых должна быть в пределах 36 кГц ± 5%, а длительность серии – не менее 8 импульсов. При этом, при поступлении световых импульсов с указанными параметрами на PIN фотодиод, на выходе формируется отрицательный импульс напряжения соответствующей длительности. В отсутствие световых импульсов с необходимыми параметрами на выходе присутствует высокий логический уровень. Следует отметить, что в этой же серии имеются и другие фотоприемники с отличающимися частотами встроенного фильтра: 30, 33, 35, 36.7, 38, 40 и 56 кГц. Однако «несущая» частота 38 кГц является наиболее распространенной среди стандартных пультов дистанционного управления телевизионных приемников, музыкальных центров, CD проигрывателей и другой бытовой техники. Напряжение питания такого приемника – 5 В, ток потребления и выходной ток примерно равны 5 мА. Фотодиод приемника имеет максимум чувствительности в ближней инфракрасной области при длине волны излучения 950 нм. Приемник обеспечивает обнаружение сигнала при угловых отклонениях от оси, перпендикулярной плоскости PIN фотодиода (корпуса), до ±50º.

Принципиальная схема подключения ИК фотоприемника к микроконтроллеру приведена на рис.3.

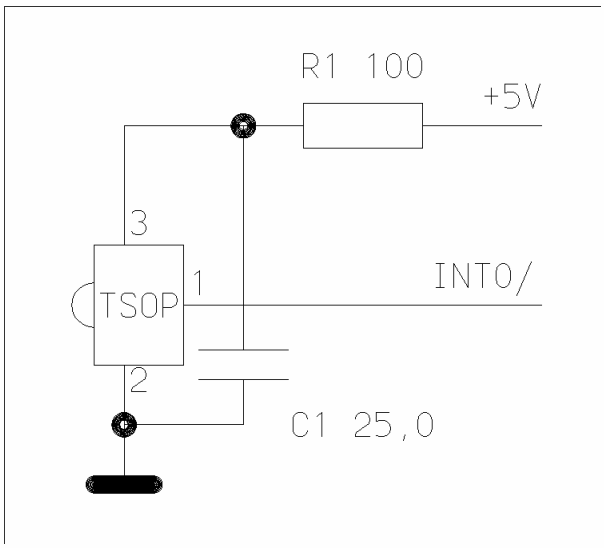

Рис.3. Принципиальная схема подключения ИК фотоприемника к микроконтроллеру

Резистор R1 и конденсатор C1 образуют фильтр питания. Изготовитель рекомендует применять электролитический конденсатор емкостью 100 мкФ, однако при использовании танталовых конденсаторов емкость может быть снижена до 20-25 мкФ. Выход ИК приемника подключен непосредственно на вход прерывания микроконтроллера.

Теперь обоснуем применение универсального «скользящего» алгоритма распознавания. На этапе разработки описываемой подсистемы автор провел поиск документации для пультов дистанционного управления. К сожалению, оказалось, что таких протоколов в настоящее время существует более 60, т.к. каждый из изготовителей бытовой техники использует, как правило, свой протокол. Наиболее широко известен и доступен протокол RC-5 (и даже примеры программ для его декодирования с помощью микроконтроллера[2]), используемый в отечественной технике, однако не смотря на то, что в стандарте оговорены базовые тактовые частоты, используемые для формирования передаваемых ИК последовательностей, во многих современных пультах кварцевые резонаторы заменяются на более дешевые конденсаторные генераторы, имеющие отклонение от базовой тактовой частоты и низкую стабильность частоты, зависящую от температуры и напряжения питания. Для других протоколов, используемых зарубежными производителями, вообще не удалось найти не только информацию о используемом протоколе (кроме названия), но и об используемых передающих микросхемах, т.к. в многих пультах используются заказные специализированные микросхемы, производимые только для внутреннего потребления. Следует отметить, что в последнее время появилось довольно много малогабаритных пультов дистанционного управления – брелков, имеющих довольно малые размеры, примерно как спичечный коробок, и небольшое количество кнопок, как правило, до 8. Эти пульты питаются от одной плоской батарейки (таблетки) с напряжением питания 1,2 В и позволяют управлять оборудованием на небольшом расстоянии – до 3 метров. Эти пульты очень хорошо подходят для наших целей - управления микроконтроллером.

Учитывая описанную ситуацию с документацией на протоколы, автором была поставлена задача создания универсального алгоритма распознавания, который бы позволял принимать и декодировать кодовые последовательности любых пультов, работающих в довольно широком диапазоне базовых тактовых частот.

Известно, что многие пульты дистанционного управления используют так называемый Манчестерский код, в котором каждый бит информации передается, как два равных по длительности логических уровня. Причем, логический ноль передается как последовательность низкого, а затем высокого уровней, а логическая единица, как последовательность высокого, а затем низкого уровней. Длительность каждого из потенциалов примерно одинакова и составляет от 0,5 до 1,5 мс для различных протоколов, и соответственно пультов. Некоторые пульты повторяют последовательность, соответствующую нажатой кнопке пульта столько, сколько будет удерживаться кнопка, другие пульты вырабатывают только одну последовательность в момент нажатия или отпускания кнопки. Многие пульты перед собственно кодовой последовательностью передают своеобразную преамбулу – длинные нулевой и более короткий единичный уровни, длительность которых может достигать нескольких мс. Количество передаваемых байтов данных различно для разных протоколов и, как правило, не превышает 6 байт.

Идея универсального «скользящего» алгоритма распознавания заключается в том, чтобы обнаружить по прерыванию преамбулу (т.е. обнаружить первый длинный нулевой импульс), пропустить длинный единичный импульс, а затем посчитать длительность каждого нулевого и единичного интервалов для каждого из передаваемых битов. Работу алгоритма поясняет рисунок 4 и приведенные ниже подпрограммы.

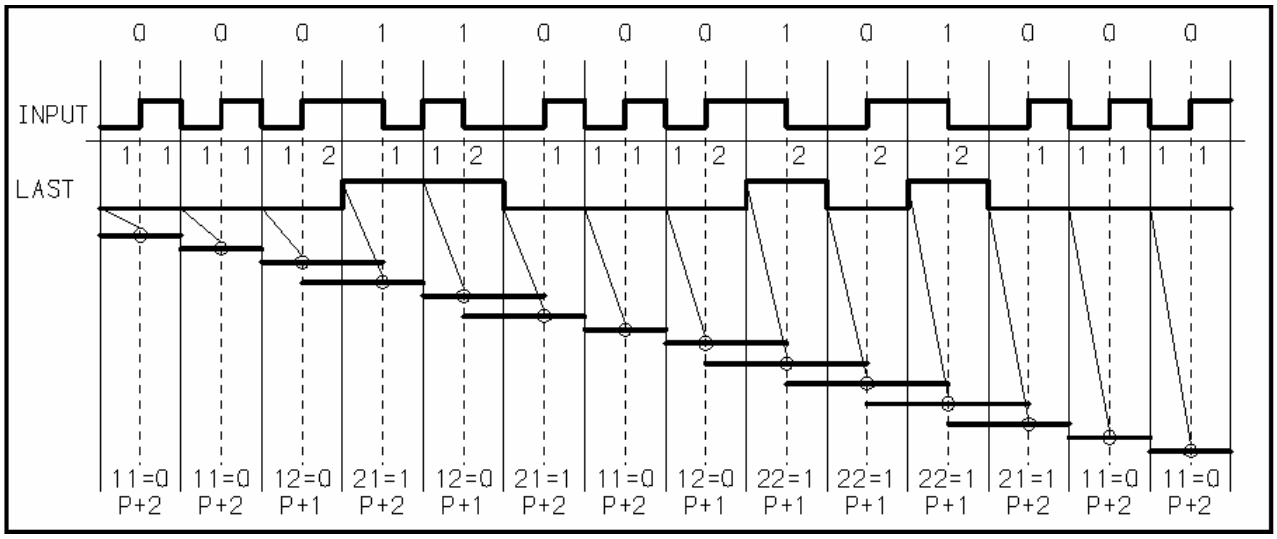

Рис.4. Фрагмент временной диаграммы работа ИК приемника, поясняющий алгоритм

Прежде чем приступить к рассмотрению подпрограммы, приведенной для микроконтроллера C8051F021 фирмы SiLabs[3] и написанной на языке C фирмы Keil[4], определим глобальные переменные, используемые приводимыми подпрограммами:

```
//*************************************************
// Код команды, переданной выбранным пультом
xdata byte      RM_COM;
// Входной массив фазовых битов, размер которого должен
// быть больше, чем максимальное число байтов ИК посылки,
// умноженное на число битов (8) и умноженное на число фаз
// (2). В нашем случае 5 байт * 8 * 2 = 80
xdata byte      RMB[80];
// Входной массив битов последовательности = 5* 8
xdata byte      RMS[40];
// Входной байтовый массив 5+1
xdata byte      RM_Byte[6];
// Флаг готовности
xdata byte      RM_Ready;
//*************************************************
```

Собственно весь алгоритм распознавания располагается в теле подпрограммы обслуживания используемого прерывания. Однако, прежде чем рассматривать эту подпрограмму отметим, что она использует внешнюю функцию перезапуска охранного таймера WDT():

```
void WDT (void)                 {WDTCN=0xA5;}
```

Для выбранного процессора, работающего на частоте 22.1184 МГц, эта функция выполняется примерно за 100 нс. Отметим, что при конфигурировании входов прерывания необходимо настроить используемый вход на работу по отрицательному перепаду напряжения. Иными словами, прерывание будет генерироваться тогда, когда с выхода ИК приемника поступит первый длинный нулевой импульс.

Фрагмент передаваемый значений битов на рис.4 в виде временной диаграммы LAST, а соответствующий фрагмент реального входного сигнала показан в виде временной диаграммы INPUT. Для каждого из потенциальных уровней реального входного сигнала INPUT мы посчитаем значения длительности в условных единицах времени выполнения времязависимой подпрограммы и запишем последовательно в массив RMB.

```
//*************************************************
void Ext0_INT (void) interrupt 0
{
register byte i;            // Универсальный указатель
register byte OPTR;     // Указатель «реальных» битов
static bit LAST;                    // Бит предыдущего состояния
static byte MID;        // Пороговое значение счетчика фазы
static byte MIN;        // Минимальное значение счетчик фазы

//***** ИЗМЕРИТЕЛЬНАЯ ЧАСТЬ *****//

// В момент поступления первого нулевого импульса запретить
// все прерывания, чтобы не нарушать правильный подсчет
// входных импульсов
        EX0=0;
// Ожидать завершения первого отрицательного (нулевого)
// импульса, собственно и вызвавшего прерывание
        while(!INT0) WDT();
// Ожидать завершения положительного (единичного) импульса,
// следующего за отрицательным
        while(INT0) WDT();
```

```
// Вызвать подпрограмму подсчета длительности каждого из
// фазовых полубитов с общим количеством, равным 5*8*2
// Условная длительность каждого из фазовых полубитов пос-
// ледовательно записывается в каждый байт массива RMB
        for (i=0;i<80;i+=2)
        {
            if (RM_GetBit(i)==ERROR) break;
        }
// Разрешить все прерывания, поскольку критичная к времени
// часть завершена
        EX0=1;
        IE0=0;  // Сбросить флаг прерывания

//***** ОПРЕДЕЛЕНИЕ ПОРОГА РАСПОЗНАВАНИЯ *****//
// В этой части подпрограммы определяется минимальное
// значение счетчика i-ой фазы полубита, необходимое для
// дальнейших преобразований

// Установить максимальное значение порога
        MIN=255;
// Перебрать весь входной массив фазовых полубитов
        for (i=0;i<80;i++)
        {
// Если значение счетчика текущего полубита меньше величины
// MIN и не равен 0, присвоить величине MIN значение счетчика
// текущего полубита
            if ((RMB[i]<MIN)& (RMB[i]!=0))  MIN= RMB[i];
        }
// Определяем среднее значение порога для сравнения
        MID=MIN+MIN/2;


//***** ОПРЕДЕЛЕНИЕ ПОРОГА РАСПОЗНАВАНИЯ *****//
// В данном фрагменте мы определяем, какому количеству полу-
// бит соответствует полученная условная длительность, запи-
// санная в каждом из байтов входного массива RMB. Для
// сравнения условной длительности используем полученное
// среднее значение условной длительности MID

// Перебрать весь входной массив фазовых полубитов
        for (i=0;i<80;i++)
        {
// Если значение равно нулю – продолжить сравнение
            if (!RMB[i])     continue;
// Если условная длительность больше средней, записать
// в этот же байт значение 2, иначе 1.
            RMB[i]=(RMB[i]>MID) ? 2 : 1;
        }
// В результате этой операции входной массив, содержащий
// измеренные значения условных длительностей фазовых
// полубитов мы преобразовали в массив, содержащий значения
// чередующихся полубитов (1 или 2 или 0)


//***** ПРЕОБРАЗОВАНИЕ МАССИВА ПОЛУБИТОВ В
            МАССИВ ИСХОДНЫХ БИТОВ *****//
// Внимательно рассмотрев рис.4 можно заметить вполне четкую
```

```
// закономерность восстановления исходных переданных битов
// из полученного выше массива чередующихся полубитов,
// значения которого указаны цифрами под нулевой линией INPUT

        memset (RMS,0,sizeof(RMS));  // Обнулим массив битов

        OPTR=0;         // Обнулим указатель битов
        i=0;            // Обнулим указатель полубитов
        LAST=0;         // Обнулим значение восстанавливаемого бита

// Выполнять до восстановления 40 битов = 5 байтам
        while (OPTR<40)
        {
// Если текущий логический уровень имел двойную длительность
// Необходимо проинвертировать восстанавливаемый          исходный бит
// Смотри предпоследний ряд надписей на рис.4 типа xx=0(1), где
// первая цифра – текущий фазовый полубит;  вторая цифра –
// следующий фазовый полубит;  0 или 1 после знака равенства
// означает значение восстанавливаемого исходного бита.

        if (RMB[i]==2)          LAST=~LAST;
// Записать бит в массив битов и увеличить указатель битов
                RMS[OPTR++]=LAST;
// Прервать, если очередное значение фазового полубита - нулевое
                if (!RMB[i+1]) break;
// Если следующий полубит двойной, сдвинуть указатель на 1,
// иначе сдвинуть указатель на 2
                if (RMB[i+1]==2) i+=1; else     i+=2;
        }

//***** ПРЕОБРАЗОВАНИЕ МАССИВА ИСХОДНЫХ БИТОВ
                  В БАЙТОВЫЙ МАССИВ *****//
        for (i=0;i<5;i++) {RM_Byte[i]=RM_GetB (i);}
// Установить флаг готовности принятой последовательности
// Флаг должен обнуляться, подпрограммной, выполняющей
// принятые команды после завершения выполнения
        RM_Ready=1;
}
```

Подпрограмма подсчета длительности двух полубитов. Величина NUM определяет адрес (номер) бита. Условная длительность каждого из фазовых полубитов записывается последовательно в байты массива RMB. Длительность определяется в условных единицах с интервалом примерно 32 мкс
(время выполнения внешней подпрограммы Time(2)). Длительность временного интервала выбрана из тех соображений, чтобы за время двойного потенциала любой фазы значение счетчика не превысило 250, т.е. размерности байта. Если за время измерения фазы ее длительность превысила значение 250 – это означает, что ИК последовательность завершилась либо произошла потеря сигнала. При этом подпрограмма возвращаем значение ERROR = 0, иначе значение OK=1; При возвращении функцией значения ERROR, дальнейший прием входных полбитов прекращается и последующие байты массива RMB останутся нулевыми. Следует помнить, что подпрограмма, выполняющая полученные команды (монитор) должна после выполнения очередной команды сбросить флаг готовности RM_Ready=0 и обнулить входной массив!

```
 //*************************************************
byte RM_GetBit (int i)
{
        while (!INT0)           // Пока на входе 0
        {
```

```
        RMB[i+0]++;   // Увеличить счетчик полубита i
        Time(2);          // Задержка ~ 32 мкс
        if (RMB[i+0]>250) // Если больше 250
              return ERROR;    // Вернуть ошибку
        }
        while ( INT0)             // Пока на входе 1
        {
             RMB[i+1]++;       // Увеличить счетчик полубита i+1
             Time(2);   // Задержка ~ 32 мкс
             if (RMB[i+1]>250) // Если больше 250
                    return ERROR; // Вернуть ошибку
        }         return OK;       // Нормальное завершение
}

//*****************************************************
// Подпрограмма восстанавливает значение байта  из битов
byte RM_GetB (int NUM)
{
register byte i;
register byte ch=0;

        for (i=0;i<8;i++) {if (RMS[NUM*8+i])  ch |= 0x80>>i;}
        return ch;
}

//*****************************************************
// Подпрограмма инициализации переменных
void RM_Init (void)
{
        RM_Ready=0;
        RM_COM=0;
        memset (RMB,0,sizeof(RMB));
        memset (RM_Byte,0,sizeof(RM_Byte));
}
//*****************************************************
```

На этом можно было бы и закончить описание «скользящего» алгоритма, однако для полноты рассмотрения напомним читателю, что еще необходимо делать для организации управления микроконтроллером от телевизионного пульта с помощью ИК приемника.

Итак, мы получили массив байтов. Далее в подпрограмме монитора необходимо для каждой из возможных последовательностей байтов выполнить определенные действия, затем сбросить флаг готовности и обнулить входной массив полубитов, а еще проще – вызвать подпрограмму RM_Init ().

Приведенную подпрограмму несомненно можно оптимизировать, исключив массив RMS и функцию RM_GetB и восстанавливая значения битов непосредственно в массив RM_Byte.

Рассмотренный алгоритм и приведенная подпрограмма имеет один недостаток – значительный объем используемой памяти для входного массива RMB, но это есть неизбежная плата за универсальность и не зависимость от частоты ИК посылок и частотной стабильности несущего генератора. Автор надеется, что приведенная статья окажется полезной читателям, интересующимся описанной проблемой.

Литература:

1. http://www.vishay.com/docs/82135/82135.pdf
2. InfraRed Remote Control - http://www.ustr.net
3. http://www.silabs.com
4. http://www.keil.com