МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
Московский государственный технологический университет «СТАНКИН»

Кафедра «Робототехника и мехатроника»

Поливанов А.Ю.

Методическое пособие по курсу

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА В МЕХАТРОНИКЕ И РОБОТОТЕХНИКЕ

Москва 2016

1. Однокристальный микроконтроллер 8051

1.1 Общие сведения

Однокристальный микроконтроллер 8051 был выпущен фирмой Intel в 1980 г. и получил широкое распространение во всем мире как наиболее удачный универсальный восьмиразрядный микроконтроллер. В отличие от многих других устройств с течением времени он не был снят с производства, а прочно занял нишу простого распространенного восьмиразрядного устройства, имеющего открытую архитектуру. Эта архитектура носит название mcs-51. Лицензию на производство данного микроконтроллера получили многие фирмы, выпускающие электронные компоненты.

С течением времени структура ядра и система команд микроконтроллера осталась неизменной, а усовершенствование пошло по пути добавления новых устройств, усовершенствованию памяти программ (использование электрически стираемой Flash памяти) и повышению тактовой частоты (до 25 МГц).

В настоящее время наиболее распространены разновидности микроконтроллеров, выпускаемых фирмой Atmel: AT89C51, AT89S51, AT89C2051, AT89C52. Кроме того, существуют модели микроконтроллеров, которые содержат в своем составе дополнительные устройства: ЦАП, АЦП, последовательный интерфейсы, дополнительные таймер-счетчики и т.п.

Область применения микроконтроллера 8051 очень широка благодаря своей универсальности и функциональности, но в первую очередь он удобен для построения простых систем управления и сбора информации.

Микроконтроллер 8051 удобно использовать в процессе обучения микропроцессорной технике, т. к. он имеет простую структуру, содержит небольшое количество периферийный устройств и регистров конфигурации, обладает удобной, интуитивно понятной системой команд. Обучение программированию на ассемблере не занимает много времени.

В рамках лабораторного практикума изучается построение системы управления шаговым двигателем и организации ввода информации с клавиатуры. В этих заданиях, проектируемые устройства будут являться частью многопроцессорной системы.

Структура однокристального контроллера 8051 показана на рис.1.

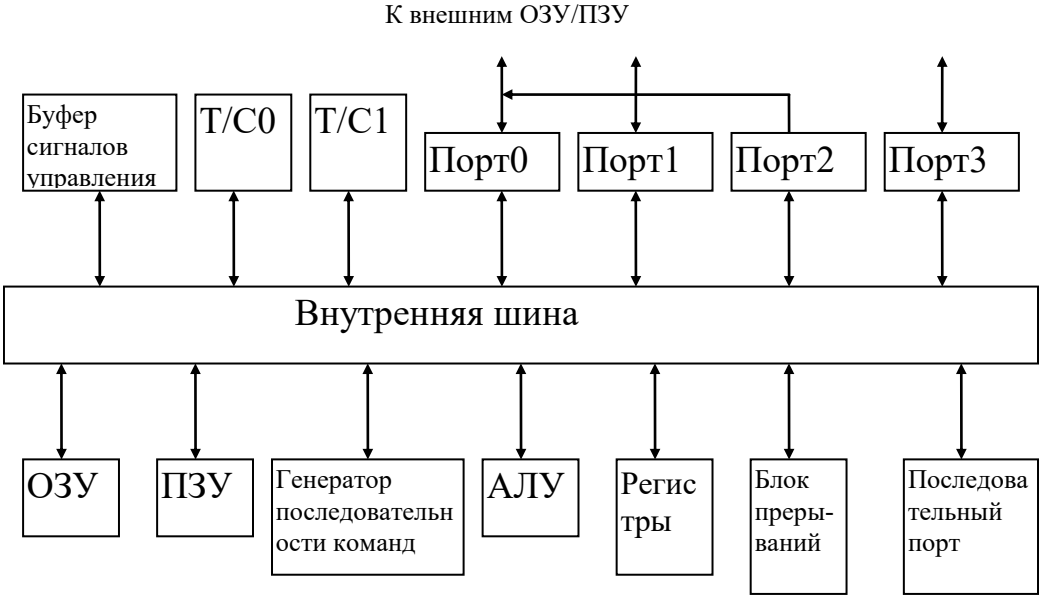

Рис.1 Структурная схема однокристального микроконтроллера 8051

Здесь Т/С0 и Т/С1 – таймер-счетчики, предназначенные для подсчета количества импульсов или задания временных интервалов, ОЗУ – резидентное (внутреннее) оперативное запоминающее устройство емкостью 128 байт, ПЗУ – резидентное (внутреннее) постоянное запоминающее устройство емкостью 4Кбайт, АЛУ – арифметико-логическое устройство, с помощью которого выполняются арифметические и логические операции. К регистрам относятся так называемые регистры общего назначения (РОН), предназначенные для хранения операндов (данных, используемых в операциях) и их адресов, т.е. номеров ячеек памяти, где эти операнды хранятся. Всего может быть использовано 8 РОН (R0 - R7), а также ряд специальных регистров. Изображение микроконтроллера на принципиальных электрических схемах представлено на рис.2. Микроконтроллер представляет собой микросхему с сорока выводами.

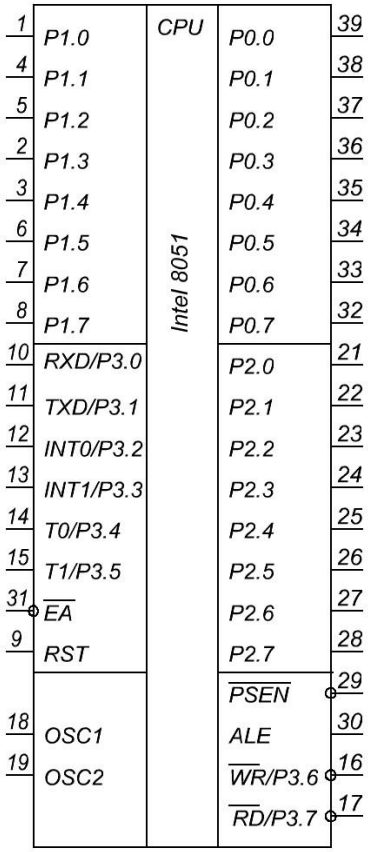

Рис.2 Графическое обозначение микроконтроллера

Выводы однокристального микроконтроллера имеют следующее назначение (инверсно активные сигналы подчеркнуты):

P0.0 - P0.7 (выводы 39 - 32) **–** порт P0, предназначен для подключения внешней памяти программ и/или данных (для систем с внешней памятью) или для ввода/вывода сигналов.

P1.0 - P1.7 (выводы 1 - 8) – порт P1, предназначен для ввода/вывода сигналов.

P2.0 - P2.7 (выводы 21 - 28) **–** порт P2, предназначен для выдачи старших разрядов адреса для систем с внешней памятью или для ввода/вывода сигналов.

P3.0 - P3.7 (выводы 10 - 17) – порт P3, используется для ввода/вывода сигналов или как специальные входы/выходы (RXD и TXD - вход и выход последовательного порта, INT0 и INT1 **–** входы сигналов внешних прерываний, T0 и T1 – входы таймеров/счетчиков, WR и RD – сигналы записи и чтения для внешней памяти).

ALE (вывод 30) – выход разрешения записи адреса, в момент среза (перехода из 1 в 0) этого сигнала при работе с внешней памятью. В этот момент на выводы портов P0 и P2 выставляется адрес.

PSEN (вывод 29) – выход сигнала считывания команд из внешней памяти, его активный уровень (0) разрешает считывание команды.

EA (вывод 31) – вход разрешения работы с внешней памятью программ, его активный уровень (0) разрешает работу с внешней памятью программ.

RST (вывод 9) – вход перезапуска микроконтроллера, после прихода импульса положительной полярности на этот вход происходит очистка регистров (см. режим перезапуска) и начинается выполнение программы с нулевого адреса;

OSC1, OSC2 (выводы 18, 19) – выводы для подключения кварцевого резонатора, LC-цепочки или внешнего генератора для задания тактовой частоты.

RXD, TXD (выводы 10, 11) – вход и выход последовательного порта (если он используется в системе) или выводы P3.0 и P3.1.

INT0, INT1 (выводы 12, 13) – входы внешних прерываний (если они используется в системе) или выводы P3.2 и P3.3.

T0, T1 (выводы 14, 15) – входы счетчиков (если они используется в системе) или выводы P3.4 и P3.5.

WR, RD (выводы 16, 17) – выходы сигналов записи и считывания для внешней памяти данных (если она используется в системе) или выводы P3.6 и P3.7.

Тактовая частота работы микроконтроллера обычно задается внешним кварцевым резонатором, подключаемым к выводам OSC1 и OSC2, и ограничивается верхним пределом 12 МГц (для некоторых модификаций микроконтроллера может быть выше).

Машинный цикл микроконтроллера 8051 состоит из 12 периодов тактовой частоты, разделенных на 6 состояний S1 - S6. Временная диаграмма показана на рис. 3.

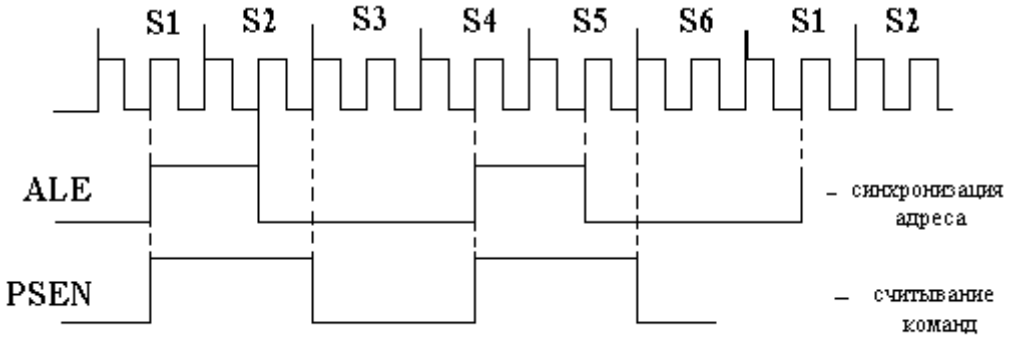

Рис. 3 Диаграмма машинного цикла микроконтроллера

Сигнал **ALE** используется для синхронизации адреса, его задний фронт сигнализирует о том, что на выводы портов P0 и P2 при обмене информацией с внешней памятью выставлен адрес. Сигнал **PSEN** предназначен для считывания команд из внешней памяти программ.

1.2 Регистры микроконтроллера

Микроконтроллер имеет в своем составе ряд программно доступных регистров, к числу которых относятся перечисленные ниже регистры.

A – аккумулятор, используется как регистр для хранения данных и результата операций во многих командах. Многие команды требуют, чтобы один из операндов находился в аккумуляторе и туда же записывается результат операции, таким образом, аккумулятор – это наиболее часто используемый регистр. Аккумулятор, как и многие другие регистры, допускает обращение к отдельным битам, которые нумеруются с 0 по 7, например младший бит аккумулятора обозначается как ACC.0, а старший – ACC.7.

B – расширитель аккумулятора, используется для расширения разрядности аккумулятора в операциях умножения и деления, либо как обычный регистр.

PSW – регистр состояния программы (Program Status Word), содержит информацию о результатах операций и управляет переключением банков РОН.

Назначение разрядов регистра PSW показано ниже.

PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
CY	**AC**	**F0**	**RS1**	**RS0**	**OV**	**-**	**P**

CY – флаг переноса.

AC – флаг дополнительного переноса.

F0 – флаг пользователя.

RS1, RS0 – флаги, определяющие выбор банка регистров общего назначения.

RS1	**RS0**	Банк
0	0	0
0	1	1
1	0	2
1	1	3

OV – флаг переполнения.

P – флаг четности.

R0 - R7 – регистры общего назначения (РОН), используются для хранения операндов. Всего можно использовать до 4-х банков по 8 РОН, их можно переключать, используя 3-ий и 4-ый биты регистра состояния программы PSW.4 и PSW.3 (RS1, RS0).

SP – указатель стека, содержит адрес последней заполненной ячейки стека. В качестве стека, т.е. памяти, организованной по принципу LIFO, служит фрагмент резидентной памяти данных (РПД).

DPTR – указатель внешней памяти данных, в него записывается 16-ти разрядный адрес ячейки внешней памяти данных, к которой производится обращение. Состоит из двух 8-ми разрядных регистров **DPL** и **DPH**.

TL0, TH0 – регистры младших и старших разрядов таймер-счетчика 0.

TL1, TH1 – регистры младших и старших разрядов таймер-счетчика 1.

TMOD – регистр режимов работы таймер-счетчиков.

TCON – регистр управления и статуса таймер-счетчиков.

SBUF – буферный регистр последовательного порта (по записи – буфер передатчика, по чтению - буфер приемника).

SCON – регистр управления и статуса последовательного порта.

IE – регистр масок прерываний, с его помощью разрешаются или запрещаются прерывания.

IP – регистр приоритетов прерываний, позволяет устанавливать для каждого прерывания высокий или низкий уровень приоритета.

P0 - P3 – регистры портов микроконтроллера. Доступны по чтению и записи для ввода и вывода данных через порты микроконтроллера.

1.3 Организация памяти

Адресное пространство микроконтроллера составляет 64 Кбайт памяти программ (ПП) и 64 Кбайт памяти данных (ПД). Резидентная, т.е. внутренняя, память программ (РПП) составляет 4 Кбайта и может быть либо однократно программируемая, либо многократно программируемая с ультрафиолетовым стиранием, либо флэш-память. Современные модификации микроконтроллера используют резидентную Flash память программ. Внешняя память программ (ВПП) обычно представляет собой внешнее постоянное запоминающее устройство (ПЗУ) емкостью до 64 Кбайт. Выбор типа используемой памяти программ (РПП или ВПП) производится с помощью сигнала на вход **EA** (низкий активный уровень выбирает ВПП).

Структура ПП представлена в таблице 1.

Таблица 1 Структура памяти программ микроконтроллера

Адрес	Команда	Примечание
0h	ljmp staddr	переход на начало программы
3h	ljmp pint0	переход на подпрограмму обработки внешнего прерывания 0
Bh	ljmp ptf0	переход на подпрограмму обработки прерывания от Т/С 0
13h	ljmp pint1	переход на подпрограмму обработки внешнего прерывания 1
1Bh	ljmp ptf1	переход на подпрограмму обработки прерывания от Т/С 1
23h	ljmp prs	переход на подпрограмму обработки прерывания от последовательного порта
. . .	. . .	. . .
(staddr)	. . .	начало основной программы

По нулевому адресу ПП передается управление после снятия сигнала RESET, поэтому там располагается команда безусловного перехода на стартовый адрес основной программы, который обозначен staddr. Начиная с адреса 3 располагается область векторов прерываний. Если какое-либо прерывание используется в системе, то по соответствующему ему адресу должна располагаться команда безусловного перехода (ljmp) на стартовый адрес подпрограммы обработки этого прерывания.

Временная диаграмма работы с ВПП показана на рис. 4.

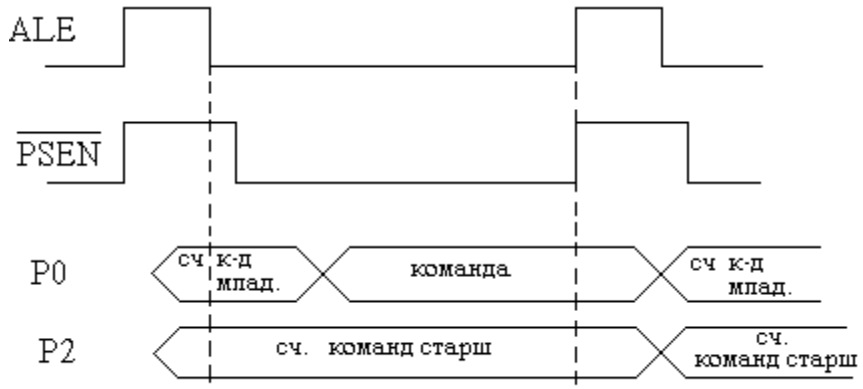

Рис. 4 Временная диаграмма работы микроконтроллера с ВПП

Память данных состоит из резидентной (внутренней) памяти данных (РПД) и внешней памяти данных (ВПД), если она используется. РПД содержит 128 байт, в число которых входят РОН, стек и 128 прямоадресуемых бит, используемых в командах с побитной адресацией. Структура РПД показана в таблице 2.

Таблица 2 Структура памяти данных микроконтроллера

Адреса	Содержимое
0 - 7	0 - банк РОН
7 - F	1 - банк РОН
10-17	2 - банк РОН
18-1F	3 - банк РОН
20-2F	область побитной адресации
30-7F	область побайтной адресации
80	порт P0
88	регистр TCON
90	порт P1
98	регистр SCON
A0	порт P2
A8	регистр IE
B0	порт P3
B8	регистр IP
D0	регистр PSW
E0	аккумулятор A
F0	расширитель B

Временные диаграммы работы с ВПД показаны на рис. 6.

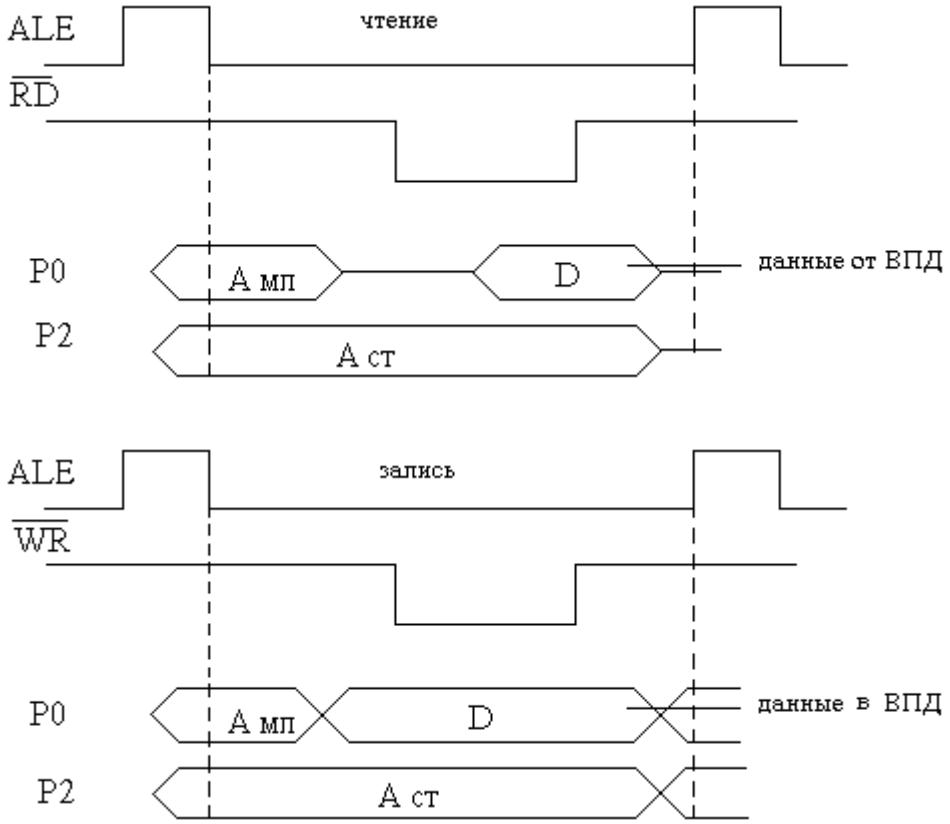

Рис. 6 Временные диаграммы работы микроконтроллера с ВПД

1.4 Таймер-счётчики

В состав однокристального микроконтроллера входят два программируемых 16-ти битных таймер-счётчика (T/C0 и T/C1), которые могут быть использованы в качестве таймеров или счётчиков внешних событий. В режиме счётчика к содержимому T/C прибавляется единица при появлении перехода из 1 в 0 (заднего фронта) внешних импульсов, подаваемых на соответствующий (T0,T1) вывод микроконтроллера. В режиме таймера содержимое T/C инкрементируется в каждом машинном цикле (через каждые 12 периодов тактовой частоты). Длительность импульса на внешнем входе должна быть больше длительности машинного цикла.

Для задания режимов работы T/C используется регистр режима работы T/C (**TMOD**).

Регистр режима работы T/C TMOD

Назначение разрядов регистра TMOD показано ниже.

TMOD.7	TMOD.6	TMOD.5	TMOD.4	TMOD.3	TMOD.2	TMOD.1	TMOD.0
GATE 1	**C/T1**	**M11**	**M10**	**GATE0**	**C/T0**	**M01**	**M00**

GATE1 – управление блокировкой T/C 1;

0 – разрешение работы от бита TR1;

1 – разрешение работы от бита TR1, при 1 на входе INT1.

C/T1 – выбор режима таймера или счётчика;

0 – таймер;

1 – счётчик.

M11, M10 – режимы работы T/C 1;

M11	**M10**	Режим
0	0	0
0	1	1
1	0	2
1	1	3

GATE0 – управление блокировкой T/C 0,

0 – разрешение работы от бита TR0,

1 – разрешение работы от бита TR0, при 1 на входе INT0;

C/T0 – выбор режима таймера или счётчика;

0 – таймер;

1 – счётчик.

M00, M01 – режимы работы T/C 0;

M01	**M00**	Режим
0	0	0
0	1	1
1	0	2
1	1	3

Для обоих T/C режимы работы 0,1 и 2 одинаковы. Режимы 3 различны.

Режим 0

Этот режим сделан для обеспечения совместимости с микроконтроллером предыдущего поколения 8048. T/C в этом режиме представляет собой 8-ми битный счётчик, на вход которого подключён 5-ти битный предделитель частоты на 32. Общая разрядность регистров T/C в этом режиме составляет 13 бит. **TL** работает как 5-ти битный предделитель. **TH** - как 8-ми битный счётчик. Установка бита **GATE** в 1 позволяет использовать таймер для измерения длительности импульсного сигнала, подаваемого на вход запроса прерывания.

Режим 1

Совпадает с режимом 0, но общая разрядность регистров составляет 16 бит. 16-ти битный таймер-счётчик. **TH1** и **TL1** включены последовательно.

Режим 2

Переполнение 8-ми битного счётчика младших разрядов **TL** приводит к установке флага **TF** и автоматически перезагружает в **TL** содержимое старших разрядов (**TH**) таймерного регистра, которое предварительно было задано программным путём.

Режим 3

В этом режиме T/C0 и T/C1 работают по-разному. T/C1 сохраняет неизменным своё текущее содержимое, т.е. таймер-счётчик 1 останавливается. **TH0** и **TL0** функционируют как два независимых 8-битных счётчика. Работу **TL0** определяют управляющие биты T/C0 (**C/T0, GATE0, TR0**), входной сигнал **INT0** и флаг переполнения **TF0**. Работу **TH0**, который может выполнять только функции таймера определяет управляющий бит **TR1**. При этом **TH0** использует флаг переполнения **TF1**.

Регистр управления/статуса таймер-счетчиков TCON

Регистр управления и статуса таймеров/счетчиков предназначен для их включения и выключения, анализа состояния и настройки режима прерывания.

TCON.7	TCON.6	TCON.5	TCON.4	TCON.3	TCON.2	TCON.1	TCON.0
TF1	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**

TF1 – флаг переполнения таймера 1. Устанавливается аппаратно при переполнении таймера-счётчика. Сбрасывается при обслуживании прерывания аппаратно.

TR1 – бит управления таймера 1. Устанавливается/сбрасывается программой.

TF0 – флаг переполнения таймера 0. Устанавливается аппаратно при переполнении таймера-счётчика. Сбрасывается при обслуживании прерывания аппаратно.

TR0 – бит управления таймера 0. Устанавливается/сбрасывается программой.

IE1 – флаг фронта прерывания 1. Устанавливается аппаратно когда детектируется срез внешнего сигнала **INT1**. Сбрасывается при обслуживании прерывания.

IT1 – бит управления типом прерывания 1. Устанавливается/сбрасывается программно для спецификации запроса (срез/низкий уровень).

IE0 – флаг фронта прерывания 0. Устанавливается аппаратно по срезу внешнего сигнала **INT0**. Сбрасывается при обслуживании прерывания.

IT0 – бит управления типом прерывания 1. Устанавливается/сбрасывается программно для спецификаци запроса (срез/низкий уровень).

1.5 Последовательный порт

Для связи системы, построенной на основе однокристального микроконтроллера, с другими устройствами и системами, а также для организации локальной сети, удобнее всего использовать последовательный порт. Через последовательный порт осуществляется приём и передача информации, представленной в последовательном коде. В состав последовательного порта, входят буферный регистр **SBUF** приёмопередатчика, являющийся программно доступным, а также принимающий и передающий сдвиговые регистры, которые скрыты от пользователя. Буферный регистр приемопередатчика аппаратно представляет собой два буферных регистра: буферный регистр приемника, доступный по чтению и буферный регистр передатчика, доступный по записи. Запись байта в буфер приводит к автоматической переписи байта в сдвигающий регистр передатчика и инициирует начало передачи байта. После окончания передачи байта устанавливается флаг передатчика **TI**, который говорит о том, что порт готов к передаче следующего байта. После приема байта принимающим сдвиговым регистром он автоматически появляется в буферном регистре. При этом устанавливается флаг приемника **RI**, сигнализирующий о том, что в буферном регистре находится байт информации, готовый к считыванию. Если к моменту окончания приёма байта предыдущий байт не был считан из **SBUF**, то он также будет потерян. Последовательный порт может работать в четырёх режимах, выбор которых осуществляется с помощью регистра управления/статуса **SCON**. Этот регистр содержит также 9-ый бит принимаемых или передаваемых данных (**RB8** и **TB8**), если выбран соответствующий режим работы, и флаги приёмопередатчика (**RI** и **TI**). Временная диаграмма сигнала последовательного порта показана на рис. 5.

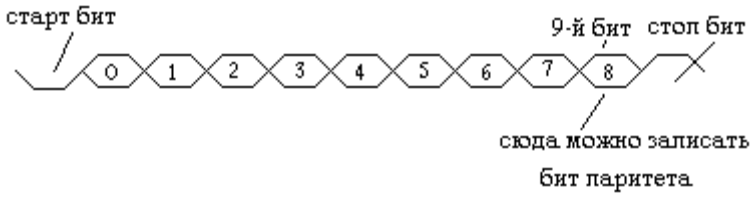

Рис. 5 Временная диаграмма сигнала последовательного порта

Регистр управления/статуса последовательного порта SCON

Для задания режимов работы и управления последовательным портом используется регистр **SCON.**

SCON.7	SCON.6	SCON.5	SCON.4	SCON.3	SCON.2	SCON.1	SCON.0
SM0	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**

SM0, SM1 – биты управления режимом работы порта, задают двоичный номер режима его работы.

SM1	**SM0**	Режим
0	0	0
0	1	1
1	0	2
1	1	3

SM2 – бит, устанавливаемый программно для запрета приёма сообщения, в котором девятый бит имеет значение 0. Используется для организации сети из нескольких устройств соединенных с помощью последовательного порта.

REN – бит разрешения приёма, устанавливается программно для разрешения приёма данных.

TB8 – передача бита 8, устанавливается/сбрасывается программно для задания девятого передаваемого бита в режиме 9-ти битной посылки.

RB8 – приём бита 8, устанавливается аппаратно для приема девятого принимаемого бита в режиме 9-ти битной посылки.

TI – флаг прерывания передатчика, устанавливается аппаратно при окончании передачи байта, сбрасывается программно после обслуживания соответствующего прерывания.

RI – флаг прерывания приёмника, устанавливается аппаратно при приёме байта, сбрасывается программно после обслуживания прерывания.

Режим 0

Информация передаётся и принимается через внешний вывод входа приёмника RXD. Принимаются или передаются 8 бит данных. Через внешний вывод выхода передатчика (TXD) выдаются синхросигналы сдвига, которые сопровождают каждый бит. Поэтому данный режим называется синхронным режимом работы. Частота передачи бита информации равна 1/12 частоты резонатора.

Режим 1

В этом режиме передаются через TXD или принимаются RXD 10 бит информации: старт-бит (0), 8 бит данных и стоп-бит. Скорость приёма и передачи величина переменная и задаётся таймером.

Режим 2

В этом режиме через TXD передаются или из RXD принимаются 11 бит информации: старт-бит (0), 8 бит данных, программируемый 9-й бит и стоп-бит. При передаче 9-й бит данных может принимать значение 0 или 1. Частота приёма/передачи выбирается программой и может быть равна 1/32, либо 1/64 частоты резонатора в зависимости от управляющего бита SMOD.

Режим 3

Режим 3 совпадает с режимом 2 во всех деталях, за исключением частоты приёма/передачи, которая является величиной переменной и задаётся таймером.

Режимы 1, 2, и 3 называются асинхронными и последовательный порт микроконтроллера в этих режимах представляет собой универсальный асинхронный приемопередатчик (УАПП).

Скорость приёма/передачи

Скорость приёма/передачи, т.е. частота работы последовательного порта в различных режимах, определяется различными способами.

В режиме 0 частота передачи зависит только от частоты кварцевого резонатора $f_0 = f_{рез}/12$. За один машинный цикл последовательный порт передаёт один бит информации.

В режимах 1, 2 и 3 скорость приёма/передачи зависит от значения управляющего бита SMOD в регистре PCON, отвечающего за энергопотребление контроллера.

В режиме 2 частота передачи определяется выражением $f_2 = \left(2^{SMOD}/64\right) f_{рез.}$. Иными словами, при SMOD=0 частота передачи равна $(1/64) f_{рез.}$, а при SMOD=1 равна $(1/32) f_{рез.}$.

В режимах 1 и 3 в формировании частоты передачи кроме управляющего бита SMOD принимает участие таймер 1. При этом частота передачи зависит от частоты переполнения (OVT1) и определяется следующим образом: $f_{1,3} = \left(2^{SMOD}/32\right) f_{OVT1}$.

Прерывание от Т/С1 в этом случае должно быть заблокировано. Сам Т/С1 может работать и как таймер, и как счётчик событий в любом из трёх режимов. Однако наиболее удобно использовать режим таймера с автоперезагрузкой (старшая тетрада TMOD=0010B). При этом частота передачи определяется выражением:

$$f_{1,3} = \frac{\left(2^{SMOD}/32\right)\cdot\left(f_{рез.}/12\right)}{\left(256-\left(TH1\right)\right)}.$$

Регистр управления мощностью (PCON)

Предназначен в первую очередь для задания специализированных режимов управления мощностью микроконтроллера (режимов пониженного энергопотребления и холостого хода).

PCON.7	PCON.6	PCON.5	PCON.4	PCON.3	PCON.2	PCON.1	PCON.0
SMOD	-	-	-	**GF1**	**GF0**	**PD**	**IDL**

Биты регистра PCON имеют следующее назначение:

SMOD – удвоенная скорость передачи если бит установлен в 1, то скорость передачи вдвое больше, чем при SMOD=0.

PCON.6, PCON.5 PCON.4 – не используются. Эти биты зарезервированы для дальнейшего развития микроконтроллера.

GF1, GF0 – флаги, определяемые пользователем.

PD – бит режима пониженной мощности (при установке в 1).

IDL – бит режима холостого хода (при установке в 1).

При одновременной записи 1 в PD и IDL, бит PD имеет преимущество. Сброс содержимого PCON выполняется путём загрузки в него кода XXX0000.

1.6 Система прерываний

Режим прерывания программы – это останов выполнения текущей программы для реакции на какое-то заранее известное событие. При наступлении этого события начинает выполняться специальная подпрограмма обработки прерывания, после чего опять продолжает выполняться прерванная программа. В зависимости от того, чем вызывается прерывание, прерывания могут быть внешними или внутренними.

Внешние прерывания вызываются сигналами на входах **INT0** и **INT1** и могут быть инициированы либо уровнем, либо переходом сигнала из 1 в 0 (срезом) на этих входах в зависимости от значений управляющих бит **IT0** и **IT1** в регистре **TCON**. От внешних прерываний устанавливаются флаги **IE0** и **IE1** в регистре **TCON**, которые инициируют

вызов соответствующей подпрограммы обслуживания прерывания. Сброс этих флагов выполняется аппаратно только в том случае, если прерывание было вызвано по срезу сигнала. Если же прерывание вызвано уровнем входного сигнала, то сбросом флага **IE** управляет соответствующая подпрограмма обслуживания прерывания путём воздействия на источник прерывания с целью снятия запроса.

Флаги запросов прерывания от таймеров **TF0** и **TF1** сбрасываются автоматически при передаче управления подпрограмме обслуживания. Флаги запросов прерывания **RI** и **TI** устанавливаются блоком управления последовательного порта аппаратно, но сбрасываться должны программой.

Прерывания могут быть вызваны или отменены программой если они были установлены или сброшены как программно, так и аппаратно.

В блоке регистров специальных функций есть два регистра, предназначенных для управления режимом прерываний и уровнями приоритета (**IE** и **IP** соответственно).

Регистр масок прерывания (IE)

IE.7	IE.6	IE.5	IE.4	IE.3	IE.2	IE.1	IE.0
EA	**-**	**-**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**

EA – снятие блокировки прерываний. Сбрасывается программно для запрета всех прерываний независимо от состояний IE4-IE0.

IE.6, IE.5 – не используются.

ES – бит разрешения прерывания от последовательного порта. Установка/сброс программой для разрешения/запрета прерываний от флагов TI или RI.

ET1 – бит разрешения прерывания от таймера 1. Установка/сброс программой для разрешения/запрета прерываний от таймера 1.

EX1 – бит разрешения внешнего прерывания 1. Установка/сброс программой для разрешения/запрета прерываний.

ET0 – бит разрешения прерывания от таймера 0. Работает аналогично ET1.

EX0 – бит разрешения внешнего прерывания 0. Работает аналогично EX1.

Регистр приоритетов прерывания (IP)

IP.7	IP.6	IP.5	IP.4	IP.3	IP.2	IP.1	IP.0
-	**-**	**-**	**PS**	**PT1**	**PX1**	**PT0**	**PX0**

IP7, IP6, IP5 – не используются.

PS – бит приоритета УАПП. Установка/сброс программой для присваивания

прерыванию от УАПП высшего/низшего приоритета.

PT1 – бит приоритета таймера 1. Установка/сброс программой для присваивания прерыванию от таймера 1 высшего/низшего приоритета.

PX1 – бит приоритета внешнего прерывания 1.Установка/сброс осуществляются программно. Для присваивания высшего/низшего приоритета внешнему прерыванию INT1.

PT0 – бит приоритета таймера 0. Работает аналогично PT1.

PX0 – бит приоритета внешнего прерывания 0. Работает аналогично PX1.

Обслуживание прерываний.

При возникновении запроса на прерывание происходит запись в стек содержимого счетчика команд и переход по адресу вектора прерывания, если не:

- обслуживается прерывание равного или высшего приоритета;
- текущий машинный цикл - последний цикл выполняемой команды;
- выполняется RETI - возвращение из обработки прерывания или команда, связанная с обращением к регистрам IE и IP.

На рис. 6 показана процедура обработки прерывания. При возникновении прерывания, если не выполняется одно из вышеперечисленных условий, то прекращается выполнение программы и происходит передача управления по адресу вектора этого прерывания. Там должна находиться команда безусловного перехода на точку входа подпрограммы обработки прерывания. Подпрограмма обработки прерывания должна заканчиваться командой **RETI** выхода из подпрограммы обработки прерывания. После ее выполнения происходит возврат к исполнению прерванной программы. В подпрограмме обработки прерывания должно быть предусмотрено сохранение содержимого регистров, если это необходимо. Для сохранения регистров можно также воспользоваться переключением банков регистров общего назначения (РОН).

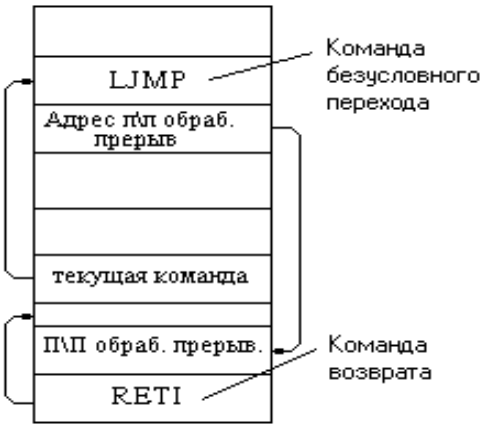

Рис. 6 Схема обработки прерывания

1.7 Типовая структура системы.

Типовая структура микропроцессорной системы, построенной на основе однокристального микроконтроллера 8051, представлена на рис. 7. Это система, содержащая внешнее постоянное запоминающее устройство программ (**ROM**) и оперативное запоминающее устройство для хранения данных (**RAM**). Для подключения к устройству управления верхнего уровня используется последовательный порт, к которому добавлены преобразователи сигналов для интерфейса “токовая петля”, собранные на основе преобразователя токового сигнала в логический с оптронной развязкой К293ЛП1 (приемник) и схемы И-НЕ с открытым коллектором К155ЛА13 (передатчик). При этом логические сигналы передаются уровнем тока в цепи, точнее его наличием или отсутствием. Стандартное значение тока для интерфейса “токовая петля” составляет 20 мА. В соответствии с этим выбираются номиналы резисторов входной и выходной цепи. В качестве альтернативного варианта вместо интерфейса “токовая петля” может быть использован стандартный интерфейс с передачей уровней по напряжению. В этом случае, вместо оптронной схемы используются стандартные преобразователи уровней, например, К170УП1, К170АП1. Параллельный порт Р1 служит для подключения других внешних устройств или в качестве входов/выходов. Для обеспечения двунаправленной передачи данных через этот порт может быть использован двунаправленный буфер, например, К555АП6. При однонаправленной передаче данных можно использовать однонаправленный буфер, например, К555АП5. Кроме того, при необходимости часть разрядов порта **P1** можно использовать для ввода данных, а оставшуюся часть – для вывода. В этом случае при программировании для доступа к порту необходимо использовать команды с побитной адресацией. Для запоминания младших разрядов адреса в циклах обращения к внешней памяти использован регистр **RGA**. Адрес должен запоминаться в нем по заднему фронту сигнала **ALE**. В качестве этого регистра может быть использован любой 8-ми разрядный статический регистр. Дешифратор адреса **DCA** служит для размещения в адресном пространстве микроконтроллера внешних запоминающих устройств, т.е. определения, по каким адресам происходит обращение к этим устройствам. Он может быть построен на основе логических интегральных схем или стандартных интегральных схем дешифраторов. Выработка сигнала **RESET** при включении питания производится за счет подключения RC-цепочки к соответствующему входу микроконтроллера.

При использовании внутренних (резидентных) памяти данных и программ отпадает необходимость в устройствах, обозначенных **RAM, ROM, RGA, DCA** (рис.7).

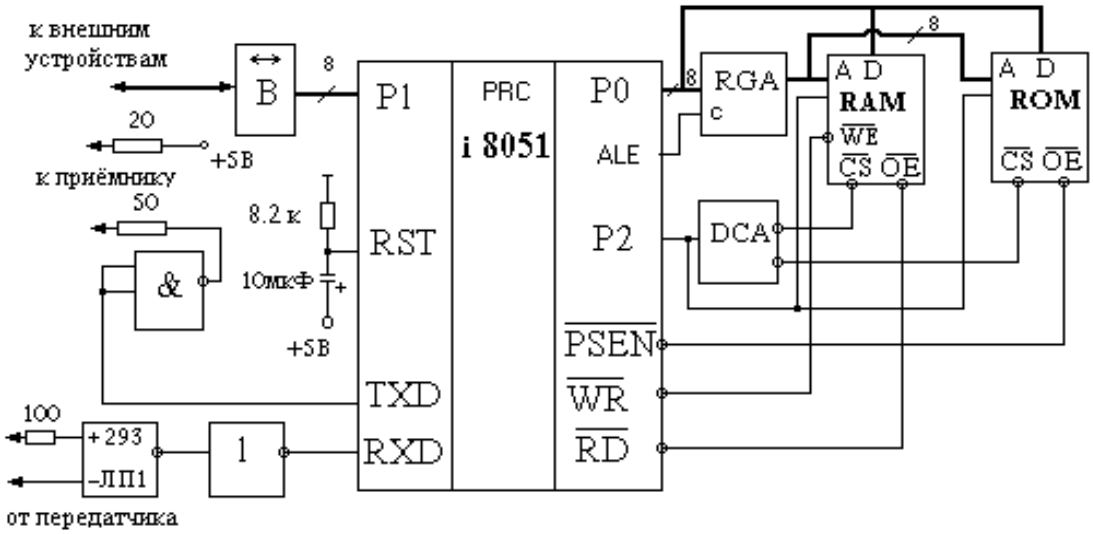

Рис. 7 Типовая структура системы на основе однокристального микроконтроллера

1.8 Система команд

Система команд содержит 111 базовых команд, которые разделяются по функциональному признаку на пять групп: команды передачи данных, арифметических операций, логических операций, передачи управления и операций с битами. Большинство команд (94) имеют формат один или два байта и выполняются за один или два машинных цикла. При анализе команд принято разделять их на две части – код операции (КОП) и операнды. Код операции – это байт, соответствующей команде, а операнд – это тот объект (бит, байт и т.д.), над которым совершается заданное кодом операции действие. Список команд микроконтроллера 8051 приведен в приложении 1.

2. Лабораторная работа «Проектирование устройства управления шаговым двигателем на основе микроконтроллера 8051»

2.1 Постановка задачи проектирования

Лабораторная работа рассчитана на 8 академических часов (2 лабораторных занятия).

Целью лабораторной работы является создание и отладка программы управления шаговым двигателем на языке ассемблера для устройства управления на основе микроконтроллера 8051, конфигурация которого задается данными варианта.

Задание на проектирование. Необходимо спроектировать микропроцессорное устройство на основе однокристального микроконтроллера, которое управляет шаговым

двигателем (ШД) с четырьмя обмотками по командам от устройства управления верхнего уровня, которое подключается к микроконтроллеру по последовательному интерфейсу.

Один из возможных вариантов схемы устройства приведен на рис. 8. Характеристики проектируемого устройства в соответствии с вариантом приведены в таблице 5.

Варианты задания различаются между собой режимами работы микроконтроллера. Во всех вариантах используется обмен с устройством верхнего уровня по последовательному каналу. Режим обмена – асинхронный 10 (десять) или 11 (одиннадцать) бит. В каждом варианте указана частота кварцевого резонатора для генератора тактовой частоты микроконтроллера.

Устройство верхнего уровня задает микроконтроллеру координату (количество шагов) и направление движения ШД в виде одного байта, где старший бит определяет направление движения: 0 – влево, 1 – вправо. Остальные семь бит определяют количество шагов. Максимальное количество шагов, которое может сделать ШД за одну команду – 128. Если требуется большее количество шагов, то посылается следующая команда микроконтроллеру. В зависимости от варианта используются две схемы включения ШД – с делением и без деления шага. Количество шагов в единицу времени фиксировано и приведено в табл. 5.

Преимущества привода на основе ШД – это отсутствие датчика обратной связи. Управление шаговым двигателем осуществляется от начального положения привода, в которое его надо переместить после включения питания устройства. Для этого используется датчик начального положения механизма – концевой выключатель. Он должен быть подключен к входу микроконтроллера, например, INT0. При включении питания двигатель осуществляет движение влево, пока не сработает концевой выключатель. Это означает, что механизм находится в начальном положении и микроконтроллер готов к приему команд управления.

Чтобы заставить двигатель вращаться, необходима последовательная коммутация обмоток в соответствии с табл. 3 – без деления шага и в соответствии с табл. 4 – с делением шага. Единица в таблице означает, что через данную обмотку протекает ток, ноль – тока в обмотке нет. Направление вращения двигателя определяется последовательностью коммутации обмоток согласно таблицам, например, сверху вниз – правое вращение, снизу вверх – левое. При составлении алгоритма и программы, рекомендуется использовать отдельные подпрограммы для движения двигателя влево и вправо на заданное число шагов.

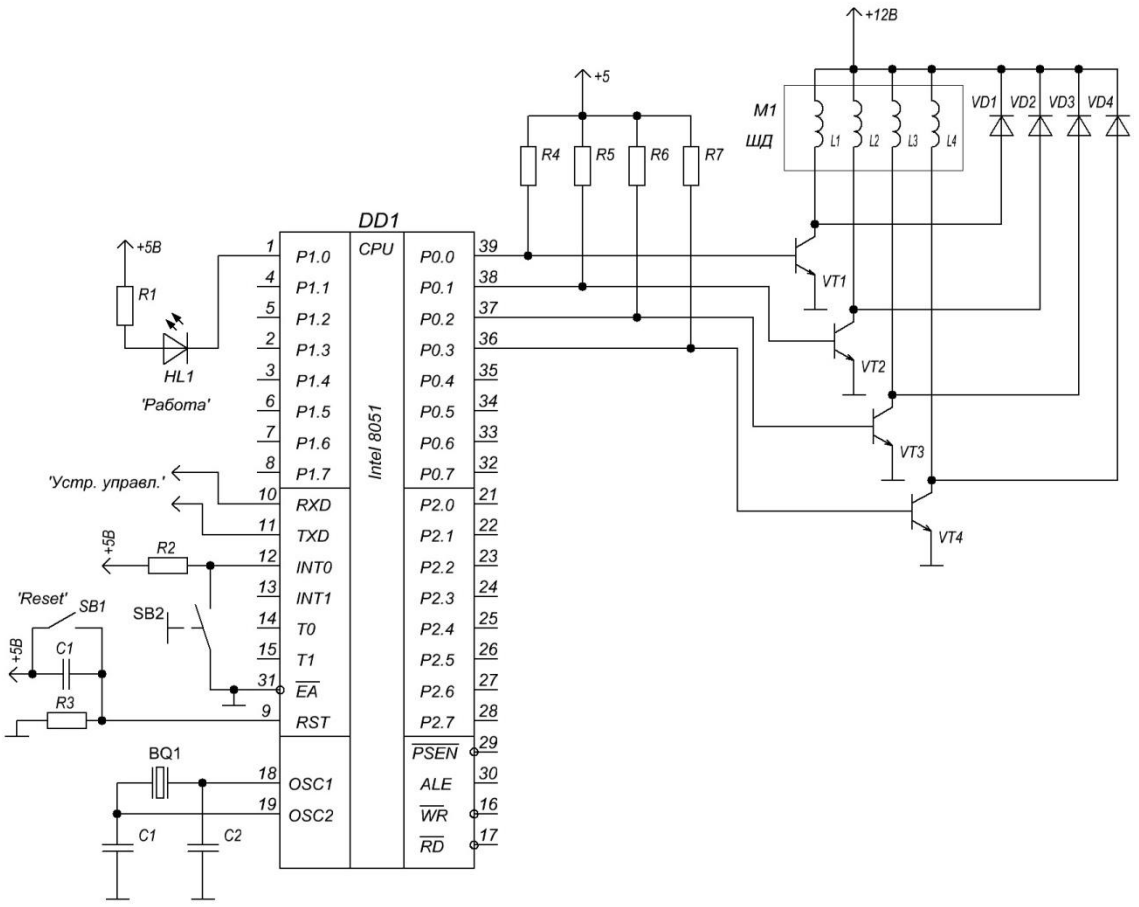

Рис. 8 Схема электрическая принципиальная устройства управления шаговым двигателем

Электрическая принципиальная схема устройства приведена на рис. 8. В качестве микроконтроллера здесь используется классический микроконтроллер 8051 в корпусе с 40 выводами. Его можно заменить современным аналогом, например, AT89C51 или AT89S51, выпускаемым фирмой ATMEL. Для упрощения схемы используется внутренняя память программ. Коммутация обмоток шагового двигателя М1 осуществляется биполярными транзисторами VT1-VT4, работающими в ключевом режиме. Они включены по схеме с общим эмиттером. Ток, проходящий через обмотки L1-L4 шагового двигателя М1, не меняет своего направления, поэтому такой двигатель называется униполярным. Порт P0 микроконтроллера имеет выходы с открытым коллектором, поэтому ток открытия транзисторов VT1-VT4 формируется резисторами R4-R7. Для закрытия транзисторов VT1-VT4 открываются выходные транзисторы порта P0 микроконтроллера. В результате, для создания протекающего тока через обмотку шагового двигателя в соответствующий разряд порта микроконтроллера записывается единица и наоборот, для обесточивания обмотки –

записывается ноль. Вместо биполярных коммутирующих транзисторов могут применяться МОП транзисторы, включенные по схеме с общим истоком. Диоды VD1-VD4 защищают транзисторы VT1-VT4 и микроконтроллер от ЭДС самоиндукции, которая возникает при выключении транзисторных ключей.

Задание частоты синхронизации микроконтроллера происходит от внутреннего генератора, частота которого задается внешним кварцевым резонатором BQ1. Для надежного запуска генератора используются конденсаторы C1 и C2 небольшой емкости (10-68 *пФ*). Для приведения всех внутренних регистров в начальное состояние при включении питания используется цепочка C1, R3. Для принудительного сброса микроконтроллера используется кнопка SB1. Для индикации работы двигателя применен светодиод HL1. Резистор R1 ограничивает ток через светодиод. Ток проходит через светодиод при записи в порт P1.0 логического нуля. Устройство управления шаговым двигателем связано с устройством верхнего уровня по последовательному порту представленному линиями RxD и TxD. По линии RxD производится прием информации, по линии TxD происходит передача данных. Фактически, для упрощения задания, задействована только линия RxD, которая принимает управляющий байт информации. Для задания начального положения механизма, приводимого шаговым двигателем, используется концевой выключатель SB2, срабатывающий в крайнем положении, например, крайнем левом положении, которое достигается при движении механизма влево. Крайнее левое положение индицируется появлением логического нуля на входе INT0 микроконтроллера.

В зависимости от варианта, возможна коммутация обмоток с делением и без деления шага. Алгоритм управления без деления шага более прост, но при этом движение вала шагового двигателя будет менее плавным, по сравнению с алгоритмом где присутствует деления шага.

Порядок коммутации обмоток без деления и с делением шага приведен в таблицах 3 и 4 соответственно. При использовании алгоритма без деления шага последовательно коммутируются четыре обмотки, т.е. в любой момент времени протекает ток через одну обмотку двигателя.

При делении шага добавляются четыре дополнительных состояния, когда ток протекает одновременно в предыдущей и последующей обмотках. При этом переход от одного устойчивого состояния (шага) в другое происходит более плавно.

Таблица 3 Порядок коммутации обмоток шагового двигателя без деления шага

L1	L2	L3	L4
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Таблица 4 Порядок коммутации обмоток шагового двигателя с делением шага

L1	L2	L3	L4
1	0	0	0
1	1	0	0
0	1	0	0
0	1	1	0
0	0	1	0
0	0	1	1
0	0	0	1
1	0	0	1

2.2 Порядок выполнения работы

1. Получение задания в соответствии с номером варианта.
2. Изучение устройства микроконтроллера 8051 и его системы команд.
3. Изучение принципиальной схемы устройства управления шаговым двигателем и ее коррекция (при необходимости) в соответствии со своим вариантом.
4. Разработка алгоритма работы программы.
5. Выполнение необходимых расчетов согласно данным варианта.
6. Ознакомление с отладчиком ассемблерных программ Pinnacle 52 или аналогичным. Выполнение пробных примеров, приведенных в разделе 2.3.
7. Составление и отладка программы на языке ассемблера.
8. Демонстрация преподавателю работы отлаженной программы в пошаговом режиме.
9. Составление отчета по лабораторной работе, включающего: данные варианта задания, принципиальную схему устройства и описание ее работы, расчеты значений управляющих регистров микроконтроллера, блок-схему алгоритма работы устройства, программный код на языке ассемблера с комментариями, подробную блок-схему алгоритма с указанием рабочих регистров микроконтроллера.

Таблица 5 Варианты заданий для выполнения лабораторной работы

Номер вариан-та	Частота кварцевого резонатора, МГц	Деление шага	Число шагов в секунду	Частота последовательного порта, бит/с	Число бит, передаваемых последователь ным портом в асинхронном режиме
1	2	-	5	1200	10
2	**3**	**+**	**7**	**2400**	**11**
3	4	-	8	4800	10
4	5	+	9	9600	11
5	6	-	10	1200	10
6	8	+	11	2400	11
7	10	-	12	4800	10
8	11	+	13	9600	11
9	12	-	14	1200	10
10	2	+	15	2400	11
11	4	-	3	4800	10
12	5	+	4	9600	11
13	6	-	5	1200	10
14	8	+	6	2400	11
15	10	-	7	4800	10
16	11	+	8	9600	11
17	12	-	9	1200	10
18	2	+	10	2400	11
19	4	-	11	4800	10
20	5	+	12	9600	11
21	6	-	13	1200	10
22	8	+	14	2400	11
23	10	-	15	4800	10
24	11	+	14	9600	11
25	12	-	13	1200	10
26	2	+	12	2400	11
27	4	-	11	4800	10
28	5	+	10	9600	11
29	6	-	9	1200	10
30	8	+	8	2400	11
31	10	-	7	4800	10
32	11	+	6	9600	11
33	12	-	5	1200	10
34	2	+	4	2400	11
35	4	-	3	4800	10

2.3 Порядок работы с отладчиком ассемблерных программ Pinnacle 52

Для разработки и отладки программ для микроконтроллеров семейства 8051 существует множество программ для персональных компьютеров. Примером может служить программа Fd51, широко распространенная в начале 90-х годов. К ее недостаткам можно отнести интерфейс операционной среды DOS, требующий набора отдельных команд в командной строке для трансляции ассемблерной программы. Для удобства пользователя удобно применять программы отладчиков, работающие под операционной средой Windows. Одной из таких программ является Pinnacle 52.

Для запуска программы *Pinnacle 52 Professional Development System* щелкните на рабочем столе соответствующую иконку. В открывшемся окне нажмите клавишу с надписью «Register Later». После загрузки основного окна программы войдите в меню “Project” – “Project Options”. В появившемся окне выберите тип микроконтроллера Standart 8051. С помощью меню “View” откройте следующие окна: “Registers”, “Ports”, ‘Timers”, “Internal RAM”.

Выбрав меню “File” – “New” откройте окно в котором предполагается ввод текста программы для отладки. Если имеется текстовый файл с программой, то воспользуйтесь меню “File” – “Open”. Файл должен иметь расширение *asm*. Редактируемый файл можно сохранить, выбрав меню “File” – “Save”. Для трансляции в машинный код отредактированного файла воспользуйтесь меню “Project” – “Compile&Link”, либо нажав сочетание клавиш “Ctrl+F2”. Справочная информация по регистрам микроконтроллера находится в меню “Help” – “8051 Microcontroller”. В меню “Help” – “Pinnacle Help” находится справочная информация как по языку программирования, так и по работе с программой.

Основным окном, необходимым для отладки программы является окно “Registers”. Здесь отображаются данные хранящиеся в регистрах на текущий момент выполнения программы. Щелкнув два раза мышкой по выбранному регистру, открывается окно с обозначениями отдельных бит этого регистра. Пользователь может в процессе отладки программы менять значение каждого бита. Флаги регистра состояния программы PSW вынесены в отдельную область именуемую “Flags” для удобства отслеживания изменений в процессе отладки программы. В окне “Registers” также отображаются регистры, отвечающие за прерывания IE, IP, регистр управления мощностью потребления PCON, регистр управления и статуса последовательного порта SCON.

Окно “Ports” содержит побитное отображение состояния портов ввода-вывода. Данное окно разделено на две части: в левой части отображаются выходные данные, записанные в регистры-защелки, а в правой части отображаются сигналы, которые поступают на линии

микроконтроллера. Пользователь может менять состояние выходных защелок и входных сигналов щелкнув мышью один раз на соответствующий бит.

Окно “Timers” отображает содержимое регистров управления и статуса таймер-счетчиков TCON и регистра режимов таймер-счетчиков TMOD. Щелкнув мышью по этим регистрам, можно побитно редактировать в отдельном окне их состояние. Содержимое регистров счета TH0 и TL0, а также TH1 и TL1 отображается одновременно в шестнадцатеричном формате. Обозначенные в окне регистры T2CON, T2MOD, Timer2, RCAP2 предназначены для других модификаций микроконтроллеров и для рассматриваемой задачи не используются. Открыв окно с описанием кода регистра можно узнать адрес этого регистра в памяти данных микроконтроллера.

Специальное окно с названием DPTR используется для отображения содержимого шестнадцатиразрядного регистра указателя внешних данных DPTR. Для обозначения состояния этого регистра выделено специальное окно из-за того, что не в каждой отлаживаемой программе используется внешняя память данных.

Особого внимания заслуживает окно Internal RAM, которое отображает внутреннюю память данных микроконтроллера (128 байт для 8051 и 256 байт для 8052). Зеленая область памяти микроконтроллера отображает выбранный в данный момент банк регистров общего назначения (РОН). Задав нужные значения флажков RS0 и RS1 можно выбрать нужный банк РОН. Четыре банка памяти РОН занимают область 00-1F. Регистр, помеченный красным цветом, обозначает элемент, который будет следующим помещен в стек. Адрес этого элемента является суммой содержимого регистра указателя стека плюс единица ((SP)+1).

Область, помеченная желтым цветом с адресами 20-2F представляет область памяти использующую побитовую адресацию. Для отображения побитного содержимого данной области может быть использовано отдельное окно “Bit Fields”, которое можно открыть, выбрав меню “View” – “Bit Fields”. Черным цветом выбрана память, которая недоступна для выбранного микроконтроллера.

После загрузки и трансляции программы можно закрыть окно с исходным текстом программы и открыть окно Code Window, выбрав соответственно меню: “View” – “Code Memory”. Это окно показывает дизасемблированый текст программы, адрес выполняемой команды и код команды. Зеленой полосой отображается команда, которая будет выполняться следующей. Нажатием клавиши Goto в данном окне осуществляется переход к нужной области памяти программ, введя соответствующий адрес. Клавишей Set PC можно задать адрес той команды, которую необходимо выполнить. Клавишей Toggle устанавливаются точки останова в текущей программе во время ее отладки. Отладчик ассемблерных программ Pinnacle 52 имеет еще ряд дополнительных возможностей для отладки программ, с которыми можно ознакомиться в разделе помощи “Help”- “Pinnacle Help”.

В качестве упражнения наберите текст следующей программы:

```
org 0
        ljmp new
new:  mov a,#150
        mov b,#30
        div ab
       end
```

Эта программа записывает число 150 в аккумулятор, а в расширитель аккумулятора – число 30. (числа десятичные). В пятой строке эти числа делятся и результат записывается в аккумулятор. Откомпилируйте программу, откройте окно дизассемблера и пошагово выполните ее. При этом наблюдайте изменения в регистрах a и b.

В качестве другого примера наберите текст более сложной программы:

```
      org 0
      ljmp m1
m1: mov r0,#01100010b
      mov r1, #01000011b
      mov a, r1
      cpl a
      add a, r0
      jnc m2
      inc a
m2: mov r2, a
      end
```

Эта программа производит вычитание двух чисел, используя команду сложения, в обратном коде. Первое число больше нуля, второе – меньше нуля, результат – больше нуля. В третьей и четвертой строках вводятся числа в бинарном виде в регистры r0 и r1. В пятой и шестой строках происходит инвертирование разрядов второго числа. В седьмой строке происходит сложение чисел. В восьмой строке проверяется наличие переноса. Если есть перенос, то в девятой строке увеличивается содержимое аккумулятора на единицу. В десятой строке результат записывается в регистр r2.

В этих двух примерах вначале присутствует служебное слово org. Оно предписывает компилятору следующую команду записать по адресу, указанному после org. В данном случае эта команда ljmp перехода на начало программы.

2.4 Пример выполнения задания

2.4.1 Исходные данные и необходимые расчеты

Приведем пример выполнения задания на примере варианта 2.

Исходные данные:

Частота кварцевого резонатора – 3 МГц;

Деление шага – присутствует;

Число шагов в секунду – 7;

Частота последовательного порта – 2400 бит/с;

Число бит передаваемых последовательным портом – 11.

В составе микроконтроллера имеется два таймер-счетчика Т/С0 и Т/С1. Назначим Т/С0 отсчитывать временные интервалы между шагами для задания скорости вращения вала двигателя, а Т/С1 для задания скорости передачи последовательного порта. При этом Т/С0 будет работать в режиме 1 (шестнадцатиразрядный счетный регистр), а Т/С1 – в режиме 2 (восьмиразрядный регистр с автозагрузкой).

Поскольку таймер-счетчики считают машинные циклы, то определим длительность машинного цикла исходя из заданной частоты кварцевого генератора:

$$T_{мц} = \frac{12}{f_Г} = \frac{12}{3 \cdot 10^6 \ Гц} = 4 \cdot 10^{-6} c = 4 \ мкс .$$

Поскольку в задании присутствует деление шага, то управление происходит по полушагам. Определим длительность полушага:

$$t_{п.ш.} = \frac{1}{2 \cdot 7} = 0{,}0714 c = 71{,}4 \ мс .$$

Количество счетов таймер-счетчика для задания полушага:

$$N_{сч.} = \frac{t_{п.ш.}}{T_{мц}} = \frac{0{,}0714 c}{4 \cdot 10^{-6} c} = 17850.$$

Поскольку таймер счетчик Т/С0 считает от начального значения N_0 до максимального значения 65535 (FFFFH), то определим N_0:

$$N_0 = 65535 - N_{сч} = 65535 - 17850 = 47685.$$

Это значение надо каждый раз записывать в шестнадцатиразрядный счетный регистр таймер-счетчика Т/С0, который представлен двумя восьмиразрядными регистрами TH0 и TL0.

Представим значение N_0 в шестнадцатеричной форме: $N_0 = $BA45H. Из этого следует, что: TH0=BAH, TL0=45H или в двоичной форме TH0=10111010B, TL0=01000101B.

Эти значения надо каждый раз записывать в счетный регистр перед запуском таймера на счет для задания времени задержки очередного полушага.

Может получиться ситуация, когда частота кварцевого резонатора велика, а время шага (полушага) велико, что N_0 примет отрицательное значение. В этом случае необходимо несколько раз в программе запускать таймер-счетчик, при этом уменьшить расчетное время таймера в такое же количество раз.

Рассчитаем значение регистра TH1, которое обеспечивает работу T/C1 в режиме 2 (восьмиразрядный регистр с автозагрузкой). T/C1 обеспечивает скорость работы последовательного порта в режиме 3.

Согласно формулы расчета скорости последовательного порта в режимах 1 и 3:

$$f_{1,3} = \frac{\left(2^{SMOD}/32\right)\cdot\left(f_{рез.}/12\right)}{\left(256-\left(TH1\right)\right)}\ .$$

Согласно варианту $f_{1,3} = 2400.$

Бит удвоения скорости последовательного порта SMOD (регистр PCON) для определенности примем равным 0.

Тогда получим уравнение:

$$2400 = \frac{1/32\ .\ 3\cdot10^6/12}{\left(256-\left(TH1\right)\right)}.$$

Откуда находим значение TH1:

$$TH1 = 256 - \frac{3\cdot10^6}{32\cdot12\cdot2400} = 252{,}74 \approx 253.$$

2.4.2 Блок-схема алгоритма управления шаговым двигателем

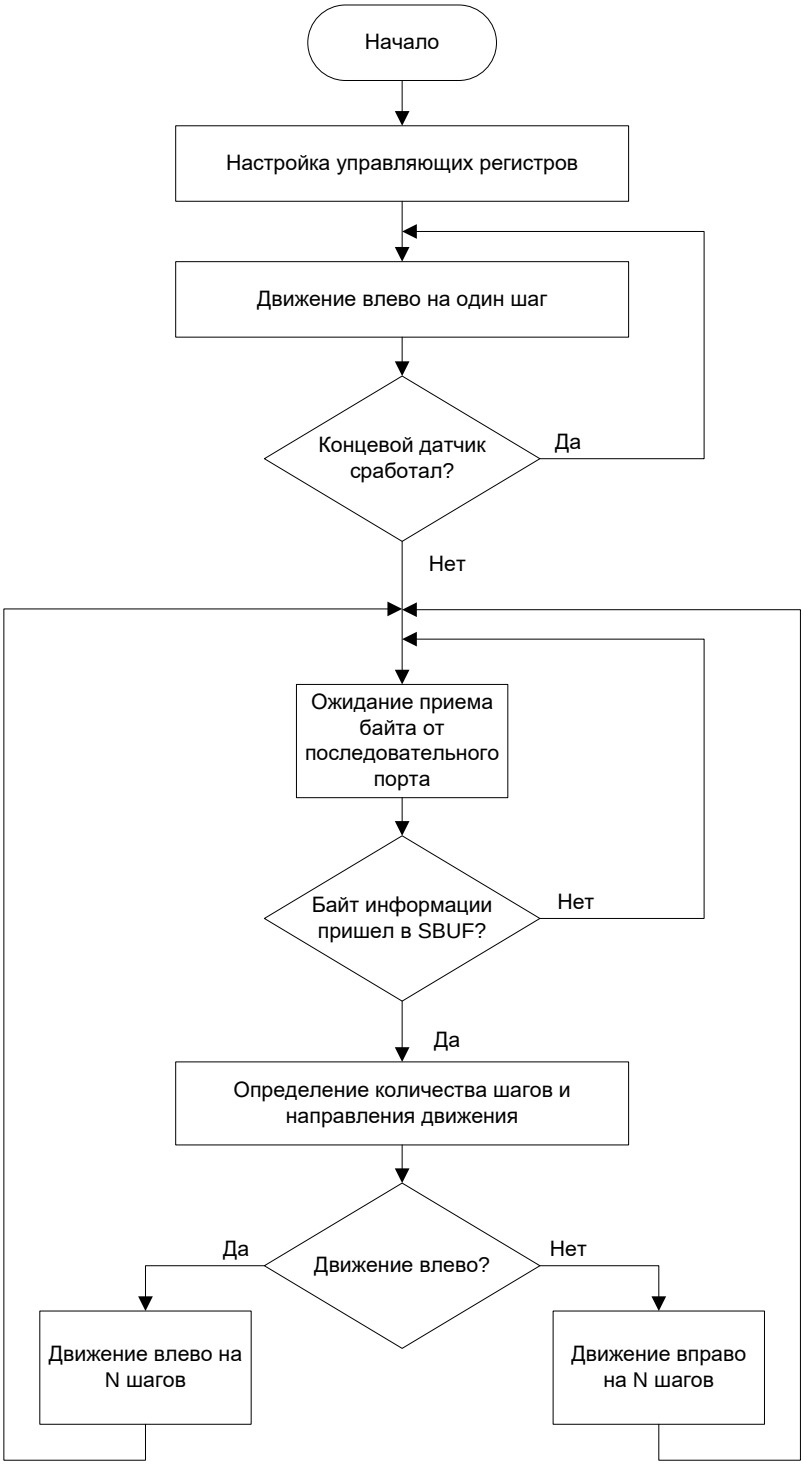

Рис. 10 Блок-схема алгоритма управления шаговым двигателем

Управление шаговым двигателем происходит согласно блок-схеме, приведенной на рис. 10. Вначале производится настройка управляющих регистров, которые задают режимы работы таймер-счетчиков (TMOD), последовательного порта (SCON), режимы прерываний

(IE). Затем производится калибровка устройства путем установки механизма в начальное состояние. Для этого производится движение механизма влево на один шаг и проверяется срабатывание концевого выключателя. Если концевой выключатель не сработал, то цикл повторяется и двигатель движется влево на один шаг до тех пор, пока не сработает концевой выключатель. Срабатывание концевого выключателя означает нахождение механизма в начальном положении. Далее происходит зацикливание программы и ожидание приема байта от последовательного порта. Байт последовательного порта имеет следующую организацию: старший бит отвечает за направление движения, например, ноль – влево, единица – вправо, а остальные семь бит представляют двоичный код количества шагов, которое должен отработать шаговый двигатель. В результате шаговый двигатель за один раз может повернуться максимум на 127 шагов. Далее происходит определение направления движения и количества шагов, которое должен сделать шаговый двигатель. Затем осуществляется само движение влево или вправо в зависимости от заданного направления на N шагов. Затем программа переходит в режим ожидания приема очередного байта от последовательного порта. В результате программа зацикливается. Блоки движение влево на N шагов и движение вправо на N шагов целесообразно выполнить в виде отдельных подпрограмм и передавать в них в качестве параметра количество шагов N.

2.4.3 Подпрограммы движения влево и вправо

Для обеспечения движения шагового двигателя необходимо осуществлять коммутацию его обмоток согласно таблицам 3, 4. В нашем случае коммутация осуществляется согласно таблице 4, т.е. с делением шага. Переход от одного состояния (одной строки таблицы) к другому происходит при одном полушаге. Т.е. поворот на один шаг означает смену двух состояний, указанных в таблице. При этом, смена состояний в таблице сверху вниз означает одно направление движения, для определенности назовем его левое, а изменение состояний снизу вверх – другое направление движения – правое.

Эти состояния необходимо записать в память данных, чтобы в дальнейшем их оттуда извлекать. Это потребуется сделать вначале программы с использованием следующих команд:

```
mov 30h, #00001000b
mov 31h, #00001100b
mov 32h, #00000100b
mov 33h, #00000110b
mov 34h, #00000010b
mov 35h, #00000011b
mov 36h, #00000001b
mov 37h, #00001001b
```

В качестве области памяти данных выбрана область начиная с адреса 30h, поскольку с этого адреса начинается область байтовой адресации.

Приведем непосредственно текст подпрограммы движения влево:

```
left:   mov p1, @r0
        dec r0
        mov th0, #10111010b
        mov tl0, #01000101b
        setb tr0
m1:     jnb tf0, m1
        clr tf0
        cjne r0, #2fh, m2
        mov r0, 37h
m2:     djnz r1, left
        ret
```

В этой подпрограмме регистр r1 содержит количество полушагов, которое должен сделать шаговый двигатель. После выполнения подпрограммы r1 содержит ноль. Т.е. подпрограмма строится на основе цикла, организованного командой djnz. В первой строке подпрограммы в порт p1 выводится строка состояния обмоток шагового двигателя из памяти данных, на которую указывает указатель r0. Фактически r0 содержит адрес ячейки памяти данных, в которых содержатся состояния обмоток шагового двигателя. Таким образом, для нашего варианта r0 может принимать значения от 30h до 37h.

Строка 2 осуществляет уменьшение содержимого указателя r0 на единицу.

В строках 3 и 4 прописываются значения счетного регистра T/C0, которые были получены ранее. В строке 5 происходит запуск T/C0. Строка 6 содержит пустой цикл, до тех пор, пока T/C0 не завершит свою работу. Окончанием цикла служит установка флага tf0 в единицу. В седьмой строке флаг tf0 сбрасывается. В строке 8 проверяется, не вышел ли указатель r0 за нижний предел 30h, если вышел, то ему присваивается максимальное значение 37h, которое в дальнейшем будет уменьшаться.

Подпрограмма движения вправо строится аналогично:

```
right:  mov p1, @r0
        inc r0
        mov th0, #10111010b
        mov tl0, #01000101b
        setb tr0
m3:     jnb tf0, m3
```

```
        clr tf0
        cjne r0, #38h, m4
        mov r0, 30h
m4:     djnz r1, left
        ret
```

Отличия от подпрограммы движения влево проявляются лишь в том, что во второй строке происходит увеличение указателя на единицу, и проверка указателя происходит по максимальному значению 37h, если значение стало 38h, то ему присваивается начальное значение 30h.

2.4.4 Настройки микроконтроллера

Приведенные выше подпрограммы движения влево и вправо помещаются в конце программы. Вначале программы необходимо реализовать последовательность действий согласно алгоритму на рис. 10.

Сформируем управляющий код регистра настройки таймер-счетчиков TMOD.

TMOD.7	TMOD.6	TMOD.5	TMOD.4	TMOD.3	TMOD.2	TMOD.1	TMOD.0
GATE 1	**C/T1**	**M11**	**M10**	**GATE0**	**C/T0**	**M01**	**M00**

GATE0, GATE1=0 (не требуется управление блокировкой от входов INT0, INT1);
C/T0=0, C/T1=0 (работа устройств в качестве таймеров);
M01=0, M00=1 (режим 1 T/C0);
M11=1, M10=0 (режим 2 T/C1).
В результате требуется выполнить следующую команду:
mov tmod, #00100001b.

Сформируем управляющий код регистра управления таймер-счетчиков TCON.

TCON.7	TCON.6	TCON.5	TCON.4	TCON.3	TCON.2	TCON.1	TCON.0
TF1	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**

TF0=0, TF1=0 (биты завершения работы принимают начальные нулевые значения);
TR0=0, TR1=0 (не требуется запуск таймер-счетчиков);
IE0=0, IE1=0 (начальная установка флагов внешнего прерывания);
IT0=0, IT1=0 (установка внешних прерываний по уровню).

В результате требуется выполнить следующую команду:

mov tcon, #00000000b

Сформируем управляющий код регистра управления/статуса последовательного порта.

SCON.7	SCON.6	SCON.5	SCON.4	SCON.3	SCON.2	SCON.1	SCON.0
SM0	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**

SM0=1, SM1=1 (режим 3 работы последовательного порта);

SM2=0 (многопроцессорный режим отсутствует);

TB8=0, RB8=0 (8-ой передаваемый и принимаемый биты);

TI=0, RI=0 (флаги прерываний от передатчика и приемника).

В результате требуется выполнить следующую команду:

mov scon, #11000000b.

Сформируем управляющий код регистра управления мощностью.

PCON.7	PCON.6	PCON.5	PCON.4	PCON.3	PCON.2	PCON.1	PCON.0
SMOD	**-**	**-**	**-**	**GF1**	**GF0**	**PD**	**IDL**

SMOD=0 (согласно расчетам скорости последовательного порта);

GF0=0, GF1=0 (биты пользователя не используются);

PD=0, IDL=0 (энергосберегающие режимы не используются).

В результате требуется выполнить следующую команду:

mov pcon, #00000000b.

Обратите внимание, что из-за аппаратных особенностей микроконтроллера к регистру PCON нельзя обращаться побитно. Для изменения значения бита в программе требуется записать байт полностью.

Сформируем управляющий код регистра масок прерываний.

IE.7	IE.6	IE.5	IE.4	IE.3	IE.2	IE.1	IE.0
EA	**-**	**-**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**

EA=0, ES=0, ET1=0, EX1=0, ET0=0, EX0=0 (прерывания в программе не используются).

В результате необходимо выполнить следующую команду:
mov ie, #00000000b.

Сформируем управляющий код регистра приоритетов прерываний.

IP.7	IP.6	IP.5	IP.4	IP.3	IP.2	IP.1	IP.0
-	-	-	**PS**	**PT1**	**PX1**	**PT0**	**PX0**

PS=0, PT1=0, PX1=0, PT0=0, PX0=0 (приоритеты установлены низкие, прерывания в программе не используются).

В результате необходимо выполнить следующую команду:
mov ip, #00000000b

Выше были рассмотрены управляющие регистры, которые настраивают и управляют работой всех внутренних устройств микроконтроллера. Следует заметить, что если управляющий код регистров принимает значение ноль, то необязательно его прописывать в начальной установке программы, поскольку все внутренние регистры автоматически устанавливаются в ноль после сброса микроконтроллера после включения питания.

2.4.5 Вывод механизма в начальное положение

После настройки управляющих регистров следует выполнить вывод механизма в начальное положение. Для этого двигатель делает один шаг влево и проверяется наличие срабатывания концевого датчика SB2 (рис. 9). Цикл проверки длится до тех пор, пока датчик не сработает (рис. 10). Это можно реализовать с помощью следующих команд:

```
pos: mov r1, #02
     lcall left
     jb p3.2, pos
```

В первой строке в регистр r1 записывается количество полушагов – два полушага (один шаг). Во второй строке вызывается подпрограмма движения влево. В третьей строке если второй бит порта 3 установлен (на рис. 9 обозначен INT0), то переход на метку m3, иначе выполняется следующая команда.

Далее согласно блок-схеме алгоритма на рис.10 требуется принять управляющий бит, определить направление движения и количество шагов (полушагов) и вызвать подпрограмму движения влево или вправо. Соответствующий фрагмент программы будет иметь следующий вид:

```
        setb ren
        setb tr1
rec: jnb ri, m4
        clr ri
        mov a, sbuf
        anl a, #01111111b
        rl a
        mov r1, a
        mov a, sbuf
        anl a, #10000000b
        jz m5
        lcall right
        ljmp rec
m5: lcall left
        ljmp rec
```

В первой строке устанавливается единичное значение бита REN. Этим разрешается работа приемника последовательного порта. При нулевом значении этого бита прием не осуществляется. В следующей строке устанавливается бит TR1, который запускает работу таймер-счетчика 1, осуществляющего тактирование последовательного порта. В третьей строке происходит зацикливание микроконтроллера до тех пор, пока не будет установлен бит TR1 в единицу, означающий, что приемник последовательного порта завершил прием байта. После приема байта в четвертой строке бит RI сбрасывается. В строках 5-6 из принятого бита в SBUF выделяется информация о количестве шагов (младшие семь бит) и аккумулятор. В строках 7,8 количество шагов умножается на два путем сдвига содержимого аккумулятора влево, и результат помещается в регистр R1. Это необходимо, поскольку управление происходит с делением шага. Затем в строках 9, 10 в аккумулятор записывается старший бит принятого байта. Он отвечает за направления движения. В строке 10 происходит ветвление: если в аккумуляторе ноль – то переход по метке m5 на строку13 и вызов подпрограммы движения left, если в аккумуляторе не ноль – то вызов подпрограмм движения вправо right в строке 11. После завершения движений происходит безусловный переход по метке m4, где снова ожидается приход байта в последовательный порт (строки 13, 15).

2.4.6 Программа управления шаговым двигателем

Далее полностью составим программу управления шаговым двигателем и отладим ее в отладчике:

```
org 0000H
```

```
mov 30h, #1000b
mov 31h, #1100b
mov 32h, #0100b
mov 33h, #0110b
mov 34h, #0010b
mov 35h, #0011b
mov 36h, #0001b
mov 37h, #1001b
ljmp start
start: mov tmod, #00100001b
       mov tcon, #00000000b
       mov scon, #11000000b
       mov pcon, #00000000b
       mov ie,   #00000000b
       mov ip,   #00000000b
       mov r0,   #37h

; Отработка входящего сообщения по последовательному порту (UART)

pos:   mov r1, #02          ; число полушагов
       lcall left           ; вызов подпрограммы движения влево (lef)
       jb p3.2, pos         ; переход к pos, если зажат концевой выключатель
       setb ren             ; установка бита ren для UART
       setb tr1             ; установка бита tr1 для UART
rec:   jnb ri, rec
       clr ri
       mov a, sbuf
       anl a, #01111111b
       rl a
       mov r1, a
       mov a, sbuf
       anl a, #10000000b
       jz m5
       lcall right
       ljmp rec

m5:    lcall left
       ljmp rec
```

```
; Поворот налево

left:  setb p3.0
       mov p1, @r0 ; before change mov p1, @r0
       dec r0
       mov th0, #10111010b
       mov tl0, #01000101b
       setb tr0
m1:    jnb tf0, m1
       clr tf0
       cjne r0, #2fh, m2
       mov r0, 37h
m2:    djnz r1, left
       setb p3.0
       ret

; Поворот направо

right: setb p3.0
       mov p1, @r0
       inc r0
       mov th0, #10111010b
       mov tl0, #01000101b
       setb tr0
m3:    jnb tf0, m3
       clr tf0
       cjne r0, #38h, m4
       mov r0, 30h
m4:    djnz r1, right
       setb p3.0
       ret
       end
```

Обратите внимание на то, что программы на ассемблере всегда начинаются со служебного слова org и заканчиваются служебным словом end. Кроме того, в программном коде появились строки clr p1.0 и setb p1.0, которые включают и выключают индикацию движения двигателя в подпрограммах left и right.

Литература

1. Боборыкин А.В., Липовецкий Г.П. Однокристальные микроЭВМ. Справочник. - М: БИНОМ, 1994.
2. Васильев А.Е. Микроконтроллеры. Разработка встраиваемых приложений: учебное пособие. – БХВ-Петербург, 2008. – 304 с.
3. Новиков Ю.В., Скоробогатов П.К. Основы микропроцессорной техники: учебное пособие. – М.:БИНОМ, 2009. – 357 с.
4. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. - М.: Энергоатомиздат, 1990
5. Фрунзе А.В. Микроконтроллеры? Это же просто! Т.4. – М.:Додека, 2008. – 464 с.

Приложение 1 Команды микроконтроллера 8051

Команды арифметических операций

Название команды	Мнемоника	Операция
Сложение аккумулятора с регистром	ADD A, Rn	(A) ← (A)+(Rn)
Сложение аккумулятора с прямоадресуемым байтом	ADD A, ad	(A) ← (A)+(ad)
Сложение аккумулятора с байтом из внутреннего ОЗУ(i=0,1)	ADD A, @Ri	(A) ← (A)+((Ri))
Сложение аккумулятора с константой	ADD A, #data8	(A) ← (A)+#data8
Сложение аккумулятора с регистром и переносом	ADDC A, Rn	(A) ← (A)+(Rn)+(C)
Сложение аккумулятора с прямоадресуемым байтом и переносом	ADDC A, ad	(A) ← (A)+(ad)+(C)
Сложение аккумулятора с байтом из внутреннего ОЗУ и переносом	ADDC A, @Ri	(A) ← (A)+((Ri))+(C)
Сложение аккумулятора с константой и переносом	ADDC A, #data8	(A) ← (A)+#data8+(C)
Десятичная коррекция аккумулятора	DA A	
Вычитание из аккумулятора регистра и заема	SUBB A, Rn	(A) ← (A)-(Rn)-(C)
Вычитание из аккумулятора прямоадресуемого байта и заема	SUBB A, ad	(A) ← (A)-(ad)-(C)
Вычитание из аккумулятора байта	SUBB A, @Ri	(A) ← (A)-((Ri))-(C)

из внутреннего ОЗУ и заема		
Вычитание из аккумулятора константы и заема	SUBB A, #data8	(A) ← (A)-(ad)-(C)
Инкремент аккумулятора	INC A	(A) ← (A)+1
Инкремент регистра	INC Rn	(Rn) ← (Rn)+1
Инкремент прямоадресуемого байта	INC ad	(ad) ← (ad)+1
Инкремент байта из внутреннего ОЗУ	INC @Ri	((Ri)) ← ((Ri))+1
Инкремент указателя данных	INC DPTR	((DPTR)) ← ((DPTR))+1
Декремент аккумулятора	DEC A	(A) ← (A)-1
Декремент регистра	DEC Rn	(Rn) ← (Rn)-1
Декремент прямоадресуемого байта	DEC ad	(ad) ← (ad)-1
Декремент байта из внутреннего ОЗУ	DEC @ Ri	((Ri)) ← ((Ri))-1
Умножение аккумулятора и регистра В	MUL AB	(B)(A) ← (A)*(B)
Деление аккумулятора на регистр В	DIV AB	(B),(A) ← (A)/(B)

Команды логических операций

Название команды	Мнемоника	Операция
Логическое И аккумулятора и регистра (n=0…7)	ANL A, Rn	(A) ← (A) И лог (Rn)
Логическое И аккумулятора и прямоадресуемого байта	ANL A, ad	(A) ← (A) И лог (ad)
Логическое И аккумулятора с байтом из внутреннего ОЗУ (i=0,1)	ANL A, @Ri	(A) ← (A) И лог ((Ri))
Логическое И аккумулятора и константы	ANL A, #data8	(A) ← (A) И лог #data8
Логическое И прямоадресуемого байта и аккумулятора	ANL ad, A	(ad) ← (ad) И лог (A)
Логическое И прямоадресуемого байта и константы	ANL ad, #data8	(ad) ← (ad) И лог #data8
Логическое ИЛИ аккумулятора и регистра (n=0…7)	ORL A, Rn	(A) ← (A) ИЛИ лог (Rn)
Логическое ИЛИ аккумулятора и прямоадресуемого байта	ORL A, ad	(A) ← (A) ИЛИ лог (ad)
Логическое ИЛИ аккумулятора с байтом из внутреннего ОЗУ (i=0,1)	ORL A, @Ri	(A) ← (A) ИЛИ лог ((Ri))
Логическое ИЛИ аккумулятора и константы	ORL A, #data8	(A) ← (A) ИЛИ лог #data8
Логическое ИЛИ прямоадресуемого байта и аккумулятора	ORL ad, A	(ad) ← (ad) ИЛИ лог (A)
Логическое ИЛИ прямоадресуемого байта и константы	ORL ad, #data8	(ad) ← (ad) ИЛИ лог #data8

Исключающее ИЛИ аккумулятора и регистра (n=0…7)	XRL A, Rn	(A) ← (A) ИЛИ искл (Rn)
Исключающее ИЛИ аккумулятора и прямоадресуемого байта	XRL A, ad	(A) ← (A) ИЛИ искл (ad)
Исключающее ИЛИ аккумулятора с байтом из внутреннего ОЗУ (i=0,1)	XRL A, @Ri	(A) ← (A) ИЛИ искл ((Ri))
Исключающее ИЛИ аккумулятора и константы	XRL A, #data8	(A) ← (A) ИЛИ искл #data8
Исключающее ИЛИ прямоадресуемого байта и аккумулятора	XRL ad, A	(ad) ← (ad) ИЛИ искл (A)
Исключающее ИЛИ прямоадресуемого байта и константы	XRL ad, #data8	(ad) ← (ad) ИЛИ искл #data8
Сброс аккумулятора	CLR A	(A) ← 0
Инверсия аккумулятора	CPL A	(A) ← /(A)
Сдвиг аккумулятора влево циклический	RL A	(A.n+1) ← (A.n), n=0…6; (A.0) ← (A.7)
Сдвиг аккумулятора влево через перенос	RLC A	(A.n+1) ← (A.n), n=0…6; (A.0) ← (C); (C) ← (A.7)
Сдвиг аккумулятора вправо циклический	RR A	(A.n) ← (A.n+1), n=0…6; (A.7) ← (A.0)
Сдвиг аккумулятора вправо через перенос	RRC A	(A.n) ← (A.n+1), n=0…6; (A.7) ← (C); (C) ← (A.0)
Обмен местами тетрад в аккумуляторе	SWAP A	(A0, A1, A2, A3) ↔ (A4, A5, A6, A7)

Команды передачи данных

Название команды	Мнемоника	Операция
Пересылка в аккумулятор из регистра (n=0…7)	MOV A, Rn	(A) ← (Rn)
Пересылка в аккумулятор прямоадресуемого байта	MOV A, ad	(A) ← (ad)
Пересылка в аккумулятор байта из внутреннего ОЗУ (i=0, 1)	MOV A, @Ri	(A) ← ((Ri))
Загрузка в аккумулятор константы	MOV A, #data8	(A) ← #data8
Пересылка в регистр из аккумулятора	MOV Rn, A	(Rn) ← (A)
Пересылка в регистр прямоадресуемого байта	MOV Rn, ad	(Rn) ← (ad)
Загрузка в регистр константы	MOV Rn, #data8	(Rn) ← #data8
Пересылка по прямому адресу аккумулятора	MOV ad, A	(ad) ← (A)
Пересылка по прямому адресу регистра	MOV ad, Rn	(ad) ← (Rn)
Пересылка прямоадресуемого байта по прямому адресу	MOV add, ads	(add) ← (ads)
Пересылка байта из внутреннего ОЗУ по прямому адресу	MOV ad, @Ri	(ad) ← ((Ri))
Пересылка по прямому адресу константы	MOV ad, #data8	(ad) ← #data8
Пересылка во внутреннее ОЗУ из аккумулятора	MOV @Ri, A	((Ri)) ← (A)
Пересылка во внутреннее ОЗУ прямоадресуемого байта	MOV @Ri, ad	((Ri)) ← (ad)

Пересылка во внутреннее ОЗУ константы	MOV @Ri, #data8	((Ri)) ← #data8
Загрузка указателя данных	MOV DPTR, #data16	DPTR ← #data16
Пересылка в аккумулятор байта из памяти программ	MOVC A, @A+DPTR	(A) ← ((A)+(DPTR))
Пересылка в аккумулятор байта из памяти программ	MOVC A, @A+PC	(PC) ← (PC)+1 (A) ← ((A)+(PC))
Пересылка в аккумулятор байта из внешнего ОЗУ	MOVX A, @Ri	(A) ← ((Ri))
Пересылка в аккумулятор байта из расширенного внешнего ОЗУ	MOVX A, @DPTR	(A) ← ((DPTR))
Пересылка во внешнее ОЗУ из аккумулятора	MOVX @Ri, A	((Ri)) ← (A)
Пересылка в расширенное внешнее ОЗУ из аккумулятора	MOVX @DPTR, A	((DPTR)) ← (A)
Загрузка в стек	PUSH ad	(SP) ← (SP)+1; ((SP)) ← (ad)
Извлечение из стека	POP ad	(ad) ← ((SP)); ((SP)) ← (SP)-1
Обмен аккумулятора с регистром	XCH A, Rn	(A) ↔ (Rn)
Обмен аккумулятора с прямоадресуемым байтом	XCH A, ad	(A) ↔ (ad)
Обмен аккумулятора с байтом из внутреннего ОЗУ	XCH A, @Ri	(A) ↔ ((Ri))
Обмен младшей тетрады аккумулятора с младшей тетрадой байта внутреннего ОЗУ	XCHD A, @Ri	(A0…3) ↔ ((Ri0…3))

Команды операций с битами

Название команды	Мнемоника	Операция
Сброс переноса	CLR C	(C) ← (0)
Сброс бита	CLR bit	(bit) ← (0)
Установка переноса	SETB C	(C) ← (1)
Установка бита	SETB bit	(bit) ← (1)
Инверсия переноса	CPL C	(C) ← /(C)
Инверсия бита	CPL bit	(bit) ← /(bit)
Логическое И бита и переноса	ANL C, bit	(C) ← (C) И лог (bit)
Логическое И инверсии бита и переноса	ANL C, /bit	(C) ← (C) И лог /(bit)
Логическое ИЛИ бита и переноса	ORL C, bit	(C) ← (C) ИЛИ лог (bit)
Логическое ИЛИ инверсии бита и переноса	ORL C, /bit	(C) ← (C) ИЛИ лог /(bit)
Пересылка бита в перенос	MOV C, bit	(C) ← (bit)
Пересылка переноса в бит	MOV bit, C	(bit) ← (C)

Группа команд передачи управления

Название команды	Мнемоника	Операция
Длинный абсолютный переход в полном объеме памяти программ	LJMP ad16	(PC) ← (ad16)
Абсолютный переход внутри двухкилобайтной страницы	AJMP ad11	(PC) ← (PC)+2, затем (PC0…10) ← (ad11)
Короткий относительный переход внутри страницы 256 байт	SJMP rel	(PC) ← (PC)+2, затем (PC) ← (PC)+rel

Косвенный относительный переход	JMP @A+DPTR	(PC) ← (A)+(DPTR)
Переход, если аккумулятор равен 0	JZ rel	(PC) ← (PC)+2, затем если (A)=0, то (PC) ← (PC)+rel
Переход, если аккумулятор не равен 0	JNZ rel	(PC) ← (PC)+2, затем если (A)=0, то (PC) ← (PC)+rel
Переход, если перенос равен единице	JC rel	(PC) ← (PC)+2, затем если (C)=1, то (PC) ← (PC)+rel
Переход, если перенос равен нулю	JNC rel	(PC) ← (PC)+2, затем если (C)=0, то (PC) ← (PC)+rel
Переход, если бит равен единице	JB bit, rel	(PC) ← (PC)+3, затем если (bit)=1, то (PC) ← (PC)+rel
Переход, если бит равен нулю	JNB bit, rel	(PC) ← (PC)+3, затем если (bit)=0, то (PC) ← (PC)+rel
Переход, если бит установлен, с последующим сбросом бита	JCB bit, rel	(PC) ← (PC)+3, затем если (bit)=1, то (PC) ← (PC)+rel, (bit) ← 0
Декремент регистра и переход, если не нуль	DJNZ Rn, rel	(PC) ← (PC)+2, (Rn) ← (Rn)-1, затем если (Rn)=0, то (PC) ← (PC)+rel
Декремент прямоадресуемого байта и переход, если не нуль	DJNZ ad, rel	(PC) ← (PC)+2, (ad) ← (ad)-1, затем если (ad)=0, то (PC) ← (PC)+rel
Сравнение аккумулятора с прямоадресуемым байтом и переход, если не равно	CJNE A, ad, rel	(PC) ← (PC)+3, если A=(ad), то (PC) ← (PC)+rel, при этом, если (A)<ad, то (C) ← 1, иначе (C) ← 0
Сравнение аккумулятора с константой и переход, если не равно	CJNE A, #data8, rel	(PC) ← (PC)+3, если (A)=#data8, то (PC) ← (PC)+rel, при этом, если (A)<#data8, то (C) ← 1, иначе (C) ← 0
Сравнение регистра с константой и переход, если не равно	CJNE Rn, #data8, rel	(PC) ← (PC)+3, если (Rn)=#data8, то (PC) ← (PC)+rel, при этом, если (Rn)<#data8, то (C) ← 1, иначе (C) ← 0
Сравнение байта во внутреннем ОЗУ и переход, если не равно	CJNE @Ri, #data8, rel	(PC) ← (PC)+3, если ((Ri))=#data8, то (PC) ← (PC)+rel, при этом, если ((Ri))<#data8, то (C) ← 1, иначе (C) ← 0
Длинный абсолютный вызов подпрограммы	LCALL ad16	(PC) ← (PC)+3; (SP) ← (SP)+1 и ((SP)) ← (PC0…7); затем (SP) ← (SP)+1 и ((SP)) ← (PC8…15); затем (PC) ← ad16
Абсолютный вызов подпрограммы внутри двухкилобайтной страницы	ACALL ad11	(PC) ← (PC)+3; (SP) ← (SP)+1 и ((SP)) ← (PC0…7); Затем (SP) ← (SP)+1 и ((SP)) ← (PC8…15); затем (PC0…10) ← ad11
Возврат из подпрограммы	RET	(PC8…15) ← ((SP)); (SP) ← (SP)-1; затем (PC0…7) ← ((SP)); (SP) ← (SP)-1
Возврат из подпрограммы обработки прерывания	RETI	(PC8…15) ← ((SP)); (SP) ← (SP)-1; затем (PC0…7) ← ((SP)); (SP) ← (SP)-1
Пустая команда	NOP	(PC) ← (PC)+1

Влияние команд на изменение флагов

Команды	Модифицируемые флаги
ADD	CY, OV, AC
ADDC	CY, OV, AC
SUBB	CY, OV, AC
MUL	CY=0, OV
DIV	CY=0, OV
DA A	CY
RRC	CY
RLC	CY
SETB C	CY=1
CLR C	CY=0
CPL C	CY=/CY
ANL C, bit	CY
ANL C, /bit	CY
ORL C, bit	CY
ORL C, /bit	CY
MOV C, bit	CY
CJNE	CY