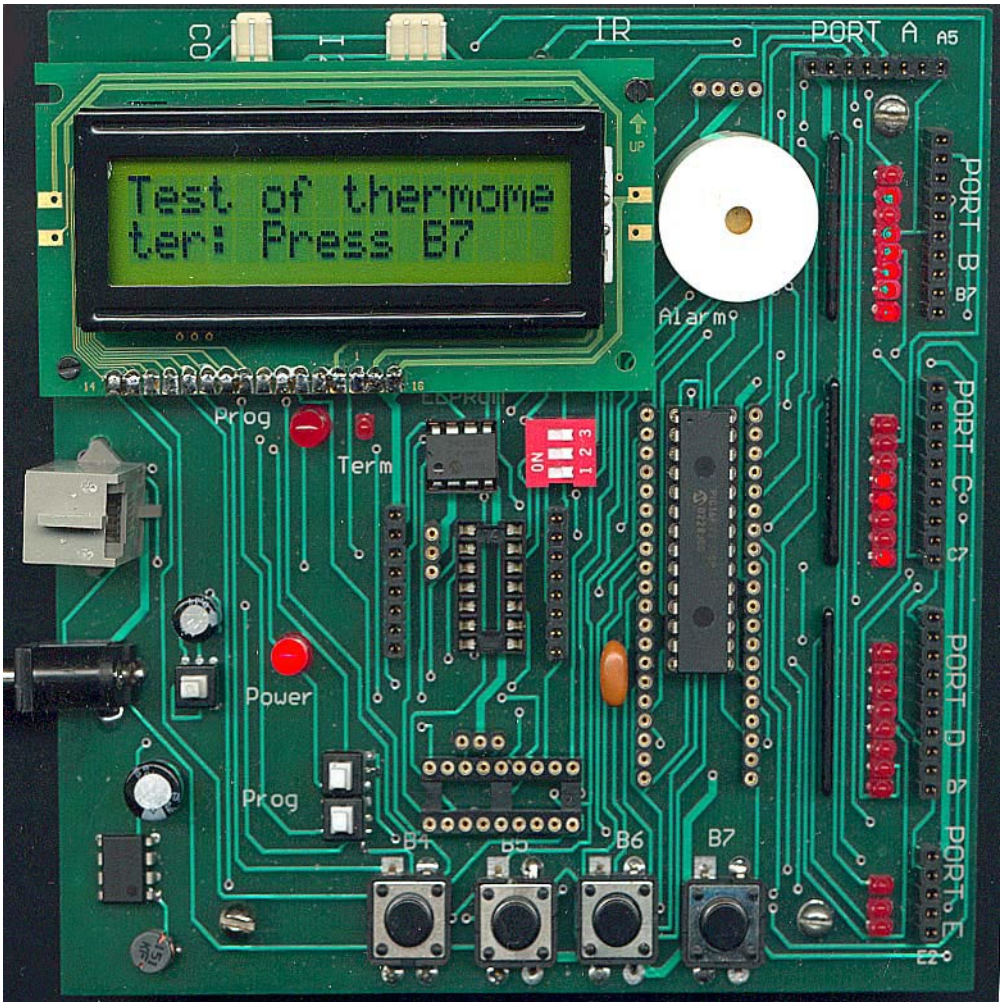
Описание лабораторно-отладочного комплекса ЛОК-1

Методические указания к выполнению лабораторных работ и курсового проектирования

Автор:
к.т.н. доц. каф. Информационных систем
Санкт Петербургского Государственного Университета
Аэрокосмического Приборостроения (быв. ЛИАП)
Ковалев С.И.

Санкт Петербург
2002г.

Оглавление



Лабораторно-отладочный комплекс для PIC-контроллеров.

Назначение. Лабораторно-отладочный комплекс (ЛОК) предназначен для макетирования и отладки различных устройств на базе PIC-контроллеров. Он может быть полезен как для разработчиков embedded-устройств так и для молодых специалистов, желающих ознакомится с основами микроконтроллерной техники и овладеть мощным инструментом для реализации широкого спектра задач цифровой обработки данных и управления различного рода периферийными устройствами.

Возможности ЛОК. В основу ЛОК положен принцип максимальной универсальности и удобства программирования и отладки программ на базе PIC-контроллеров. ЛОК является идеальным средством обучения, позволяющим в короткий срок научиться решать типовые задачи сопряжения PIC-контроллеров с наиболее часто встречающимися на практике периферийными модулями. В частности, ЛОК позволяет изучить сопряжение PIC-контроллеров с жидкокристаллическим алфавитно-цифровым индикатором (LCD), связать контроллер с внешней последовательной памятью (EEPROM) по протоколу связи I2C, организовать связь с последовательным портом персонального компьютера (COM-порт), связь с периферийными устройствами через канал инфракрасной связи (IR-связь), организовать работу с интеллектуальными датчиками температуры (на примере микросхемы DS1621 фирмы Dallas).

ЛОК имеет “на борту” все перечисленные выше элементы и не требует напайки внешних навесных деталей. Вместе с тем, каждый порт PIC-контроллера имеет выход на отдельный разъем, куда можно подключить любое внешнее устройство. Предусмотрена

светодиодная индикация сигналов на входах-выходах портов B,C,D,E PIC-контроллера. Таким образом, ЛОК можно использовать не только как средство обучения, но и как устройство для макетирования и наладки реальных практических разработок на базе PIC-контроллеров.

ЛОК имеет несколько панелек для установки PIC-контроллеров в корпусах DIP-8, DIP-14, DIP-18, DIP-28, DIP-40, куда можно устанавливать практически любые контроллеры, включая самые последние модели флешевых восьми- и четырнадцати-выводных PIC-контроллеров, а также последние версии широко используемых контроллеров семейств F7x и F87x(A). Вообще говоря, к ЛОК может быть подключен любой контроллер, совпадающий по выводам питания и программирования с наиболее широко распространенными контроллерами 12C509,16C505,16F84,16F87x. Через переходные модули (не прилагаются) можно подключить и другие контроллеры в любых других корпусах. Рядом с каждым "посадочным местом" установлен разъем для подключения внешнего кварцевого резонатора. Т.о. можно варьировать тактовую частоту генератора контроллера установкой соответствующего кварца.

Особое удобство ЛОК заключается в наличие встроенного программатора, поддерживающего практически все существующие разновидности PIC-контроллеров. Встроенный программатор позволяет заменить программу PIC-контроллера, не вынимая его из посадочной кроватки ЛОК. Достаточно присоединить лабораторный комплекс кабелем к LPT-порту компьютера, запустить на компьютере программное обеспечение программатора, нажать одну кнопку на плате ЛОК, и можно "зашивать" новую программу в установленный в ЛОК PIC-контроллер. Список микросхем, поддерживаемых встроенным программатором, весьма обширен и периодически обновляется авторами программного обеспечения программатора.

Авторы ЛОК не распространяют и не продают программное обеспечение программатора, которое можно свободно взять в Интернете на сайте производителей [*http://microengineeringlabs.com/support/epicbeta.htm*](http://microengineeringlabs.com/support/epicbeta.htm) *. Для коммерческого использования этого программного продукта рекомендуется связаться с его авторами.*

Наиболее близким аналогом ЛОК является LAB-X1 (см.http://microengineeringlabs.com/products/labx1.htm), характеристики которого приведены ниже.

К LAB-X1 предлагается приобрести дополнительно программатор (еще 60$). В цену LAB-X1 ($199.95) не входит стоимость микроконтроллера, EEPROM, датчика температуры и еще ряда элементов. У этого устройства есть как некоторые плюсы, так и заметные минусы по сравнению с ЛОК, но в целом по своим функциональным возможностям эти устройства примерно одного класса. Вот краткое описание этого "альтернативного" продукта:

LAB-X1 Experimenter/Lab Board $199.95 (assembled)

The LAB-X1 is microEngineering Labs, Inc.'s first pre-assembled experimenter's platform. While PICProto boards allow you to create your own projects with a minimum of hassle, the LAB-X1 goes one step further. It provides an assembled testbed containing most of the circuitry commonly used with PICmicro microcontrollers (MCUs).

The LAB-X1 contains the circuitry required by the PICmicro to operate: 5-volt power supply, oscillator, reset circuit, as well as additional application circuits. The crystal controlled oscillator includes jumpers to set speeds of 4MHz, 8MHz, 10MHz, 12MHz, 13.32MHz, 16MHz and 20MHz. Application circuits include a switch matrix, potentiometers, LEDs, LCD module, serial EEPROMS*, real time clock*, temperature sensors*, servo connectors, RS232 interface, RS485 interface*, IR interface* and speaker. A prototyping area is also included in case we missed your favorite circuit.

Описание лабораторной установки ЛОК-1

Органы управления и индикации.

На плате ЛОК имеются следующие кнопки:

1) Кнопка “Power” с фиксацией нажатого состояния служит для включения питания. Питание +5V подается на ЛОК одним из двух способов – через разъем питания на самой плате (плюс на центральный контакт разъема) либо через разъем программирования. В разъем программирования вставляется кабель для связи ЛОК с LPT-портом. К разъему кабеля, подключаемому к LPT-порту, присоединен шнур для включения в USB-порт компьютера. USB-порт используется только как источник питания +5V. Если вы собираетесь подключать к внешним шинам ЛОК значительную нагрузку, помните, что нагрузочная способность USB-порта относительно мала – порядка нескольких сотен миллиампер. Кроме того, при макетировании и отладке внешних устройств, питаемых от ЛОК, вполне возможны короткие замыкания, которые могут негативно сказаться на шинах USB-порта материнской платы вашего компьютера. Поэтому, если вы собираетесь подключать к ЛОК внешние устройства, то на начальном этапе макетирования и отладки рекомендуется пользоваться отдельным внешним блоком питания.
2) Кнопка “Prog” без фиксации нажатого состояния служит для сброса ЛОК в начальное состояние и для ввода (прошивки) новых программ в PIC-контроллеры, установленные на ЛОК.
3) Четыре информационные кнопки для ввода управляющих сигналов. Подключены к 4-м старшим разрядам порта B. Над каждой кнопкой имеется надпись, указывающая разряд порта B, к которому эта кнопка подключена. Более подробно правила работы с этими кнопками описаны ниже.

На плате ЛОК имеется следующие светодиодные индикаторы:

1) “Power” - индикатор включения питания.
2) “Prog” - индикатор включения режима программирования. Не рекомендуется производить какие-либо действия с ЛОК в то время, когда светит этот индикатор. Дождитесь окончания режима программирования, или (в случае зависания) просто отключите питание ЛОК.
3) “Term” – индикатор срабатывания электронного термометра, работающего в режиме термостата. Если вы не используете в своей работе электронный термометр, то можете не обращать внимание на этот индикатор. Описание работы с электронным термометром DS1621 будет приведено ниже.
4) По 8 индикаторных светодиодов подключено к портам B,C,D и 3 светодиода – к порту E. Светодиоды подключены к выходам портов через токоограничивающие резисторы. Применение импортных сверхярких светодиодов позволило использовать резисторы с весьма

высоким (1.5к) сопротивлением, что минимизирует влияние этой дополнительной нагрузки на выходы портов контроллеров. Во многих PIC-контроллерах порт A совмещен с входами АЦП. Чтобы не искажать показания АЦП дополнительной нагрузкой, к порту A светодиоды не присоединены. Если PIC-контроллер, с которым вы работаете, не имеет некоторых портов, то вы можете использовать индикаторные светодиоды этих портов по своему усмотрению. Например, если вы работаете с PIC16F84, в котором нет порта D, то вы можете соединить одним или несколькими проводками внешний разъем порта A (или отдельные его разряды) с неиспользуемым внешним разъемом порта D, и, таким образом, получить индикацию состояния порта A на светодиодах, подключенных к выходному разъему порта D. Будьте внимательны при таких соединениях и внимательно ознакомьтесь с расположением сигналов и линий питания на разъемах для подключения внешних устройств (см. ниже).

Общие принципы работы ЛОК-1

Принципы взаимодействия отдельных блоков ЛОК представлены в виде структурной схемы, изображенной на Fig.1. В составе ЛОК-1 имеются панельки, предназначенные для установки различных PIC-контроллеров в корпусах DIP-18, DIP-28, DIP-40. Одноименные порты (A,B,C,D,E) PIC-контроллеров в различных панельках соединены параллельно и имеют общую индикацию для всех PIC-контроллеров, но индивидуальную для каждого порта. Другими словами, куда бы вы не установили какой-либо PIC-контроллер, вы сможете наблюдать (на индикаторных светодиодах) состояния его портов B,C,D,E, если они есть у данного контроллера. Вместе с тем, если вам захочется вставить сразу два контроллера в разные панельки, то вы должны иметь в виду, что их одноименные порты будут соединены друг с другом. В принципе в этом нет ничего плохого, но вы должны будете учитывать это обстоятельство при написании программы. Также параллельно будут соединены цепи программирования контроллеров. Поэтому, при *программировании в ЛОК должен быть установлен только один контроллер, остальные должны быть на это время удалены из ЛОК!*

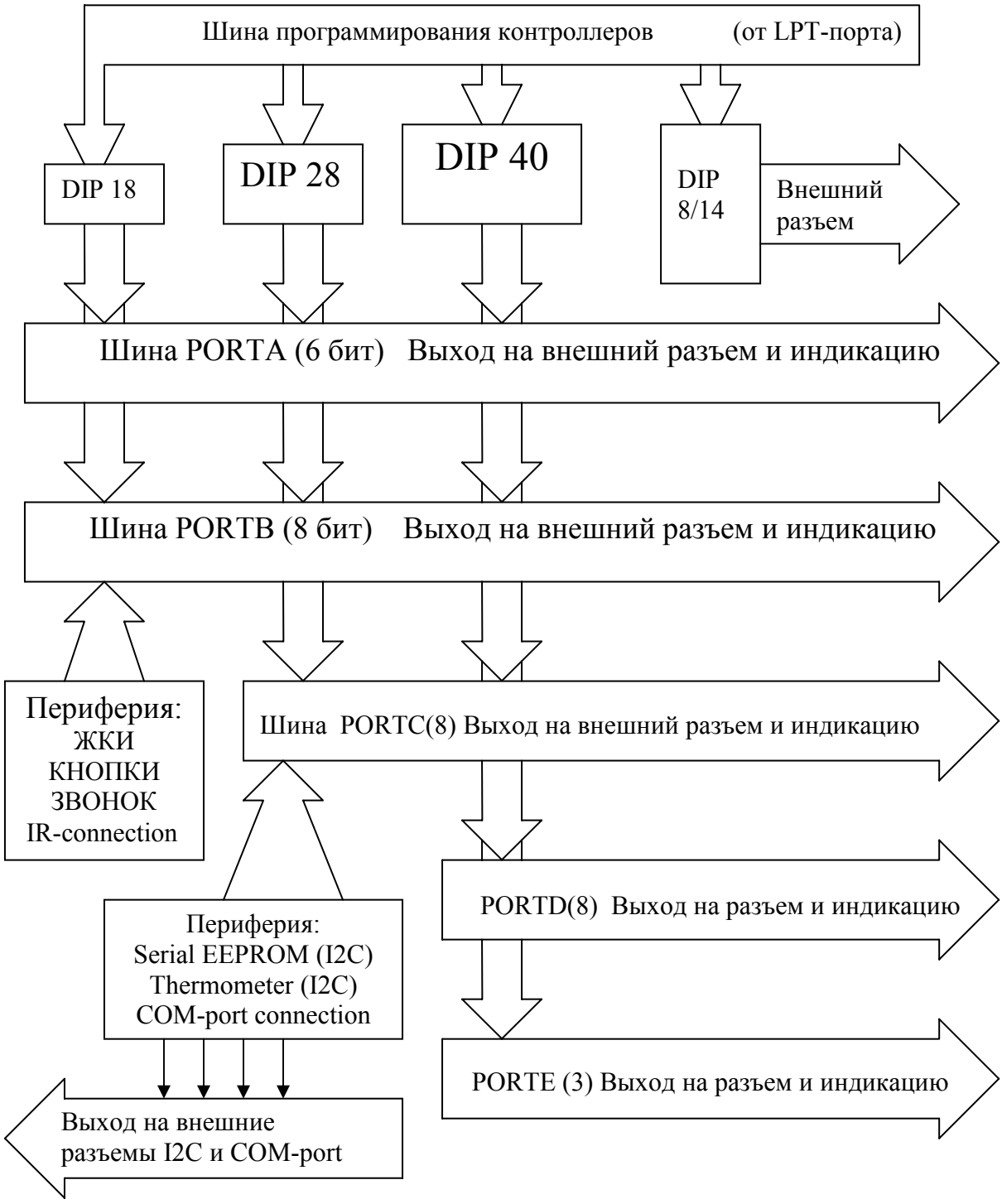


Fig1. ОБЩАЯ СТРУКТУРА ЛОК-1

Панелька для подключения корпусов DIP 8/14 имеет отдельную особую схему подключения к периферии. Она имеет разъемы для подключения внешних цепей к линиям PIC-контроллера и не имеет выхода на собственную индикацию. При необходимости для индикации можно использовать внешние соединения с разъемами порта D, как это было описано выше. Для программирования контроллера, установленного в эту панельку, необходимо перевести все три джампера на переключателе, расположенном рядом с этой панелькой, в состояние “ON”. После программирования их нужно вернуть в первоначальное положение. Эти джамперы подключают линии программирования контроллера DIP 8/14 к общей шине программирования. При разомкнутых джамперах контроллер, установленный в DIP 8/14, абсолютно автономен и не имеет общих цепей с остальными контроллерами (за исключением цепей питания).

Рядом с каждой панелькой имеется дополнительная однорядная микро-панелька, состоящая из трех контактов. (Для DIP-40 и DIP-28 предусмотрена одна общая микро-панелька). Три контакта этой микро-панельки предназначены для подключения кварцевого резонатора по следующей схеме:

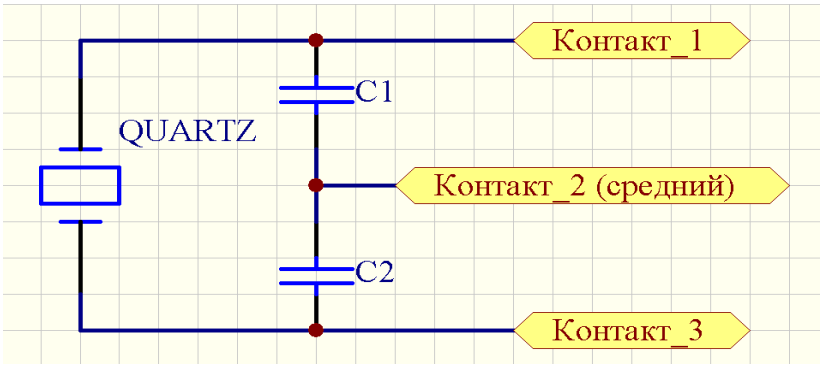


Fig2. Схема подключения кварцевого резонатора.

Это традиционная схема подключения резонатора к контроллеру. Подробности о выборе C1 и C2 можно найти в документации на контроллер. Особенно удобно использовать уже готовые “трехногие” микросборки, состоящие из кварцевого резонатора и встроенных конденсаторов. Их цоколевка совпадает с приведенной на Fig2.

Разъемы для подключения внешних цепей к выводам PIC-контроллера для всех портов устроены одинаково: первый контакт – GND (земля), второй контакт – питание +5V, далее идут линии порта, например, B0,B1…B7. Для того, чтобы легче было определить “начало и конец” разъема, самый последний контакт разъема помечен на печатной плате специальной надписью, например B7 или C7 или A5. Цоколевка разъемов, подключенных к DIP-8/14, приведена на Fig_3.

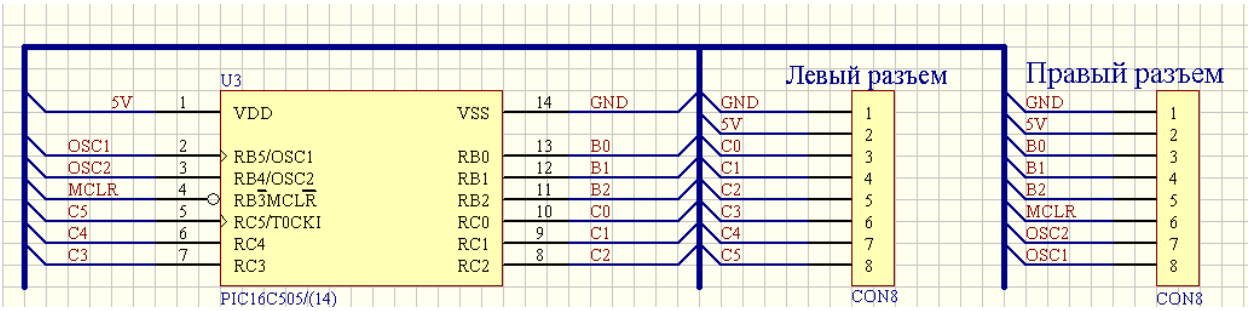


Fig3. Разъемы для подключение внешних цепей к DIP-8/14

Подключение периферии к порту B

Продолжим рассмотрение Fig1. Как видно из рисунка, с панельками соединены не только шины внешних разъемов и индикации, но и элементы внутренней периферии. Рассмотрим кратко подключение периферийных элементов к шинам PORTB и PORTC. Подключение к шине PORTB четырех кнопок, звонка и инфракрасного приемо-передатчика показано на Fig4.

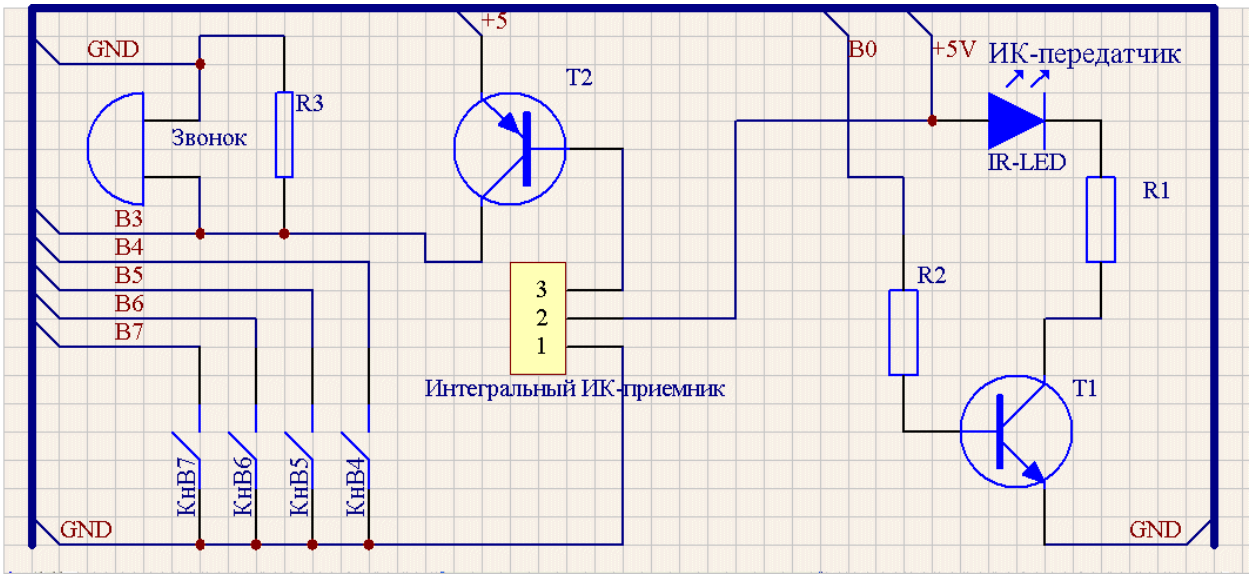


Fig4. Подключение к шине PORTB четырех кнопок, звонка, инфракрасного приемо-передатчика.

Работа с кнопками, как видно из схемы, предельно проста. Для определения состояния кнопки (нажата - отжата) надо:

1) перевести требуемый разряд порта B в режим вывода информации;
2) вывести в этот разряд логическую единицу
3) не переводя разряд порта в режим чтения, прочесть информацию на входе этого разряда

Если кнопка была нажата, то при чтении мы получим не логическую единицу, которую мы только что туда отправили, а логический ноль, полученный в результате закорачивания вывода порта на землю.

Этот способ опроса кнопок самый простой с точки зрения аппаратной реализации. Его допустимость объясняется способностью PIC-контроллера ограничивать ток через вывод порта, т.е. “короткие замыкания” отдельных выводов порта для него не страшны. Тем более допустимы относительно кратковременные замыкания, возникающие при нажатиях на кнопки. В качестве примера универсальной процедуры опроса кнопок приведем подпрограмму, которая опрашивает все кнопки и формирует байт данных, содержащий информацию о нажатых кнопках. Нажатым кнопкам соответствуют единицы в младших четырех разрядах возвращаемого байта. В старших разрядах – всегда нули. Т.е., если подпрограмма возвратила число 3, это означает, что нажаты кнопки КнВ4 и КнВ5. Если получено число 8, то нажата одна кнопка КнВ7 и т.д.

```
//-------------------------------------------------------------------------------
    byte Check_buttons(void)
    {
    byte tmp, tmp_PORTB=PORTB, tmp_TRISB=TRISB;
    TRISB&=0x0F; PORTB|=0xF0;
    tmp=PORTB^0xFF; PORTB=tmp_PORTB; TRISB=tmp_TRISB;
    return(tmp>>4);
    }
//-------------------------------------------------------------------------------
```

Тип *byte* должен быть определен пользователем как unsigned char.

Работа с пьезокерамическим звонком тоже не требует больших усилий от программиста. Для формирования звукового сигнала необходимо на вывод В3 порта В выдать периодическую последовательность нулей и единиц так, чтобы сформировался сигнал с основной частотой 1..5 кГц. Пример подпрограммы, которая производит короткий писк (beep) при тактовой частоте процессора 4MHz, приведен ниже.

```
//-------------------------------------------------------------------------------
    void Delay(unsigned int );
    void Beep(void)
    { byte tmp_TRISB=TRISB, tmp_PORTB=PORTB, i;
    TRISB3=0;  // clrbit (TRISB,3);
    i=100; while(i--){
                RB3=1;Delay(30);
                RB3=0;Delay(30);
                    }
      PORTB=tmp_PORTB;
      TRISB=tmp_TRISB;
    }
    void Delay(unsigned int tmp)    //   ( tmp=1000) ~  11 mS (4MHz)
    { while( tmp--);  return;}
//-------------------------------------------------------------------------------
```

Работа с инфракрасным приемопередатчиком будет рассмотрена ниже. Здесь мы лишь отметим, что случайные срабатывания приемника слышны как короткие щелчки звонка и могут приводить к нарушению работы программы, если вывод B3 порта B используется программой еще для каких-либо целей. Поэтому рекомендуется вынимать интегральный приемник из установочной панельки, если не предполагается его использование в отлаживаемой программе.

Работа с ЖКИ

Рассмотрим работу наиболее ответственного и интересного узла ЛОК – жидкокристаллического индикатора. С физическими принципами, положенными в основу ЖКИ, можно познакомиться по статье, содержащейся в файле “ Жидкокристаллические индикаторы.htm “

С точки зрения программиста ЖКИ можно рассматривать как некоторой прибор, имеющий несколько внутренних регистров, через которые можно воздействовать на режимы работы ЖКИ, выводить на индикатор различные символы и т.д. Программист имеет доступ к этим регистрам через шину данных/команд и шину управления индикатора. ЖКИ имеет шину данных, по которой можно посылать в ЖКИ данные и команды и считывать содержимое внутренней памяти и регистров. Управление процессом передачи производится через дополнительную шину управления, которая обычно состоит из линии подачи синхроимпульса, сигала, указывающего направление передачи информации, сигнала, указывающего тип передаваемой информации (данные или команда). Алгоритм работы с ЖКИ существенно зависит от типа специального микроконтроллера, установленного на плате ЖКИ и управляющего всей его работой. В нашем случае это контроллер HD44780. Имеется обширная литература, посвященная архитектуре и правилам работы с ЖКИ на базе HD44780. Для дальнейшего чтения данного описания совершенно необходимо внимательно ознакомится с одним из наиболее удачных и полных, на наш взгляд, описаний правил работы с HD44780, содержащихся в файле lcd.pdf.

Теперь, если вы уже изучили работу с HD44780, можно двигаться дальше. Следует заметить, что существует большое многообразие ЖКИ на базе HD44780, выпускаемых различными фирмами. Индикаторы отличаются по размерам, наличию русского шрифта, типу подсветки и т.д. Наш ЖКИ имеет организацию ”2 строки по 16 символов”, имеет светодиодную подсветку, русифицированный шрифт, работает только при положительных температурах. Он является аналогом ЖКИ **PC1602LRS-FNH-B** фирмы Powertip. Информацию о продукции этой фирмы и об условных обозначениях ЖКИ можно найти в файле ЖКИ.htm и в Приложении 1.

Теперь рассмотрим более подробно, как ЖКИ подключен к линиям шины PORTB в нашей установке.

На Fig5. показано подключение к шине PORTB жидкокристаллического индикатора. Как видно из рисунка, в нашем случае используется подключение половины шины данных, т.е байт информации передается в два приема – сначала старший полубайт, затем младший. Кроме того, вход R/W заземлен, т.е. предусмотрена передача информации только из внешнего мира в ЖКИ. В остальном схема вполне стандартная и других особенностей не имеет.

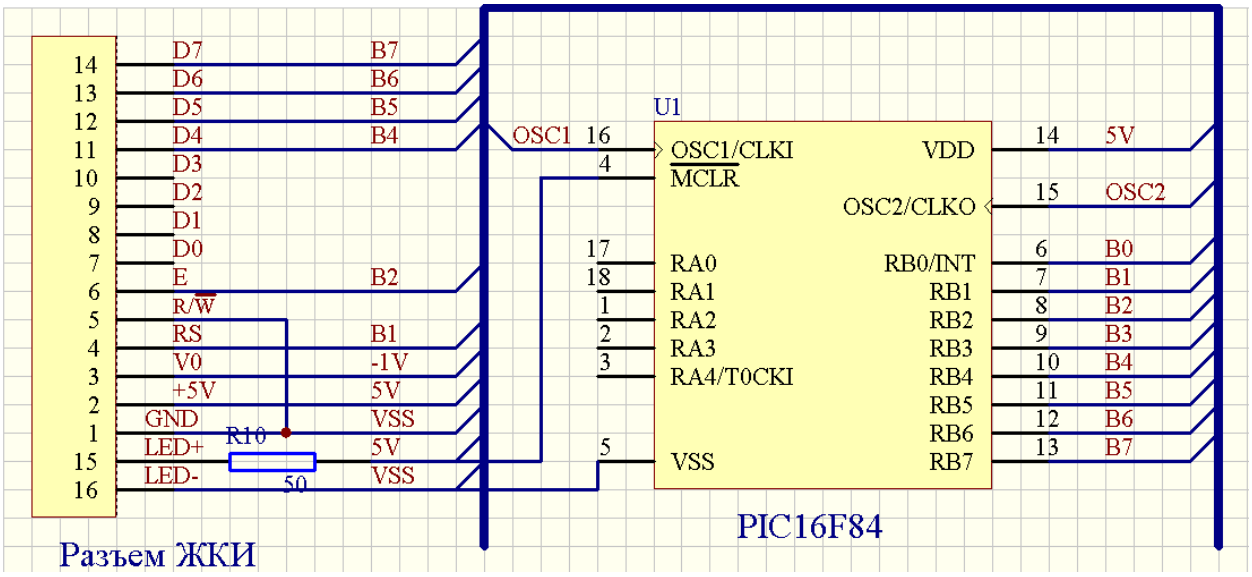


Fig5. Подключение жидкокристаллического индикатора (ЖКИ) к PORTB.

Сигналы разъема ЖКИ:

LED- и LED+ служат для подключения питания к светодиодам внутренней подсветки ЖКИ. Если не подключать питание к этим светодиодам, то ЖКИ можно будет пользоваться только в хорошо освещенном помещении, т.к. цифры будут видны только в отраженном свете. При включении внутренних “осветительных” светодиодов отпадает необходимость во внешнем источнике света и индикатором можно пользоваться даже при плохом освещении.

GND(“земля”) и +5V – линии подачи основного питания на ЖКИ.

V0 линия подачи небольшого отрицательного напряжения для регулировки контрастности изображения на ЖКИ. Некоторые типы ЖКИ могут обходиться и без отрицательного смещения, однако для большинства типов ЖКИ требуется подавать на этот вход до нескольких вольт отрицательного напряжения, чтобы символы стали видны на индикаторе. Как правило, требуемая величина отрицательного напряжения зависит от температуры, и данная величина напряжения регулируется специальным потенциометром, которым можно подстраивать контрастность для текущей температуры окружающей среды. Для лабораторной установки вполне достаточно

работоспособности в диапазоне положительных температур, поэтому в ЛОК-1 такая регулировка не предусмотрена. Используемый в ЛОК-1 тип ЖКИ требует подачи напряжения на вход V0 около -0.5V при потребляемом токе по этому входу около 1 mA.

RS Входной сигнал указывает ЖКИ, что сейчас в ЖКИ будет передана команда (RS=1) или данные (RS=0).

R/W Входной сигнал указывает ЖКИ, куда сейчас должна передаваться информация – от ЖКИ во внешний мир (R/W=1) или из внешнего мира в ЖКИ (R/W=0). В данном макете этот сигнал всегда заземлен, т.е. возможность передачи сигналов от ЖКИ к контроллерам не предусмотрена. Такое схемное решение значительно упрощает работу с ЖКИ, но при этом несколько снижает разнообразие функций, выполняемых ЖКИ. Например, ЖКИ не может “сказать” микроконтроллеру, что ЖКИ не готов к приему очередной команды, т.к. еще не успел выполнить предыдущую. Эта особенность включения ЖКИ требует введения задержки после подачи на ЖКИ очередной команды, чтобы ЖКИ гарантированно успел выполнить данную команду. Требуемая задержка определяется опытным путем и, как правило, не превышает нескольких миллисекунд. Общее быстродействие ЖКИ достаточно велико, и введение задержек практически не заметно для внешнего наблюдателя.

E Входной синхросигнал, подача которого обеспечивает считывание индикатором данных с его входов. Сигнал имеет вид положительного импульса, т.е. он должен изменить свое значение след. образом: 0-1-0. Длительность импульса рекомендуется выбирать достаточно большую – не менее нескольких микросекунд.

D0..D7 Входы данных. Предназначены для передачи в ЖКИ данных и команд и для передачи данных из ЖКИ (последнее не используется в данной установке).

Программная инициализация ЖКИ перед началом работы может быть реализована, как в приведенной ниже подпрограмме. Здесь и далее примеры программ приведены для тактовой частоты контроллера 4 МГц. Использование более высокой частоты может потребовать изменения некоторых задержек в сторону увеличения.

Смысл выполняемых действий вам предлагается установить самим, используя приведенные выше ссылки на литературу.

```
//---------------------------------------------------------------------
   void Pulse(unsigned int x){ RB2=1; Delay(x); RB2=0;Delay(x);}
   //--------------------------------------
   void Init_LCD(void)
   {
   Delay(1000); TRISB=0; PORTB=0x30;
   Pulse(100); Pulse(100);Pulse(100);

   PORTB=0x20;   Pulse(100);
```

```
    Send_Command_LCD  (0x28);
    Send_Command_LCD  (0x0C);
    Send_Command_LCD  (0x06);
    Send_Command_LCD  (0x02);
    Current_ind=0; //should be described above as type " byte" (global)
    }
//-------------------------------------------------------------
```

Данная подпрограмма инициализации использует подпрограмму *Send_Command_LCD*. При написании этого программного модуля необходимо помнить, что шина данных ЛСД в нашей установке совмещена с шиной кнопок, т.е. при нажатой кнопке возможно искажение информации, передаваемой в ЖКИ. Поэтому перед передачей информации в ЖКИ необходимо убедиться в том, что старший полубайт шины PORTB не занят нажатой кнопкой. Затем надо переслать в ЖКИ сначала старший байт команды, затем младший, сопровождая эти передачи соответствующими сигналами шины управления. Пример такой программы приведен ниже.

```
//-------------------------------------------------------------
    void Send_Command_LCD  (byte tmp)
    {
    while (Check_buttons())Delay(1000);
    RB1=0;
    PORTB=(PORTB & 0x0F)+ (tmp & 0xF0);
    Pulse(100);
    PORTB=(PORTB & 0x0F)+ (tmp<<4);
    Pulse(100);
    RB1=1; Delay(1000);
    }
//-------------------------------------------------------------
```

При записи данных в видеопамять ЖКИ (для отображения на дисплее), полезно завести специальную ячейку (*Current_ind*), которая показывала бы, в какое место экрана мы сейчас выводим символ. Если текущая позиция является последней в текущей (первой) строке, то необходимо выдать на ЖКИ команду перевода курсора в начало следующей (второй) строки. Приводимые ниже программные модули позволяют легко выполнять эти действия (*Send_Byte_LCD*), а также устанавливать курсор в нужное место дисплея (*Set_Coord_LCD*) или очищать экран дисплея (*Clr_LCD),* а также выводить на экран дисплей содержимое строковых констант (*Show_String_LCD*).

```
//=============================================
    void Send_Byte_LCD  (byte tmp)
    {while (Check_buttons())Delay(1000);
    PORTB=(PORTB & 0x0F)+ (tmp & 0xF0);
```

```
    Pulse(10);
    PORTB=(PORTB & 0x0F)+ (tmp<<4);
    Pulse(10);
    Current_ind++;
    if(Current_ind==16)Set_Coord_LCD(1,0);
    //if(Current_ind==32)Set_Coord_LCD(0,0);
    }
    //====================
    void Set_Coord_LCD(byte i, byte j)
    {
       if(i==0){Current_ind=j;Send_Command_LCD(0x80+j);}
       else {Current_ind=16+j;Send_Command_LCD(0xC0+j);};
          return;
    }
    //====================
    void Show_String_LCD(const char * mySTRING)
    {while(*mySTRING){Send_Byte_LCD(*(mySTRING++));}; }
    //====================
    static const char str_BLANK[] = "               ";
    void Clr_LCD(void)
    {
    Set_Coord_LCD(0,0);
    Show_String_LCD(str_BLANK);
    Show_String_LCD(str_BLANK);
    Set_Coord_LCD(0,0);
    }
//================================================
```

Работа с инфракрасным приемопередатчиком

Инфракрасный (ИК) приемопередатчик состоит из двух частей – передатчика, собранного на одном транзисторе T1 и интегрального трехвыводного приемника (см. Fig.4). С возможностями передающей схемы все ясно – транзистор открывается при подаче на его базу с выхода B0 высокого уровня напряжения, и чрез светодиод начинает течь ток. Светодиод начинает излучать свет в ИК диапазоне световых волн. При подаче низкого уровня напряжения с выхода B0 транзистор T1 закрывается, и светодиод гаснет. Таким образом, в нашем распоряжении простейший модулятор света, работающий в режиме “светит - не светит ”.

Приемники ИК излучения бывают разных типов, в зависимости от сигналов, которые они должны принимать. В компьютерной технике часто используют приемники IrDA сигналов, ориентированные на прием серий коротких импульсов с высокой скважностью. На базе этих приемников нетрудно обеспечить связь на небольшом расстоянии

(порядка метра) со скоростью до 4 Мб/сек. Подробную информацию по IrDA сигналам и приемникам легко найти в Интернете (см., например, http://www.hw.cz/english/docs/irda/irda.html).

Значительно более простые приемники используются в бытовой технике в системах дистанционного управления RC (remote control). Приемники RC предназначены для приема сигналов, имеющих вид пачки (пакета) импульсов излучения, причем длительность каждого импульса в пачке совпадает с длительностью паузы между импульсами. Число импульсов в пачке должно быть не слишком маленьким (обычно не менее десятка). Принято называть всю такую пачку "импульсом с высокочастотным заполнением". Частота заполнения обычно лежит в диапазоне 30-50 кГц. Типичное значение частоты – 36 кГц.

Использование высокочастотного заполнения позволяет значительно повысить помехоустойчивость системы связи, введя в состав приемника узкополосный фильтр, настроенный на частоту заполнения. После предварительного усиления и фильтрации сигнал поступает с выхода фильтра на выпрямитель-интегратор, и, затем, на квантователь, на выходе которого формируется логический сигнал. Разумеется, это лишь поверхностное описание только основных узлов приемника. Реальная схема должна устойчиво работать как при сильном входном сигнале так и при слабом, не реагировать на сильную засветку приемника посторонними источниками света, например, осветительными приборами, быть особенно устойчивой к мощным низкочастотным помехам частотой 100 Гц (люминесцентные лампы дневного света) и т.д.

Основные элементы внутренней структуры приемника приведены на Fig. 6.

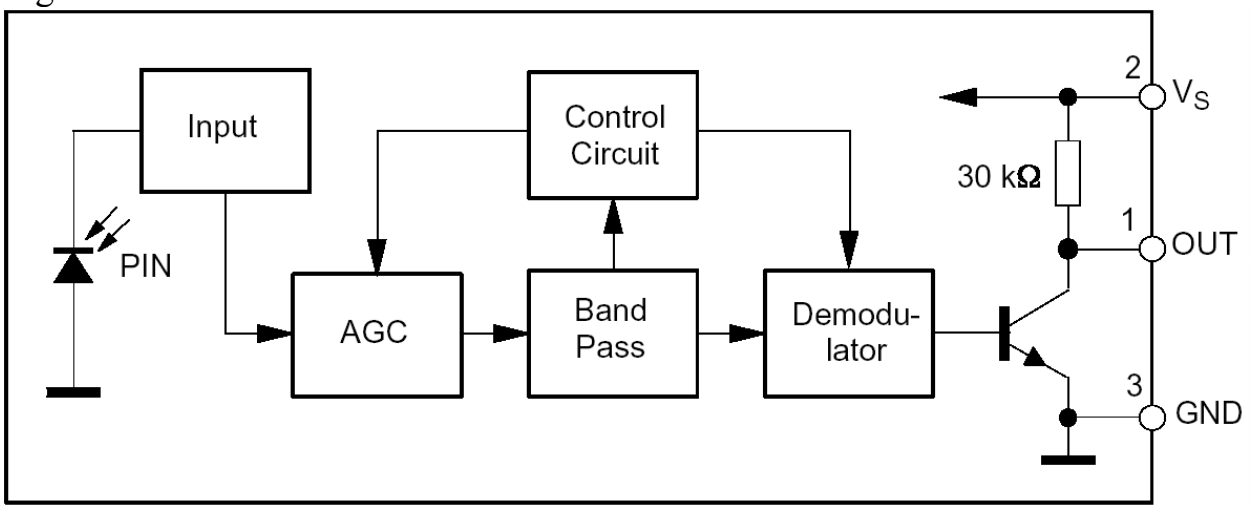
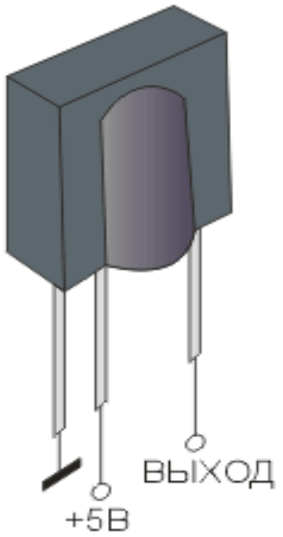


Fig. 6. Основные блоки интегрального ИК-приемника

Здесь **PIN** обозначает фотодатчик, называемый PIN-диодом, который отличается от обычного фотодиода повышенной чувствительностью и улучшенными частотными характеристиками. Input – входные цепи (предварительный усилитель); **AGS**- (automatic gain control) схема АРУ (автоматическая регулировка усиления); **Band Pass** – полосовой фильтр,

который пропускает сигнал вблизи частоты заполнения и отфильтровывает все остальное; **Control Circuit** – схема управления; **Demodulator** включает в себя интегратор и квантователь, а также (совместно с выходным транзистором) формирователь выходного логического сигнала .



Самое удивительное, что вся эта сложная схема на практике реализуется в виде маленького пластмассового “трехногого” модуля, на которой надо лишь подать питание и снять с его выхода готовый логический сигнал! Пластмассовый корпус попутно выполняет роль оптического фильтра который пропускает только ИК-диапазон и защищает входные цепи приемника от перегрузок даже при прямом солнечном освещении. Такие приемники производятся многими фирмами, например, SFH-506 фирмы Siemens, TFMS5360 фирмы Temic, ILMS5360 производства ПО «Интеграл» и др. Последняя микросхема применяется в нашей установке, но с равным успехом можно использовать и любые другие микросхемы. Обычно несущая частота указывается прямо в маркировке микросхемы. Число 36, присутствующее в обозначении схемы однозначно указывают на несущую частоту 36 кГц.

Итак, на первый взгляд все просто. В исходном состоянии входной сигнал отсутствует, выходной транзистор приемника закрыт, и на выходе схемы имеется высокий уровень напряжения за счет подтягивающего резистора. Теперь предположим, что появился входной сигнал требуемой частоты. После приема 6-10 импульсов несущей частоты, выходной транзистор открывается, и на выходе приемника появляется низкий уровень напряжения. Далее, если входной сигнал прекратился, то через 150-250 мкс после окончания действия полезного сигнала выходное напряжение возвратится к высокому уровню. Если входной сигнал продолжает поступать на приемник, то через некоторое время (0.1 .. 0.5 секунды) выход приемника все равно перейдет в состояние высокого выходного напряжения. Дело в том, что приемник не рассчитан на очень длинные пакеты несущей частоты. И если такой пакет появляется на входе приемника, он (через некоторое время) начинает восприниматься приемником как помеха. Большинство приемников RC имеют ограничения на соотношение между активным состоянием входного сигнала и пассивным. Это соотношение, как правило, не должно превосходить ¾. Для длинных пакетов (порядка 50-70мс) требуется еще более длительная пауза, по длительности равная или большая, чем сам пакет. Это очень важный нюанс, который необходимо учитывать при организации связи. Программа, под управлением которой будет работать ваш передатчик, должна анализировать этот параметр, и если активное состояние передатчика составляет более ¾ от пассивного (в среднем по

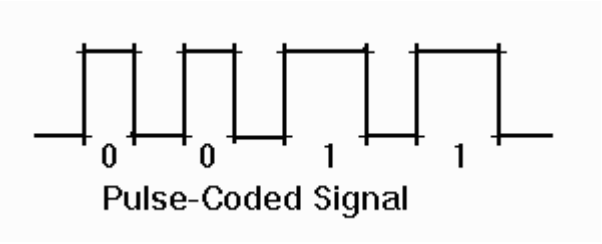
некоторому интервалу времени, скажем 0.1сек), то передатчик должен прервать передачу и дать возможность приемнику восстановиться и подготовиться к продолжению приема.

Во многих приложениях такой проблемы вообще не возникает. Например, если требуется передавать информацию о нажатых клавишах, то время передачи кода клавиши намного меньше времени между нажатиями различных клавиш, т.е. между передаваемыми пакетами информации имеются “естественные” длительные интервалы времени, в течение которых передатчик не работает. Вместе с тем, если требуется передать по ИК каналу связи какой-то достаточно большой блок информации, то обязательно надо контролировать описанное выше соотношение пассивного и активного времени работы передатчика и при необходимости вводить в передачу принудительные паузы. Конкретные значения временных параметров необходимо уточнять по документации завода-изготовителя.

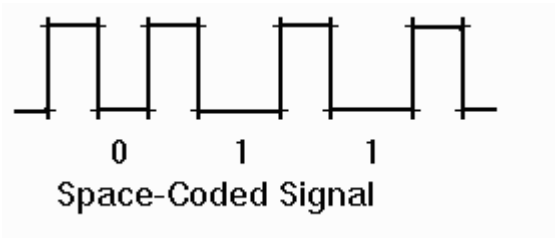
Высокая помехоустойчивость и чувствительность интегрального приемника имеет оборотную сторону. Максимальная скорость передачи, достижимая для такого приемника в старт-стопной асинхронной системе связи (типа RS-232), имеет величину порядка 1200 бод для непрерывного потока данных и порядка 2400 бод для коротких блоков данных. Можно перечислить несколько основных причин такой низкой скорости: медленная работа интегратора, накапливающего десяток импульсов до момента опознавания пачки и потом так же медленно возвращающегося к исходному состоянию, относительно низкая частота заполнения пачки, введение принудительных пауз между передаваемыми блоками данных.

Известно несколько способов передачи данных с помощью пакетов с высокочастотным заполнением.

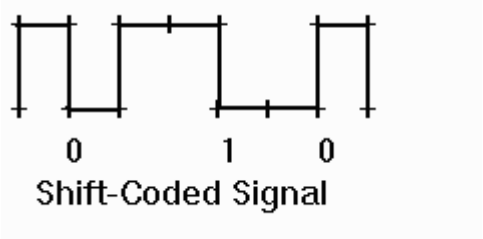
1. Широтно-импульсная модуляция. При этом виде модуляции ноль передается короткой пачкой, а единица – длинной пачкой.



2. Модуляция паузы между импульсами. Информация заключена в наличии короткой или длинной паузы между пачками.



3. Совмещенная модуляция, которая меняет длительность как пачки так и паузы после пачки в зависимости от передаваемого сигнала. Например, ноль передается как короткая пачка и короткая пауза, а единица как длинная пачка и длинная пауза.



Конкретный выбор типа модуляции зависит от особенностей решаемой задачи. Например, при реализации старт-стопной асинхронной системы связи (типа RS-232), можно каждый ноль заполнять высокочастотными импульсами, т.е. передавать в виде пачки такой же длительности, а единицу – в виде отсутствия пачки. Тогда выход приемника будет точно соответствовать входу высокочастотного модулятора передатчика.

Программная реализация такой системы связи предельно проста. Предположим, мы хотим реализовать передачу через ИК-канал связи информации о нажимаемых клавишах на клавиатуре PC с одного компьютера на другой, используя пару ЛОК. Один ЛОК выступает в качестве передатчика, а второй в качестве приемника. Соединяем ЛОК-передатчик с выходом COM-порта компьютера - источника информации. Запускаем на этом компьютере программу “HyperTerminal”, входящую в состав ОС Windows. Теперь при нажатии любой клавиши на клавиатуре компьютера, ASCII-код символа будет поступать на ЛОК в обрамлении стартового и стопового битов в соответствии с протоколом RS-232. Нетрудно написать программу для PIC-контроллера, которая будет постоянно опрашивать выход COM-порта и посылать пачку с высокочастотным заполнением на передатчик ИК в то время, когда из порта на контроллер поступает нулевой сигнал. Передатчик готов. Приемник сделать еще проще. Подключаем ЛОК-приемник к входу COM-порта второго компьютера, запускаем на этом компьютере программу “HyperTerminal”. PIC-контроллер должен постоянно опрашивать выход интегрального приемника и транслировать сигнал с выхода приемника на вход COM-порта второго компьютера. Соответствующая программа требует записи не более 15 строчек на ассемблере. Теперь следует выставить на обоих компьютерах одинаковую (небольшую) скорость передачи, и связь по ИК каналу связи готова! Все, что вы будете набирать на клавиатуре одного компьютера, будет отображаться на терминале другого компьютера, причем информация будет передаваться через оптический беспроводный канал ИК связи! Разумеется, нетрудно придумать более интересные системы, в которых кратковременное нарушение связи (закрыть рукой приемник) не будет приводить к потере передаваемой информации за счет вводимой в

информацию избыточности. К сожалению, освещение вопросов помехоустойчивого кодирования выходит за рамки данного материала.

Организация асинхронной последовательной связи с PC

Для связи с внешними устройствами по последовательному каналу связи (RS-232) используются разряды C6 и C7 порта C (см. Fig. 7).

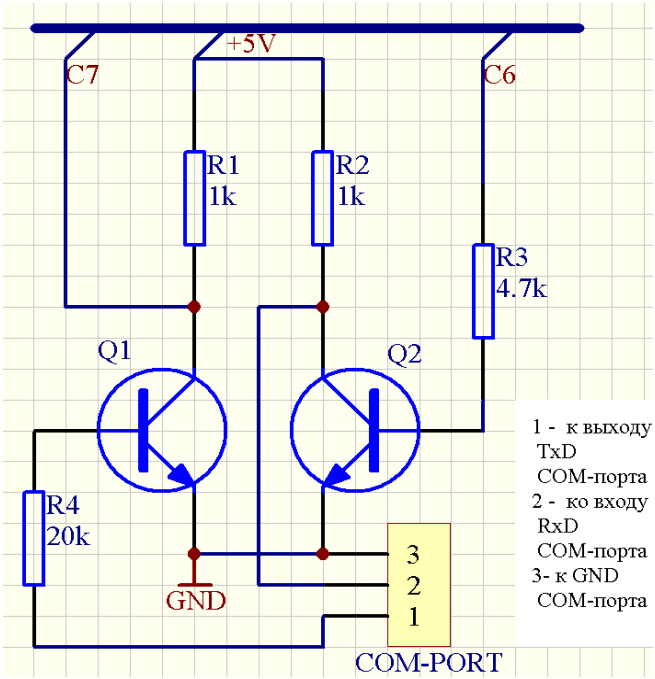


Fig. 7 Схема подключения разъема COM-PORT к порту C.

Как видно из рисунка, разряд C6 должен быть выходом, а C7- входом последовательного порта передачи данных микроконтроллера.

Возможны два способа организации последовательной связи в PIC-контроллере – программный и аппаратный. При аппаратной реализации используется модуль USART (Universal Synchronous Asynchronous Receiver Transmitter), имеющийся в составе внутренней архитектуры некоторых PIC-контроллеров. Для использования USART надо внимательно ознакомится с документацией для PIC-контроллера, с которым вы собираетесь работать. Многие PIC-контроллеры, особенно наиболее дешевые, не имеют в своем составе USAR, и для них возможен только программный способ реализации последовательной связи. Ниже мы рассмотрим именно программный способ, как наиболее универсальный. Принципы асинхронной старт-стопной передачи данных были подробно изложены в последней лабораторной работе предыдущего семестра (LabRab_5), поэтому здесь мы сразу переходим к программной реализации этих принципов. В приведенной ниже программе реализован передатчик данных. Базовым элементом передатчика является задержка (подпрограмма Delay_Fast ()), которая призвана формировать требуемый интервал времени, в течении которого будет передаваться

каждый элементарный сигнал (стартовый, стоповый биты и биты информации). Программа предельно проста и в пространных комментариях не нуждается. Настоятельно рекомендуется проверить длительность задержки, создаваемой подпрограммой Delay_Fast () в симуляторе MPLAB-а.

```
//-------------------------------------------------------------------------------
 void myTransmit_COM(byte i)
{
#define myH 0
#define myL 1
#define myD 0x65

RC6=myH;   //start bit
Delay_Fast(myD);

if(testbit(i,0)) RC6=myL; else RC6=myH; Delay_Fast(myD);
if(testbit(i,1)) RC6=myL; else RC6=myH; Delay_Fast(myD);
if(testbit(i,2)) RC6=myL; else RC6=myH; Delay_Fast(myD);
if(testbit(i,3)) RC6=myL; else RC6=myH; Delay_Fast(myD);
if(testbit(i,4)) RC6=myL; else RC6=myH; Delay_Fast(myD);
if(testbit(i,5)) RC6=myL; else RC6=myH; Delay_Fast(myD);
if(testbit(i,6)) RC6=myL; else RC6=myH; Delay_Fast(myD);
if(testbit(i,7)) RC6=myL; else RC6=myH; Delay_Fast(myD);

RC6=myL; //stop bits
Delay_Fast(myD);
Delay_Fast(myD);
}
//-------------------------------------------------------------------------------
```

Приемник строится по аналогичному принципу, с той лишь разницей, что вначале следует выполнить задержку на половину длительности элементарного сигнала, чтобы регистрировать 0 или 1 в середине этого сигнала, что повышает надежность приемника. А затем нужно делать задержку на один элементарного сигнал, чтобы попасть в середину следующего элементарного сигнала.

Приемник также следует проверить в симуляторе, учитывая, что дополнительную задержку создают операции, связанные с упаковкой принимаемых битов в байт, входы-выходы в подпрограммы задержки и т.д. Эти небольшие добавки могут давать значительную погрешность в определении середины элементарных сигналов для последних принимаемых битов (эффект накапливания ошибки). Особенно внимательно надо проверить временные характеристики приемника для высоких скоростей передачи (57600, 115200 бит/сек). Программу приемника напишите самостоятельно.

Работа с I2C шиной

По протоколу связи Inter-Integrated Circuit (I^2C) имеется большое количество литературы и ссылок в Интернете. Мы остановимся здесь лишь на общих принципах связи по этому протоколу и рассмотрим элементы программной реализации этого протокола для случая, когда микроконтроллер работает в режиме “Master”.

С аппаратной точки зрения шина I2C представляет собой два проводника, “подтянутые” к +5V резисторами с сопротивлением в несколько кОм. (см. Fig. 8)

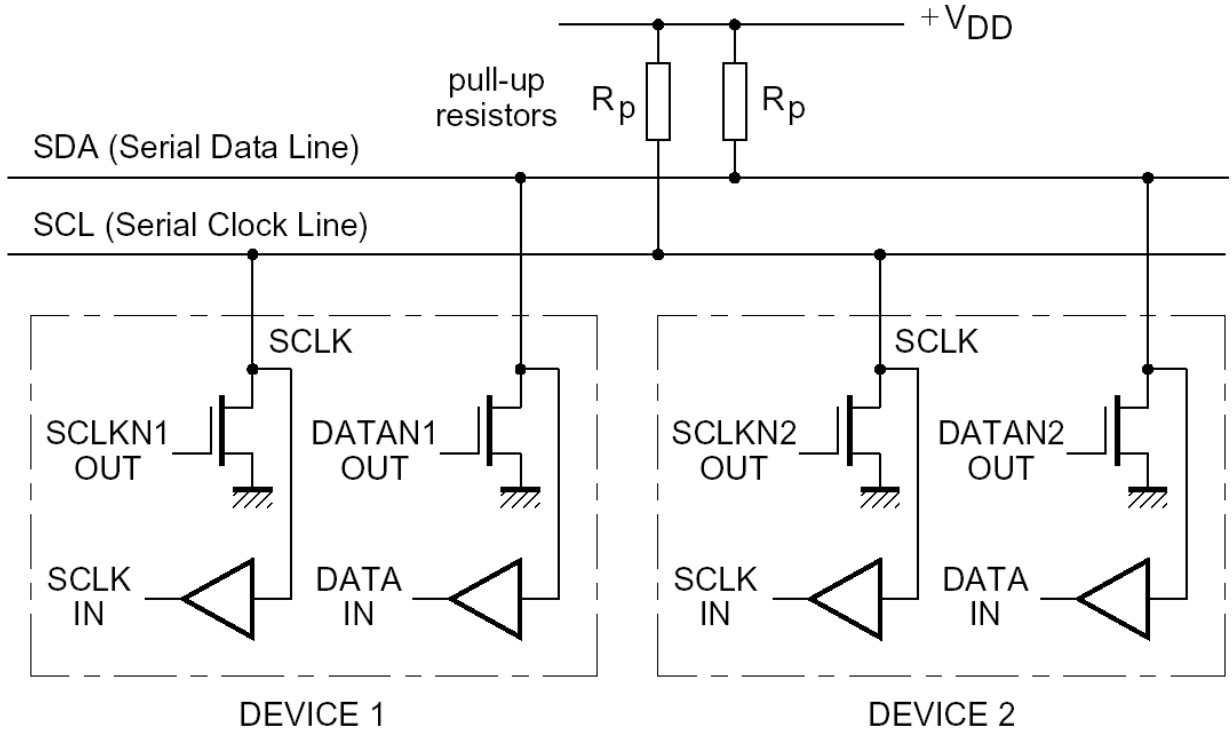


Fig. 8. Подключение периферийных устройств к I2C шине.

Все устройства, подключенные к этой шине, имеют выходные каскады типа “открытый коллектор” или “открытый сток”. Когда все выходные транзисторы всех устройств закрыты – на линии присутствует высокое напряжение, а когда хотя бы один транзистор открыт – низкое. Одна из линий отвечает за передачу импульсов синхронизации (SCL), а другая (SDA) – за передачу данных между устройствами, подключенными к шине. В нашем макете линия SCL соединена с разрядом RC3 порта C, а линия SDA – с разрядом RC4 порта C.

В рассматриваемом нами режиме работы инициатором обмена всегда является микроконтроллер, он называется Master – хозяин или ведущий), а периферийные элементы лишь отвечают на запросы микроконтроллера (они называются Slave – раб - ведомый). Синхросигнал по линии SCL всегда формируется контроллером независимо от того, куда передаются данные – из

контроллера или в контроллер. Каждый бит передаваемой или принимаемой информации сопровождается синхроимпульсом по линии SCL. Если передача информации ведется от внешнего устройства в микроконтроллер, то и в этом случае устройство выставляет на линию данных очередной бит информации только по началу синхроимпульса, полученного от микроконтроллера.

Общий принцип организации связи по I2C очень прост. Сначала микроконтроллер передает в шину I2C специальную комбинацию сигналов на линиях SCL и SDA, которая называется стартовой комбинацией или просто сигналом START. Сигнал START – это просто перевод линии SDA из состояния высокого напряжение в нулевое состояние в то время как на SCL присутствует высокий уровень напряжения. Сигнал STOP – это перевод линии SDA из нулевое состояние в состояние высокого напряжения в то время, как на SCL присутствует высокий уровень напряжения. Сигналы START и STIOP – особые синалы, для которых разрешено изменение значение SDA в то время, как SCL находится в единичном состоянии. Для всех остальных режимов работы I2C допускается изменение SDA только при нулевом значении SCL (см. Fig. 9).

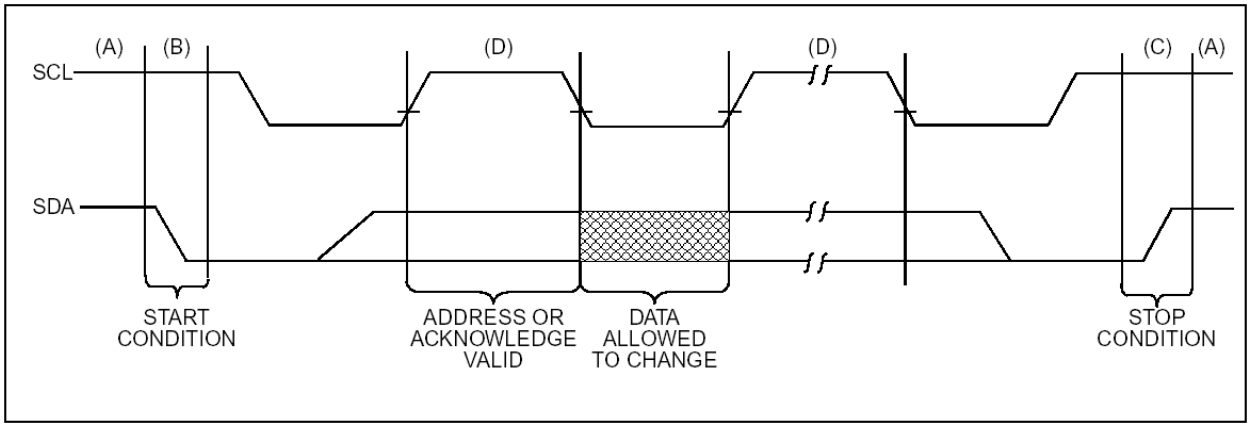


Fig. 9 Формирование основных сигналов на линиях SCL и SDA

Сигнал START говорит всем периферийным устройствам, что сейчас к кому-то из них будет произведено обращение. Теперь микроконтроллер передает по I2C байт данный, который мы ниже будем называть запросным. Этот запросный байт содержит информацию о “номере” периферийного устройства, с которым желает связаться микроконтроллер. Этот номер (также часто называемый адресом slave-устройства) принимается всеми устройствами, подключенными к шине I2C. Каждое периферийное устройство “знает”, какой за ним закреплен номер и сверяет этот номер с тем адресом, который содержится в запросном байте. Если эти адреса совпали, то устройство готово к дальнейшему обмену данными с контроллером. Если нет, то устройство “отключается” от I2C до тех пор, пока там не появится следующий сигнал START. Выбранное устройство “отзывается” на

обращение к нему сигналом ACK - выдает нулевой бит на SDA во время действия девятого синхроимпульса. Напомним, что первые 8 синхроимпульсов соответствовали передаче запросного байта (см. Fig. 10).

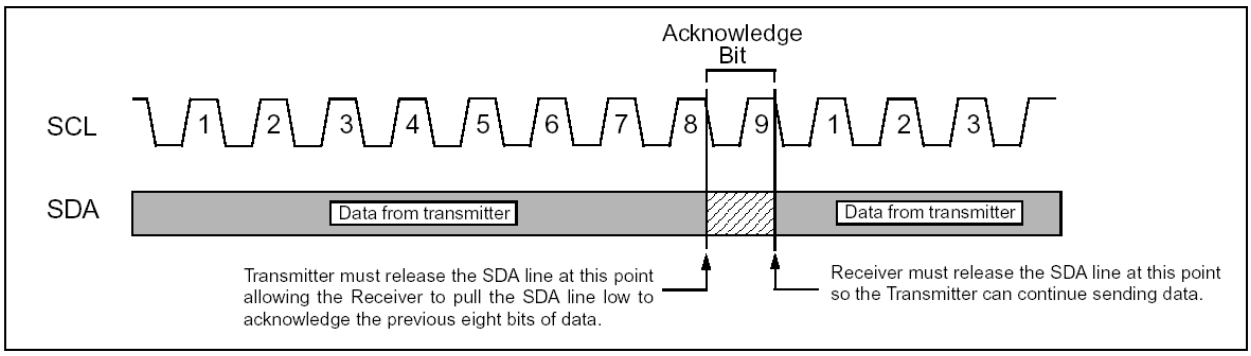


Fig. 10 Формирование сигнала подтверждения АСК

Сигнал АСК (acknowledge - подтверждение) говорит о нормальном завершении приема байта данных. Микроконтроллер должен освободить линию данных перед приемом ACK, чтобы периферия могла выдать нулевой бит на эту линию. Под освобождением линии понимается перевод выходного каскада в закрытое состояние, т.е. на линии появляется высокий уровень напряжения за счет “подтягивающего” резистора. Далее последовательность сигналов START+запросный байт+ACK и сопровождающие эти сигналы синхроимпульсы мы будем называть запросным пакетом.

В запросном байте последний (младший) бит содержит информацию о том, какое действие хочет совершить микроконтроллер – передачу данных в периферийное устройство или чтение данных из периферийного устройства. Если последний бит запросного байта нулевой, то это запрос на запись, в противном случае – на чтение.

Заметим, что в процессе работы микроконтроллера с выбранным периферийным устройством сигнал START сформироваться “случайно” не может в принципе. Поэтому одновременное включение в работу нескольких периферийных устройств совершенно исключено.

Структура запросного байта, таким образом, формируется из двух частей – адресной части и бита направления передачи данных. Адресная часть, в свою очередь, состоит из двух полей – поле типа устройства (4 бита) и поле номера конкретной микросхемы (3 бита). Дело в том, что на одной шине I2C могут “висеть” несколько однотипных микросхем, и их надо как-то отличать друг от друга. Например, двоичный код 1010 закреплен фирмами-производителями последовательной памяти за любыми устройствами последовательной памяти. Таких устройств на одной шине I2C может быть до 8 штук, т.к. отводится три бита на адрес конкретного устройства этого типа.

Следует пояснить, что все микросхемы последовательной памяти “знают”, что их код 1010. Кроме того, у каждой микросхемы имеется три “ножки”, на которые заводятся высокие или низкие уровни сигналов с внешних цепей общей схемы. Обычно эти уровни не меняются и жестко

определяются разводкой печатной платы и сапой принципиальной схемой. Вот эти-то три входных сигнала и составляют вторую часть адреса каждой конкретной микросхемы. Разумеется, все однотипные микросхемы, подключенные к одной шине I2C, должны иметь различные адреса во втором поле адреса slave-устройства. Структура запросного пакета представлена на Fig.11. Напомним, что сигнал ACK поступает от периферии (автоматически), остальные сигналы – от микроконтроллера, т.е. должны формироваться программой пользователя.

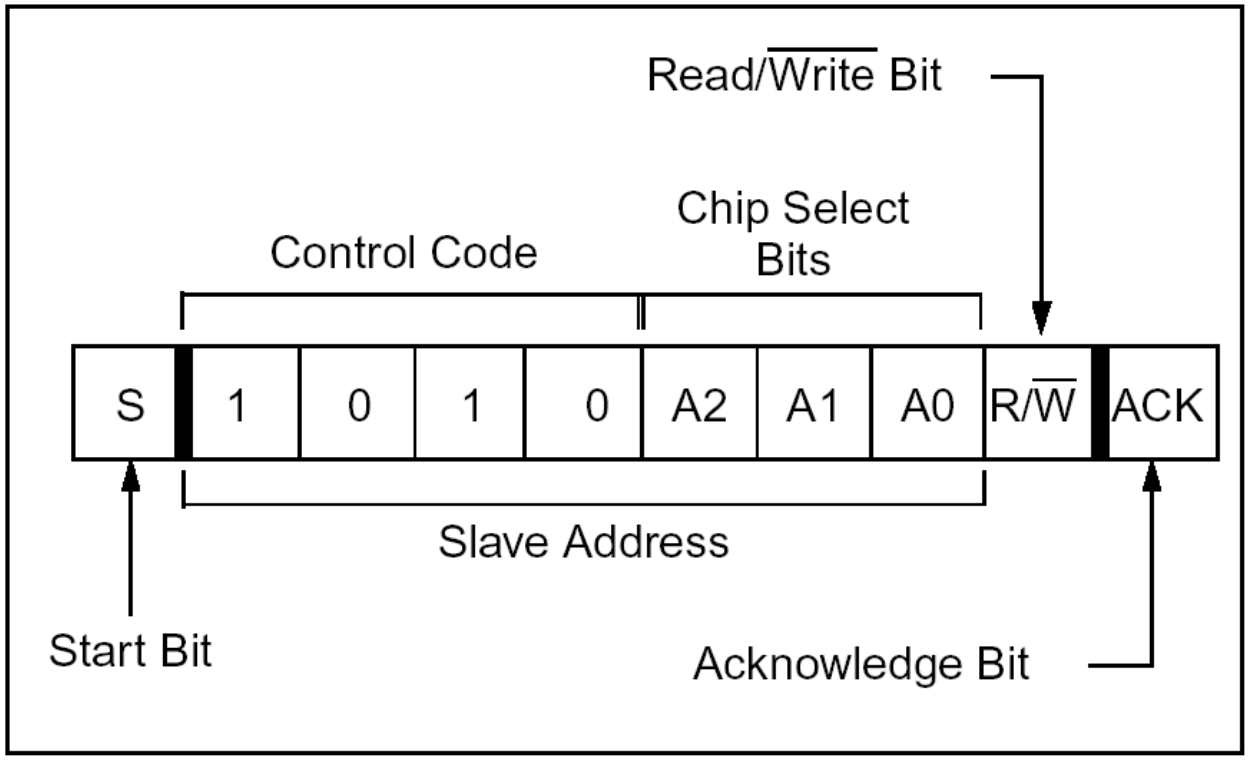


Fig.11 Структура запросного пакета (линия данных).

Рассмотрим теперь дальнейший обмен данными между микроконтроллером и выбранным периферийным устройством. Их дальнейшее взаимодействие зависит от конкретного типа периферийного устройства. Здесь мы рассмотрим обмен данными с микросхемой последовательной памяти 24LC256 фирмы Microchip и с микросхемой датчика температуры (двоичный код типа устройства - 1001) фирмы Dallas (теперь принадлежащей фирме Maxim).

Работа с последовательной памятью

Запись в память отдельного байта.

Вслед за запросным байтом микроконтроллер должен передать два байта адреса ячейки последовательной памяти, к которой выполняется обращение. Каждый такой байт сопровождается восемью синхроимпульсами, и девятый импульс дает возможность принять от периферии сигнал ACK.

Если все в порядке (все ACK приняты контроллером), то следующим шагом микроконтроллера будет передача байта, предназначенного для записи. После принятия байта данных периферия должна выдать ACK, после чего контроллер завершает связь с микросхемой памяти, сформировав на линиях SCL и SDA специальную комбинацию сигналов, которая называется стоповой комбинацией или просто сигналом STOP. Процесс записи одиночного байта по указанному адресу проиллюстрирован рисунком Fig.12

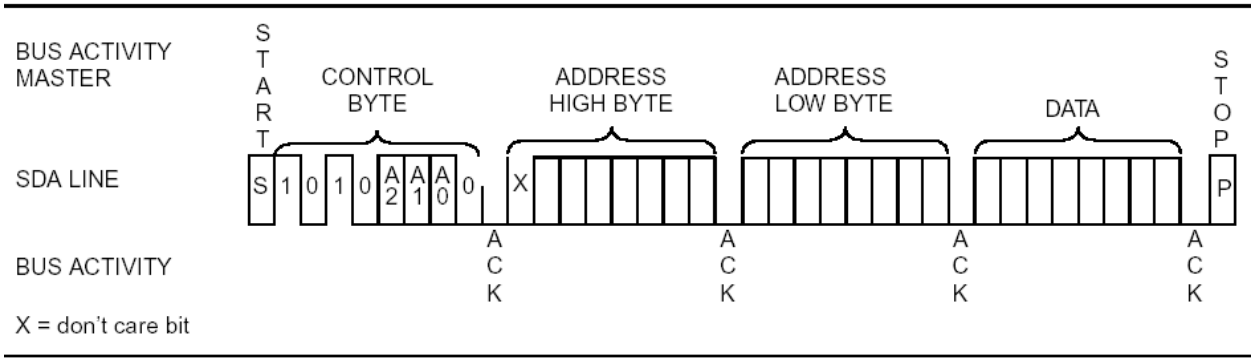


Fig.12 Запись одиночного байта

После получения сигнала STOP, микросхема памяти на некоторое время становится недоступной для связи. Дело в том, что в действительности прием байта микросхема памяти производит во внутренний промежуточный буфер, и только после получения сигнала STOP происходит физическая запись информации в основную память микросхемы. Запись информации требует порядка 5 миллисекунд (10мс для старых версий микросхем). Это весьма значительный интервал времени, если учесть, что передача синхроимпульсов может вестись с частотой до 400кГц (до 100кГц для старых микросхем), т.е. передача данных в микросхему происходит весьма быстро. У пользователя есть два пути для продолжения работы с памятью – ждать 5-10 мс и затем обращаться к памяти с запросным пакетом, либо постоянно посылать запросный пакет до тех пор, пока микросхема памяти не откликнется сигналом ACK.

Запись в память последовательности байт (пакетный режим записи).

Как мы отмечали выше, запись информации по одному байту требует достаточно длительного времени. Для ускорения процесса записи в составе микросхемы памяти имеется специальный буфер, в который происходит предварительная запись получаемой от контроллера информации. Запись последовательности байт в принципе аналогична записи одного байта с той лишь разницей, что микроконтроллер не передает сигнал STOP после первого байта, а просто продолжает передачу, посылая второй байт, третий и т.д. (не забывая получать ACK в конце каждого байта). Когда буфер в микросхеме памяти заполняется полностью, необходимо прекратить передачу и выдать сигнал STOP, чтобы инициализировать физическую запись буфера в память. Интересно, что время записи информации из буфера

в память практически не зависит от степени заполнения буфера, т.е. на запись одного байта из буфера потребуется примерно столько же времени, что и для записи полного буфера (в нашем случае буфер имеет размер 64 байта). Процесс записи одиночного байта по указанному адресу проиллюстрирован рисунком Fig.13

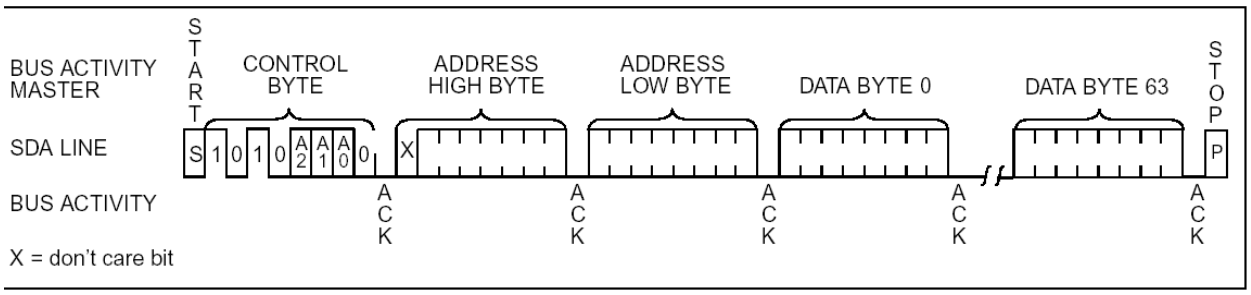


Fig.13 Запись последовательности байт.

В работе с буфером имеются некоторые тонкости. Эти тонкости связаны с тем, что передаваемая информация не обязательно начинает заполнять буфер с его начала.

Вкратце этот процесс можно описать следующим образом. Предположим, что мы начали запись информации в микросхему последовательной памяти с адреса 30. Этот адрес был передан в микросхему памяти, и ее внутренний счетчик адреса установился в указанное значение. Теперь мы начали записывать байты один за другим в микросхему памяти (а точнее – в ее буфер) и дошли до конца буфера. Когда это произойдет? Это произойдет через 34 записи, т.к. длина буфера в нашем случае – 64 байта, а начали мы его заполнять не с начала, а с адреса 30. После записи 34-х байт следует сформировать сигнал STOP, и из буфера будут записаны в память переданные из микроконтроллера 34 байта. Каким образом можно определить, в какое место буфера происходит текущая запись? Можно считать, что **младшие 6 битов адреса ячейки памяти, куда производится запись, являются адресом ячейки памяти в буфере, куда заносится текущий байт**. Когда в процессе записи микроконтроллер доходит до последней ячейки буфера, младшие биты адреса будут иметь вид 111111. После записи байта в эту последнюю ячейку буфера, адрес имеет в младших битах все нули. Это и является признаком того, что нужно прекратить передачу и очистить буфер, выдав STOP.

У программиста нет прямого доступа к внутреннему счетчику адреса микросхемы памяти. Однако, программа пользователя может иметь свою собственную переменную - аналог внутреннего счетчика адреса микросхемы памяти. Программа должна инкрементировать значение этой переменной с каждой новой записью байта данных. Таким образом, программа пользователя может самостоятельно определять заполнение буфера и выдавать сигнал STOP в нужный момент времени.

Затем можно снова начать запись, отправив в микросхему памяти новый начальный адрес записи, который, кстати, будет иметь младшие разряды вида 000000, т.е. соответствует началу буфера. Если дальше передача ведется длинной последовательностью подряд идущих байт, то перезапись будет происходить уже по целому буферу , а не по его части, что заметно повышает среднюю скорость записи информации в память.

Чтение одного байта или нескольких последовательно расположенных (в адресном пространстве) байт из памяти.

При чтении данных из памяти надо выполнить следующую, на первый взгляд не совсем естественную процедуру. А именно, надо сначала сделать вид, что мы собираемся выполнять запись байта в указанный адрес, а затем переключиться на чтение данных.

Такая “хитрая” операция необходима для того, чтобы установить внутренний счетчик адреса микросхемы памяти в нужное значение. Чуть более подробно эта процедура (с точки зрения микроконтроллера) выглядит так:

1. Выдать запросный пакет (с признаком записи информации). Получить ACK.
2. Выдать два байта адреса. На каждый получить ACK.
3. Выдать запросный пакет (с признаком чтения информации). Получить ACK.
4. Принять байт информации из памяти. Далее возможны два варианта. Если микроконтроллер собирается продолжить чтение данных, то необходимо выдать (из микроконтроллера !) сигнал ACK - подтверждение принятия байта из памяти. По этому сигналу микросхема памяти подготовит для выдачи следующий байт. Затем контроллер переходит к началу пункта 4 и т.д.

Если контроллер не собирается продолжать чтение следующего байта, он должен вместо ACK выдать специальную комбинацию сигналов на линиях SCL и SDA, которая называется сигналом NACK (отсутствие подтверждения). Далее контроллер формирует STOP и заканчивает сеанс связи с периферией. Описанные выше варианты чтения иллюстрируются рисунками Fig.14 и Fig.15

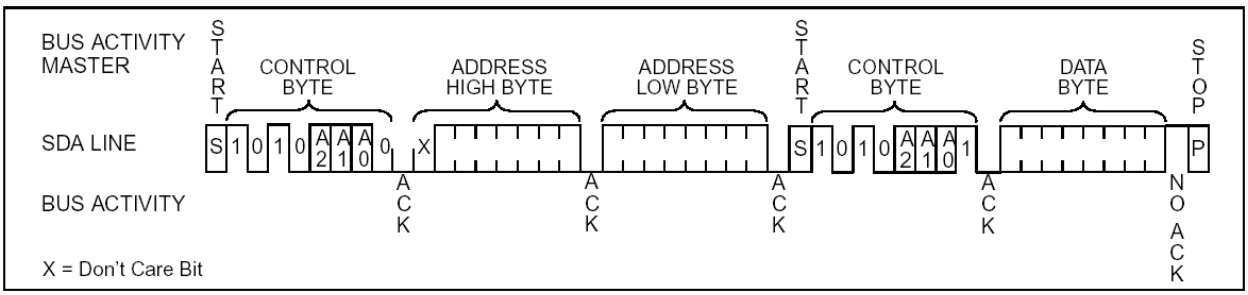


Fig.14 Чтение одного байта.

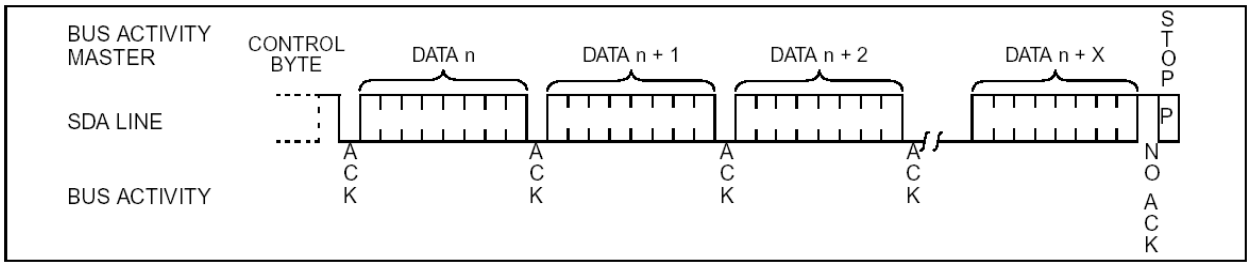


Fig.15 Чтение последовательности байт.

Чтение из памяти происходит очень быстро и никаких специальных задержек не требует. Фактически быстродействие при чтении ограничивается только максимальной частотой подачи синхроимпульсов, о которой мы упоминали выше. Отметим, что у разных микросхем памяти может быть различный размер буфера. Обычно, чем меньше объем памяти, тем меньше размер буфера. Для уточнения размера буфера у каждого конкретного типа памяти следует обратиться к техническому описанию данной микросхемы.

Теперь коротко сформулируем основные моменты, которые необходимо понимать при работе с последовательной памятью.

1. Имеются два специальных сигнала, один из которых начинает сеанс связи с периферией (сигнал START), второй заканчивает сеанс связи (сигнал STOP).
2. Вслед за сигналом START всегда следует запросный байт с адресом slave-устройства и признаком записи/считывания информации, говорящим о дальнейших намерениях микроконтроллера, инициирующего сеанс связи.
3. Далее из микроконтроллера в память передаются два байта адреса, которые должны указать ячейку памяти с которой контроллер собирается выполнить какие-то действия.
4. Каждая передача данных из контроллера в память должна завершаться передачей от памяти к микроконтроллеру специального сигнала ACK, подтверждающего успешный прием микросхемой памяти байта информации от контроллера. Аналогично, микроконтроллер должен формировать ACK при каждом получении байта данных от микросхемы памяти. Имеется одно исключение – выдача контроллером сигнала NACK в случае, если контроллер хочет прервать сеанс последовательного чтения данных из памяти.

Понимание этих основных принципов и внимательное изучение приведенных выше рисунков позволяет успешно реализовать операции обмена данными с последовательной памятью. Правила формирования специальных сигналов START,STOP,ACK,NACK весьма просты и хорошо изложены в полном описании сигналов протокола I2C, содержащемся в файле I2C_BUS_SPECIFIC.pdf, с которым рекомендуется ознакомиться

самостоятельно. В заключение заметим, что некоторые PIC-контроллеры имеют встроенные аппаратную поддержку шины I2C, что позволяет написать более эффективную программную библиотеку для обмена данными с последовательной памятью и несколько облегчить работу программиста, возложив большинство операции на аппаратную часть PIC-контроллера.

И последнее замечание. Последовательная память 24LC256 и аналогичная ей память другой емкости и других фирм предназначена в первую очередь для хранения и чтения данных и только во вторую очередь – для записи. Ресурс микросхемы памяти по чтению практически неограничен. А вот ресурс по количеству записи в одну ячейку памяти (по данным фирм-изготовителей) может составлять "всего" сто тысяч перезаписей. На первый взгляд это очень много, но легко подсчитать, что при желании этот ресурс легко исчерпать за несколько дней работы прибора. А вот ресурс по времени хранения записанной информации действительно большой - может достигать 200 лет! (Проверить самостоятельно ;-).

Управление по шине I2C работой датчика температуры

Микросхема DS1621 (Digital Thermometer and thermostat) имеет несколько режимов работы. Эти режимы включаются путем занесения специальных управляющих байт во внутренние регистры DS1621. Доступ к внутренним регистрам датчика температуры осуществляется по шине I2C. Двоичный код типа устройства – 1001, номер устройства – 001, т.е. адрес slave-устройства в двоичном виде имеет вид 1001001. Связь по шине I2C практически не имеет никаких особенностей по сравнению с рассмотренным выше примером. Однако, специфика самого устройства измерения температуры сказывается на порядке передачи в микросхему тех или иных управляющих байт. Эти особенности касаются как бы "верхнего уровня" протокола обмена информации. На нижнем уровне (способ передачи или приема байта данных, формирование старта, стопа, подтверждений) организация связи полностью аналогична стандартному протоколу I2C.

Подробное описание возможностей микросхемы и правил ее программирования приведено в документации фирмы-производителя (файл – 1621.pdf). Здесь мы приведем лишь основные характеристики и опишем тот минимум управляющих воздействий , который необходим для извлечения информации о температуре из внутренних регистров микросхемы.

DS1621 позволяет измерять температуру окружающей среды с точностью 0.5 град. Цельсия. Диапазон измеряемых температур – от -55 до +125 градусов. Данные о температуре представляются в виде 9 бит и передаются из DS1621в виде двух байт. Первый байт содержит величину целой части температуры. Старший бит второго байта содержит дробную часть температуры. Остальные биты второго байта всегда нулевые. Если температура отрицательная, то 9-ти битное число представлено в дополнительном коде. В следующей таблице приведены несколько примеров представления температуры в виде двух байт, получаемых из DS1621.

TEMPERATURE	DIGITAL OUTPUT (Binary)
+125°C	01111101 00000000
+25°C	00011001 00000000
$+^{1}/_{2}$°C	00000000 10000000
+0°C	00000000 00000000
$-^{1}/_{2}$°C	11111111 10000000
–25°C	11100111 00000000
–55°C	11001001 00000000

Минимальное время преобразования, необходимое для очередного измерения температуры – около одной секунды (незначительно зависит от напряжения питания, температуры и т.д.). Возможны два режима работы – с внешним запуском измерения или с периодическим автоматическим запуском измерителя с периодом около секунды. В первом случае необходимо на микросхему выдать специальную команду (байт данных через I2C), который инициирует запуск измерителя. Затем, через секунду можно извлечь из DS1621 результат измерения. Во втором случае можно время от времени “запрашивать” DS1621 о последнем результате измерений. Первый случай более экономичный, второй – более простой.

Имеется специальный режим работы DS1621, при котором микросхема программируется один раз, а затем может функционировать без всяких дополнительных воздействий со стороны микроконтроллера. При этом DS1621 будет периодически (раз в секунду) измерять температуру и сравнивать ее с двумя порогами – верхним и нижним. Если верхний порог превышен, то на специальном выводе микросхемы уровень сигнала становится высоким или наоборот низким – по вашему желанию (это указывается при программировании режима). Если же температура стала ниже нижнего порога, то уровень напряжения на выводе микросхемы меняется на противоположенный. Такой режим работы схемы называется термостатированием. Верхний и нижний пороги заносятся в DS1621 микроконтроллером при программировании режима работы. Этот режим очень удобен при построении автономных систем поддержания температуры в заданных пределах, т.к. он требует однократного программирования DS1621, который далее может работать совершенно автономно, поскольку все настройки режимов сохраняются в его внутренней энергонезависимой памяти.

Информация, управляющая режимом работы DS1621, хранится в регистре конфигурации:

CONFIGURATION/STATUS REGISTER

DONE	THF	TLF	NVB	1	0	POL	1SHOT

Назначение разрядов конфигурационного слова:

DONE

Если при чтении конфигурационного слова имеем DONE=1, это означает, что последнее измерение температуры завершено, если DONE=0, то датчик еще не готов к выдаче температуры.

THF - Temperature High Flag.

Если при чтении конфигурационного слова имеем THF =1, это означает, что после включения питания хотя бы один раз температура превысила верхний порог. Этот бит может быть обнулен прямой записью в конфигурационное слово.

TLF - Temperature Low Flag.

Если при чтении конфигурационного слова имеем TLF =1, это означает, что после включения питания хотя бы один раз температура снизилась ниже нижнего порога.Этот бит может быть обнулен прямой записью в конфигурационное слово.

NVB - Nonvolatile memory busy

Если при чтении конфигурационного слова имеем NVB =1, это означает, что микросхема занята записью данных в энергонезависимую память.
(Например, происходит запись величин температурных порогов в память).
Следует подождать, пока NVB не станет равным нулю.

POL - Output Polarity

Устанавливает полярность выходного сигнала при срабатывании термостата.

1SHOT

Если 1SHOT установить в 1 , то это будет означать однократный режим работы датчика, т.е. каждое измерение будет производиться по специальной команде, поступающей по I2C. Если 1SHOT сбросить в 0, то включится режим непрерывного преобразования (с периодом около секунды).

Рассмотрим обмен информацией между микроконтроллером и датчиком температуры. Все обмены данными инициируются микроконтроллером. Начинаются все пересылки одинаково:

1) Выдать START, послать запросный байт с признаком записи, получить ACK
2) Послать код операции (один байт), получить ACK

Код операции говорит датчику температуры о том, какое действие собирается совершить микроконтроллер. Удобно оформить эту последовательность действий в виде одной подпрограммы, имеющей параметром код операции. Примерный вид подпрограммы приведен ниже. В ней используются простейшие действия с шиной I2C, которые были рассмотрены при работе с последовательной памятью.

```
//---------------------------------------
    void Start_Thermo ( byte code )
    { START_I2C(); OUT_BYTE_I2C(0x92); OUT_BYTE_I2C(code); }
//---------------------------------------
```

Вот несколько основных кодов операций (полный список см. в документации):

0xAC - Access Config - доступ к конфигурационному регистру
0xEE - Start Convert - начать измерение температуры
0x22 - Stop Convert - остановить измерение температуры
0xA1 - Access TH - доступ к регистру с верхним пороговым значением температуры
0xA2 - Access TL - доступ к регистру с нижним пороговым значением температуры
0xAA - Read Temperature - чтение последнего результата измерения температуры

После посылки кода операции возможны различные развития событий в зависимости от кода операции. Например, если мы хотим записать новое значение в конфигурационный регистр, то после передачи 0xAC (доступ к конфигурационному регистру), следующим байтом надо просто передать новое значение конфигурационному регистра и закончить связь сигналом STOP. Для начальной инициализации микросхемы DS1621 надо выполнить следующие действия:

1) Загрузить требуемой константой конфигурационный регистр
2) Загрузить регистры верхнего и нижнего порогов.
3) Выдать команду на запуск преобразования (в автоматическом режиме)
4) Периодически считывать результаты измерений температуры и, например, выводить их на индикатор.

Первые три пункта можно реализовать следующим образом.

```
1) Start_Thermo(0xAC);
   OUT_BYTE_I2C(0x0); Delay(600);
   STOP_I2C(); Delay(6000);
2)
   Start_Thermo(0xA1);
   OUT_BYTE_I2C(30); Delay(60000);
   OUT_BYTE_I2C(0x0); Delay(60000);
   STOP_I2C(); Delay(60000);
        Delay(60000);
   Start_Thermo(0xA2);
   OUT_BYTE_I2C(29); Delay(60000);
   OUT_BYTE_I2C(0x0); Delay(60000);
    STOP_I2C(); Delay(60000);
3)
Start_Thermo(0xEE);
STOP_I2C(); Delay(60000);
```

Четвертый пункт реализуйте самостоятельно, используя следующие рисунки (см. Fig.16), поясняющие порядок обращения к DS1621 в различных ситуациях.

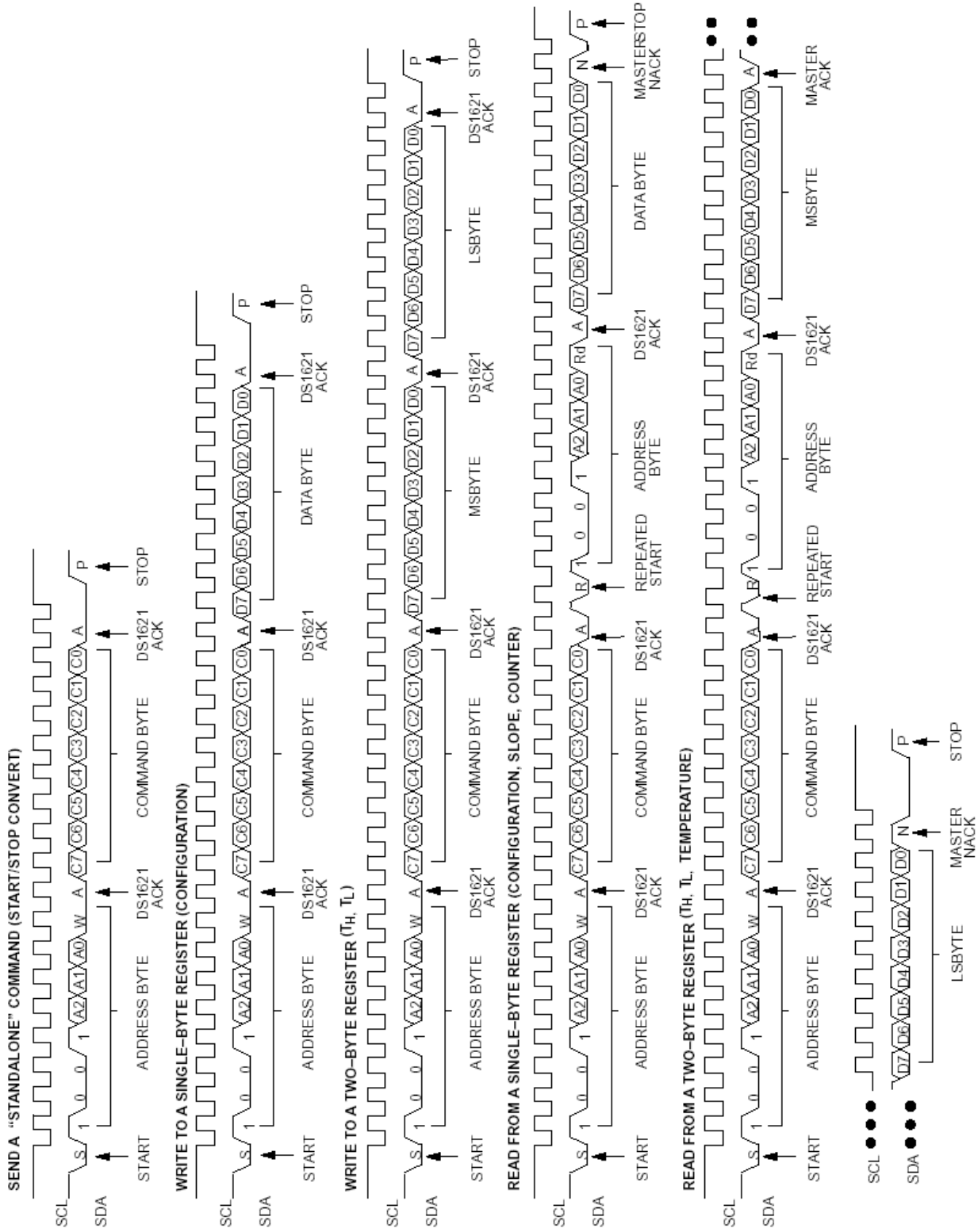


Fig.16 Структура сигналов на шине I2C в различных режимах работы датчика температуры.

Работа с встроенным программатором

Следует запустить программное обеспечение встроенного программатора EPIC и внимательно изучить его возможности, посмотреть установку различных режимов его работы. Рекомендуется установить автоматическое обновление загружаемого файла и конфигурационного слова. В этом случае вам потребуется только один раз (после запуска программы программатора) считать из hex-файла его содержимое, а далее при каждом новом программирование будет автоматически загружаться последняя версия этого файла. Разумеется, текст вашей программы должен содержать указание конфигурационного слова с помощью оператора __CONFIG().

Таким образом, работа с ЛОК выглядит так : вы редактируете в MPLAB свою программу, выполняете компиляцию проекта, и если не было ошибок просто нажимаете кнопку программирования в окошке программатора (и предварительно кнопку “Prog” на плате ЛОК). Через несколько секунд новая версия программы будет занесена в PIC-контроллер и начнет работать сразу после отпускания кнопки “Prog”.

Обратите внимание на то, чтобы конфигурация, которую всегда можно посмотреть в соответствующем окне программы EPIC, соответствовала той конфигурации, которую вы пытались установить с помощью оператора __CONFIG(). Типичной причиной проблем с программированием является неправильная работа с конфигурационными битами. Например, пользователь забыл установить автоматическое обновление конфигурации, и в контроллере была прошита некорректная комбинация конфигурационных бит. Это может привести к “зависанию” контроллера и к последующим проблемам с дальнейшей прошивкой даже правильной комбинации бит. В этом случае следует выключить питание ЛОК, вытащить кварцевый резонатор, включить питание и выполнить корректную прошивку. После этого можно вставить кварц и вернуться к обычному режиму работы с ЛОК.

Если к ЛОК через внешние разъемы подключены какие–то дополнительные блоки, то они также могут быть причиной проблем с программированием. Внешние схемы желательно отключать на время записи программы в контроллер.

Приложение 1.

//==========

Обзор ЖКИ с сайта www.gaw.ru

ЖК-модули фирмы Powertip

Базирующаяся в Тайване фирма Powertip производит широкую гамму жидкокристаллических индикаторов и оснащенных контроллерами или драйверами алфавитно-цифровых и графических модулей на их основе.

Алфавитно-цифровые ЖКИ-модули выполнены на базе контроллера HD44780, или его аналогов (в том числе русифицированных) и позволяют управлять ЖКИ, содержащими до 80-ти символов, существуют модули, содержащие два контроллера HD44780, способные управлять 160-ю символами. Алфавитно-цифровые ЖКИ-модули выпускаются нескольких форматов: однострочные, двустрочные и четырехстрочные, по 8, 12, 16, 20, 24, и 40 символов в строке, нескольких габаритных размеров и посадок в рамках одного формата, всего 38 различных конструктивов. Габаритные размеры модулей изменяются от 55,7X32 мм (2 строки по 12 символов), до 280X88 мм(4 строки по 40 символов), размеры символов от 2,7X5,5 мм, до 6X14,5 мм.

Кроме того, ЖКИ-модули могут иметь ЖКИ различных конструкций (TN, STN, FSTN), обладающих различными потребительскими свойствами (контрастность, угол обзор, цвет фона и знаков), а также могут быть оснащены задней подсветкой на основе светодиодов различных цветов, электролюминесцентной панели или люминесцентной лампы с холодным катодом. ЖКИ-модули производятся в двух климатических исполнениях: с обычным диапазоном температур - 0*C…60*C, и с расширенным диапазоном температур - -20*C…70*C.

Алфавитно-цифровые ЖКИ-модули представляют собой недорогое и удобное решение, позволяющее сэкономить время и ресурсы при разработке новых изделий, при этом обеспечивают отображение большого объема информации при хорошей различимости и низком энергопотреблении.

Основные области применения ЖКИ-модулей: измерительные приборы, медицинское оборудование, промышленное оборудование, информационные системы, аппаратура с автономным питанием.

Значительно большей гибкостью обладают графические ЖКИ-модули. В отличие от алфавитно-цифровых модулей, жестко фиксирующих размеры и положение символов, графические модули не накладывают сколь либо серьезных ограничений на отображаемую информацию, причем это могут быть не только символы алфавита, но и спецзнаки, графики, диаграммы, элементы оформления.

В выпускаемую фирмой Powertip номенклатуру графических ЖКИ-модулей входят модули 22-х конструктивов и имеющие форматы матриц от 80X80 точек, до 320X240 точек. ЖКИ-модули различных форматов могут содержать разные контроллеры, отличающиеся функциональными возможностями и предельными размерами управляемой матрицы точек, либо совсем не содержать контроллера, а только микросхемы драйверов, требующие внешних схем развертки. Последнее в основном относится к крупноформатным ЖКИ-модулям: 256X128, 320X240. Фирма Powertip применяет 4 типа контроллеров для графических ЖКИ-модулей: SED1520, HD61202, LC7981 и T6963C. Первые два наиболее простые и не содержат средств аппаратной поддержки вывода символов, LC7981 более сложный контроллер, позволяющий работать как в графическом, так и в текстовом режимах, T6963C позволяет одновременно отображать символьную и графическую информацию. В любом случае, можно получить отображение произвольных символов чисто программными средствами.

Как и алфавитно - цифровые, графические ЖКИ-модули выпускаются в различных модификациях: ЖКИ STN и FSTN, задняя подсветка на основе светодиодов, электролюминесцентной панели или люминесцентной лампы с холодным катодом, а также в вариантах с обычным диапазоном температур - 0*C…60*C, и с расширенным диапазоном температур - -20*C…70*C.

Основные области применения графических ЖКИ-модулей аналогичны областям применения алфавитно-цифровых: измерительные приборы, медицинское оборудование, промышленное оборудование, информационные системы. Кроме того, нужно отметить, что в области компактной индикации у графических ЖКИ-модулей фактически нет альтернативы. Интересен также вариант применения графических модулей фактически в качестве алфавитно-цифровых, но с увеличенным размером знаков (при малом количестве разрядов), чему способствует наличие достаточного количества ЖКИ-модулей с малоформатными матрицами и компактными габаритами.

//=======================================