МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

АППАРАТНЫЕ И ПРОГРАММНЫЕ РЕШЕНИЯ ПРИ РАЗРАБОТКЕ УПРАВЛЯЮЩИХ УСТРОЙСТВ

Методические указания
к выполнению лабораторных работ
для студентов всех форм обучения по направлениям
12.03.01 – Приборостроение,
12.03.04 – Биотехнические системы и технологии,
15.03.04 – Автоматизация технологических процессов и производств,
15.03.06 – Мехатроника и робототехника

Севастополь
СевГУ
2023

УДК 004.353:621.316.34(076.5)
ББК 32.965.7я73
А769

Р е ц е н з е н т:
Н. И. Покинтелица – д-р техн. наук, профессор, заведующий кафедрой «Пищевые технологии и оборудование»

Составители: Л. Е. Карташов, Н. И. Чаленков

А769 Аппаратные и программные решения при разработке управляющих устройств : методические указания к выполнению лабораторных работ для студентов всех форм обучения по направлениям: 12.03.01 – Приборостроение, 12.03.04 – Биотехнические системы и технологии, 15.03.04 – Автоматизация технологических процессов и производств, 15.03.06 – Мехатроника и робототехника / Севастопольский государственный университет, Политехнический институт ; сост.: Л. Е. Карташов Н. И. Чаленков. – Севастополь : СевГУ, 2023. – 58 с. – Текст : электронный.

Целью методических указаний является оказание помощи обучающимся при выполнении лабораторных работ по дисциплине «Программируемые управляющие цифровые устройства». Излагаются основные теоретические сведения, порядок выполнения расчетов и требования к оформлению отчетов к лабораторным работам.

Предназначены для обучающихся всех форм обучения направлений подготовки 15.03.04 – Автоматизация технологических процессов и производств, 15.03.06 – Мехатроника и робототехника, 12.03.01 – Приборостроение и 12.03.04 – Биотехнические системы и технологии.

УДК 004.353:621.316.34(076.5)
ББК 32.965.7я73

Методические рекомендации рассмотрены и утверждены на заседании кафедры «Приборные системы и автоматизация технологических процессов», протокол № 7 от 28 февраля 2023 г.

© Карташов Л. Е., Чаленков Н. И., сост., 2023
© ФГАОУ ВО «Севастопольский государственный университет», 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4
1. ИССЛЕДОВАНИЕ ПРОСТРАНСТВА ВНУТРЕННЕЙ ПАМЯТИ ДАННЫХ 7
2. МОДЕЛИРОВАНИЕ РАБОТЫ ПУЛЬТА УПРАВЛЕНИЯ ОБОРУДОВАНИЕМ 9
3. РАЗРАБОТКА И ИССЛЕДОВАНИЕ ПРОЦЕДУРЫ СРАВНЕНИЯ ДАННЫХ 14
4. ИССЛЕДОВАНИЕ ПЕРЕДАЧИ ИНФОРМАЦИИ В ПОСЛЕДОВАТЕЛЬНОМ КОДЕ 18
5. ИССЛЕДОВАНИЕ МЕТОДОВ ПРОГРАММИРОВАНИЯ ИНДИКАЦИИ 25
6. ИССЛЕДОВАНИЕ СИСТЕМЫ ПРЕРЫВАНИЙ 35
7. СИНТЕЗ И ИССЛЕДОВАНИЕ РАБОТЫ АЦП ДВОЙНОГО ИНТЕГРИРОВАНИЯ 39
8. СИНТЕЗ И АНАЛИЗ ШИРОТНО-ИМПУЛЬСНОГО РЕГУЛЯТОРА 45
9. РАЗРАБОТКА ПРОГРАММЫ УПРАВЛЕНИЯ ОБОРУДОВАНИЕМ НА ОСНОВЕ ЦИКЛОГРАММЫ 56

ВВЕДЕНИЕ

Микропроцессорная система может рассматриваться как частный случай электронной системы, предназначенной для обработки входных сигналов и выдачи выходных сигналов. В качестве входных и выходных сигналов при этом могут использоваться аналоговые сигналы, одиночные цифровые сигналы, цифровые коды и последовательности цифровых кодов. Если система цифровая (а микропроцессорные системы относятся к разряду цифровых), то входные аналоговые сигналы преобразуются в последовательности кодов выборок с помощью АЦП, а выходные аналоговые сигналы формируются из последовательности кодов выборок с помощью ЦАП. Обработка и хранение информации производятся в цифровом виде.

Характерная особенность традиционной цифровой системы состоит в том, что алгоритмы обработки и хранения информации в ней жестко связаны со схемотехникой системы, то есть изменение этих алгоритмов возможно только путем изменения структуры системы, замены электронных узлов, входящих в систему, и связей между ними. Это практически невозможно сделать в процессе эксплуатации, поскольку нужен новый производственный цикл проектирования, изготовления и отладки всей системы. Именно поэтому традиционная цифровая система называется системой на «жесткой логике».

Любая система на «жесткой логике» обязательно представляет собой специализированную систему, настроенную исключительно на одну задачу или на несколько близких, заранее известных задач. С одной стороны, это имеет свои преимущества.

Во-первых, специализированная система (в отличие от универсальной) никогда не имеет аппаратной избыточности, то есть каждый ее элемент обязательно работает в полную силу.

Во-вторых, именно специализированная система может обеспечить максимально высокое быстродействие, так как скорость выполнения алгоритмов обработки информации определяется в ней только быстродействием отдельных логических элементов и выбранной схемой путей прохождения информации. А именно логические элементы всегда обладают максимальным на данный момент быстродействием.

Но в то же время большим недостатком цифровой системы на «жесткой логике» является то, что для каждой новой задачи ее надо проектировать и изготавливать заново. Это процесс длительный, дорогостоящий, требующий высокой квалификации исполнителей. А если решаемая задача изменяется, то вся аппаратура должна быть полностью заменена.

Путь преодоления этого недостатка очевиден: надо построить такую систему, которая могла бы легко адаптироваться под любую задачу, перестраиваться с одного алгоритма работы на другой без изменения аппаратуры. И задавать тот или иной алгоритм мы тогда могли бы путем ввода в систему некой дополнительной управляющей информации, **программы работы системы**.

Тогда система станет универсальной, или программируемой, не жесткой, а гибкой. Именно это и обеспечивает микропроцессорная система.

Но любая универсальность обязательно приводит к избыточности. Поэтому сложность универсальной системы должна быть такой, чтобы обеспечивать решение самой трудной задачи, а при решении простой задачи система будет работать далеко не в полную силу, будет использовать не все свои ресурсы. И чем проще решаемая задача, тем больше избыточность, и тем менее оправданной становится универсальность. Избыточность ведет к увеличению стоимости системы, снижению ее надежности, увеличению потребляемой мощности.

Кроме того, универсальность, как правило, приводит к существенному снижению быстродействия. Оптимизировать универсальную систему так, чтобы каждая новая задача решалась максимально быстро, попросту невозможно. Общее правило таково: чем больше универсальность, гибкость, тем меньше быстродействие. Более того, для универсальных систем не существует таких задач (пусть даже и самых простых), которые бы они решали с максимально возможным быстродействием.

Таким образом, можно сделать следующий вывод. Системы на «жесткой логике» хороши там, где решаемая задача не меняется длительное время, где требуется самое высокое быстродействие, где алгоритмы обработки информации предельно просты. А универсальные, программируемые системы хороши там, где часто меняются решаемые задачи, где высокое быстродействие не слишком важно, где алгоритмы обработки информации сложные.

Однако за последние десятилетия быстродействие универсальных (микропроцессорных) систем сильно выросло. К тому же большой объем выпуска микросхем для этих систем привел к резкому снижению их стоимости. В результате область применения систем на «жесткой логике» резко сузилась. Более того, высокими темпами развиваются сейчас программируемые системы, предназначенные для решения одной задачи или нескольких близких задач. Они удачно совмещают в себе как достоинства систем на «жесткой логике», так и программируемых систем, обеспечивая сочетание достаточно высокого быстродействия и необходимой гибкости.

В наибольшей степени все вышеперечисленные свойства проявляются в однокристальных микропроцессорах или, как их ещё называют, микроконтроллерах. В микроконтроллерах на одном кристалле объединяются все составные части компьютера: ядро микропроцессора, ОЗУ, ПЗУ, порты ввода-вывода и таймеры. Сегодня любой отважный и желающий может за сто рублей приобрести достаточно мощный микроконтроллер, а за большую, но доступную сумму – нечто, превосходящее по производительности первые суперкомпьютеры Сеймура Крея. Для массы задач, которые ставят перед разработчиками, бывает более чем достаточно производительность копеечной микросхемы, спроектированной 30 лет назад. И совершенно неважно, что разрядность её просто смешная – 8 бит, быстродействие в MIPS-единицах выглядит

просто неприлично, а объем памяти анекдотичен: десятки ведущих производителей продолжают штамповать подобные чипы в необъятных количествах, а инженеры с кажущимся необъяснимым упорством используют их в самых перспективных разработках.

Конечно, сформировалось племя «новых специалистов», в любой ситуации слепо следующих правилу «для решения задачи надо взять мощный ПК и найти программу, решающую задачу». С одной стороны, они радуют нас своей многочисленностью, а с другой – вызывают взрывоподобный рост стоимости реализаций самых примитивных проектов. С точки зрения нормального инженера, любой микропроцессор характеризуется множеством факторов, главными из которых являются: регистровая архитектура, система команд и особенности аппаратного интерфейса. Выражаясь человеческим языком, разработчика интересуют «внутренности» процессора, доступные программным средствам, возможности и удобство программирования и, наконец, приспособленность процессора к мирному сосуществованию с другими необходимыми аппаратными компонентами системы. Именно поэтому процессоры, попадающих в условный класс «малоразрядных», не собираются сдавать позиции в эпоху сверхдешёвых вычислений, а напротив – благодаря им стало возможным появление огромного числа сильно поумневших бытовых устройств, обеспечивающих доступный, массовый и ежедневный комфорт.

Лабораторные работы, представленные данных методических указаниях, в предназначен для закрепления знаний, полученных в теоретической части дисциплины, и приобретения навыков исследования микропроцессорных устройств на основе использования эмулятора. В качестве базовых использованы БИС микропроцессорной серии Intel 8048. Работы построены таким образом, что задания позволяют изучить язык программирования от нулевого уровня до уровня, позволяющего писать программы средней сложности. В начале каждой работы содержатся теоретические сведения и пояснения к ее выполнению.

1. ИССЛЕДОВАНИЕ ПРОСТРАНСТВА ВНУТРЕННЕЙ ПАМЯТИ ДАННЫХ

1.1 Цель работы

Исследовать структуру внутренней памяти данных микропроцессора Intel 8048, определить адреса регистров, изучить особенности организации регистров и стека микропроцессора. Изучить возможности использование дампов памяти и регистров для ускорения процесса отладки ассемблерных программ.

1.2 Теоретический раздел

Пространство внутренней памяти данных *DSEG* микропроцессора Intel 8048 (К1816ВЕ48) содержит 256 байт, адресуемых косвенно через два адресных регистра *R0, R1* выбранного в данный момент регистрового банка. В архитектуре отсутствует прямой способ адресации памяти, что привело к коротким однобайтовым форматам команд пересылки и обработки, за исключением команд с непосредственными данными.

Физическая оперативная память однокристальных микропроцессоров, как правило, не покрывает всего пространства внутренней памяти данных из-за нехватки площади кристалла. Так, в Intel 8048 только первые 64 байта покрыты физическим ОЗУ. С усовершенствованием интегральной технологии ожидается увеличение объемов внутреннего физического ОЗУ микропроцессоров и его постепенное приближение к потенциальному барьеру 256 байт. Например, в однокристальном микропроцессоре К1816ВЕ49, имеющем такую же базовую архитектуру, что и Intel 8048, объем физического ОЗУ на кристалле увеличен до 128 байт, а в Intel 8050 все пространство в 256 байт покрыто физическим ОЗУ.

Особенностью архитектуры Intel 8048 является совмещение обоих банков рабочих регистров и системного стека с пространством внутренней памяти данных, что дает возможность рассматривать хранящиеся в них объекты с двух сторон. Возможен единственный способ адресации внутренней памяти данных – косвенный регистровый по *R0* или *R1*. Однако часть ячеек все же оказывается доступной с помощью регистрового способа адресации, который может интерпретироваться как короткий прямой.

Совмещение системного стека и банков рабочих регистров с пространством внутренней памяти данных уменьшает и без того небольшой объем физического ОЗУ, свободного для хранения переменных.

В данной работе необходимо при помощи отладчика-эмулятора исследовать размещение регистров в оперативной памяти. Для этого используются дампы содержимого регистров и содержимого памяти.

Дампы содержимого регистров. Для анализа состояния регистров процессора необходимо отображать их содержимое на экране дисплея; такое отображение выполняется системными программами, которые выводят на экран имена регистров и их содержимое в шестнадцатеричном представлении. Затем пользователь может изменить содержимое регистра и продолжить выполнение программы с измененными данными.

Дампы содержимого памяти. Аналогично отображению содержимого регистров пользователь может выполнить дамп памяти (вывод содержимого запоминающих устройств). Осуществив дамп некоторой части запоминающих устройств, пользователь может проанализировать состояние соответствующего участка памяти и, если он имеет дело с ОЗУ, изменить содержимое. Удобно производить дамп целых блоков памяти, а не отдельных ее ячеек.

1.3 Задание.

1) Составить программу, заполняющую регистры банка *RB0* и банка *RB1* последовательностью чисел, начиная с числа NUM. Определить адреса ячеек внутренней памяти данных, которые соответствуют регистрам

1.4 Индивидуальное задание.

Число NUM определяется по формуле: NUM = i + j, где i, j – последние цифры номера зачетной книжки.

1.5 Содержание отчета.

1. Задание.
2. Текст программы на языке Ассемблер.
3. Адреса регистров во внутренней памяти данных.
4. Выводы по работе.

1.6 Контрольные вопросы.

1. Нарисуйте структурную схему Intel 8048 и объясните назначение всех функциональных узлов.
2. До каких предельных размеров возможно расширение памяти программ, памяти данных и линий ввода–вывода?
3. Какая опасность подстерегает программиста при использовании адресов памяти данных *DSEG* с 08h по 17h?
4. В чем особенности архитектуры Intel 8048?
5. Какие методы адресации к памяти данных существуют?
6. В чем особенность организации регистров микропроцессора Intel 8048?
7. Нарисуйте формат слова состояния программы и расскажите о флагах, устанавливаемых по результатам выполнения операций обработки данных в АЛУ.

2. МОДЕЛИРОВАНИЕ РАБОТЫ ПУЛЬТА УПРАВЛЕНИЯ ОБОРУДОВАНИЕМ

2.1 Цель работы

Исследовать методы опроса кнопок и композиции кнопок, присоединенных к микропроцессору Intel 8048, изучить возможности пошагового выполнения программ в эмуляторе.

2.2 Теоретический раздел

Для построения пультов управления оборудованием используются коммутационные элементы ручного управления. Под коммутацией понимают выполнение операций включения, отключения и переключения электрических цепей. Коммутационные элементы ручного управления срабатывают при непосредственном механическом воздействии на их органы управления. К коммутационным элементам с механическим приводом относятся рубильники, кнопки управления, микропереключатели, тумблеры, пакетные и универсальные переключатели, а также концевые и путевые переключатели.

Кнопка управления – это механическое устройство для передачи сигнала о вводе информации, элемент человеко-машинного интерфейса: элементарный физический механизм передачи электрического сигнала различным устройствам путём замыкания или размыкания двух, или более контактов. По своей сути кнопка является датчиком физической величины (усилия нажатия), передающим далее информацию о нажатии соединённым с ним устройствам. Два (или три, в случае переключающего) контакта, коммутирующих сигнальные линии в процессе нажатия кнопки, называются контактной группой. Кнопка может содержать как одну, так и несколько контактных групп – нормально разомкнутых, нормально замкнутых или переключающих, в любой комбинации.

Кнопка имеет два положения – «нажато» и «отпущено». В зависимости от состояния, в котором кнопка останется после снятия усилия, кнопки делятся на кнопки фиксирующегося и нефиксирующегося типа:

1) кнопка, возвращающаяся в исходное состояние после снятия приложенного усилия, является нефиксирующейся;
2) кнопка, изменяющая своё состояние на противоположное тому, что было до нажатия, и остающаяся в нём после снятия приложенного усилия, является фиксирующейся (кнопка с арретиром);
3) кнопка с зависимой фиксацией – фиксирующаяся кнопка, которая переходит в состояние «отпущено» при нажатии другой кнопки, имеющей с ней механическую связь.

Контакты кнопки по своему состоянию в положении «отпущено» делятся на нормально разомкнутые или нормально замкнутые:

1) нормально разомкнутые контакты не соединены (разомкнуты), когда кнопка находится в положении «отпущено»;
2) нормально замкнутые контакты соединены (замкнуты), когда кнопка находится в положении «отпущено».

При проектировании пульта управления достаточно распространенной задачей является задача подключение кнопок к микроконтроллеру. Несмотря на кажущуюся простоту, эта задача имеет некоторые особенности. Рассмотрим наиболее простой и распространенный вариант кнопки, имеющей нормально разомкнутые контакты, замыкаемые при нажатии на кнопку.

Если подключить один из контактов кнопки к общему проводу («земле»), а второй к выбранной линии порта ввода–вывода микроконтроллера, то выяснится, что такой метод не работает. При нажатии кнопки линия порта микроконтроллера соединяется с землей, и программа будет считывать логический «0» с этой линии порта ввода–вывода, но при отпущенной кнопке вывод микроконтроллера не будет соединен ни с чем, что часто называют «висит в воздухе». В таком случае программа может считать с вывода и логический «0» и логическую «1» случайным образом, поскольку на не присоединённой линии порта ввода–вывода появляются наводки.

Правильное решение предполагает, что в разомкнутом состоянии вывод микроконтроллера должен быть соединен через резистор, например, с шиной питания, а в замкнутом – с землей (рис. 2.1, а), либо наоборот (рис. 2.1, б).

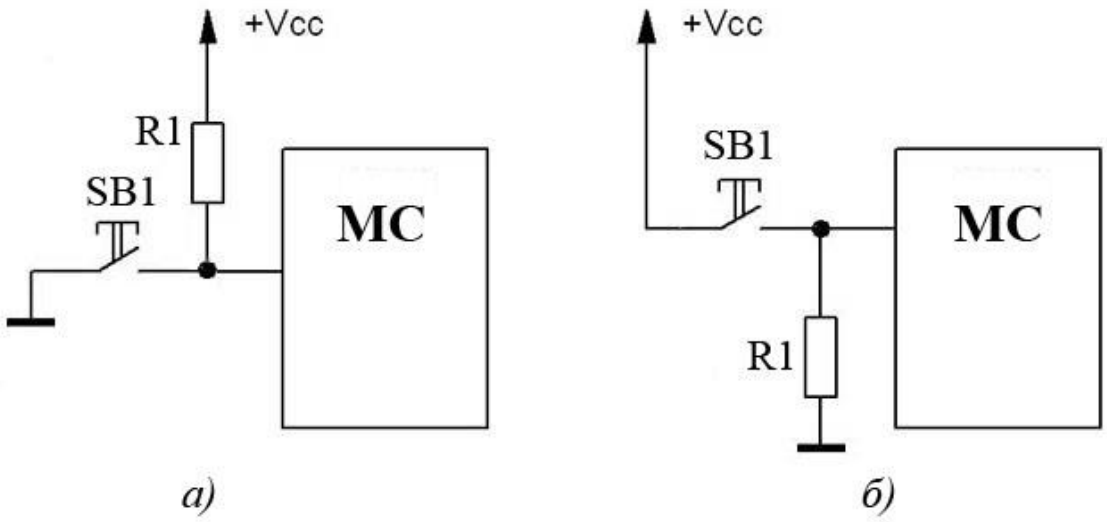

Рисунок 2.1 – Подключение кнопки к микроконтроллеру.

Сопротивление резистора не должно быть слишком мало, чтобы ток, текущий через резистор при замкнутых контактах, не был слишком большим. Обычно используют резисторы, сопротивление которых имеет порядок 10–100 кОм.

При разработке программ для пульта управления используются команды ввода–вывода, передачи управления и логические команды. Логические команды обычно применяются совместно с командами передачи управления для того, чтобы выделить определенные разряды слова; такое действие называют «маскированием». Предположим, например, что к входному порту подсоединяются восемь микропереключателей, и для того, чтобы проверить только

один разряд (один микропереключатель), процессор должен игнорировать значения всех остальных. На рисунке 2 изображена блок-схема программы, которая проверяет положение микропереключателя, связанного с разрядом 3 входного порта: если микропереключатель выключен, выходные светодиоды будут возбуждены, в противном случае они будут находиться в выключенном состоянии.

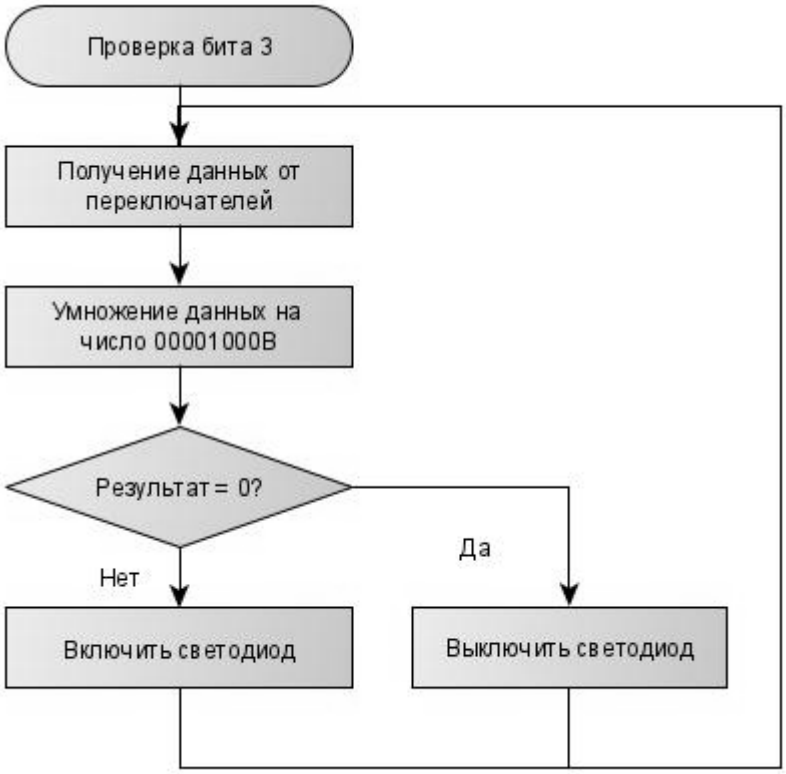

Рисунок 2.2 – Блок-схема алгоритма программы

Ниже приведен листинг фрагмента программы:

```
START:  IN A, P1          ; Считать данные из порта P1 в аккумулятор
        MOV R2, #08h      ; Установить маску в R2 (0000 1000)
        ANL A, R2         ; Установит все биты, кроме 3, в 0
        JZ OFF            ; Аккумулятор = 0?
ON:     ...               ; Включить светодиоды
        ...
        JMP START
OFF:    ...               ;  Выключить светодиоды
        ...
        JMP START
```

Микроконтроллеры, построенные на базе микропроцессоров, достаточно часто имеют дело не с одним датчиком или кнопкой, а с группой автономных (логически независимых) или взаимосвязанных (формирующих двоичный код) датчиков (группу взаимосвязанных датчиков называют компози-

цией). При этом микропроцессор может выполнять процедуру опроса датчиков или кнопок и передачи управления отдельным фрагментам прикладной программы в зависимости от принятого кода. При проектировании микропроцессорной системой управления пульт управления устройства с управляющими электроаппаратами (кнопки, переключатели и т.п.) можно рассматривать как двоичные датчики.

При опросе композиции двоичных датчиков передачу управления удобно осуществлять по таблице переходов. Программа осуществляет передачу управления одной из прикладных программ, которые расположены в пределах одной страницы памяти программ CSEG, в зависимости от кодовой комбинации, полученной от датчиков, подключенных к входам микропроцессора. Адрес строки таблицы, в которой хранятся адреса переходов, вычисляется как сумма относительного – внутри текущей страницы – начального адреса таблицы и кода, принятого от датчиков.

При отладке разрабатываемой в данной работе программы следует использовать режим пошагового выполнения программы в эмуляторе. Режим пошагового выполнения позволяет отследить изменения в ячейках памяти и регистрах после каждой команды процессора.

Кроме этого, возможен случай, когда к микроконтроллеру подключен датчик или кнопка, которые выдают импульсный сигнал. Ожидание импульсного сигнала предполагает, что микроконтроллер должна обнаружить не только факт появления, но и факт окончания сигнала. Для программной реализации этой процедуры можно смонтировать последовательно в линейную программу процедуры ожидания событий, причем последовательность процедур зависит от формы импульса. Для случая «положительного» импульса сначала должно следовать ожидание появления логической «1», а затем – ожидание логического «0», для «отрицательного» (инверсного) импульса очередность опроса обратная.

После отладки отдельных модулей программы во избежание напрасного расхода времени вряд ли целесообразно повторять их выполнение в пошаговом режиме. Многие современные отладчики позволяют задавать контрольные точки останова программы: средства, реализующие эту возможность, позволяют разработчику указывать адрес команды программы, с которой он будет начинать выполнение процедуры отладки. Затем программа выполняется безостановочно до тех пор, пока эмулятор не обнаружит контрольную точку останова. После останова пользователь может приступить к отладке в пошаговом режиме.

В используемом эмуляторе имеется возможность задания контрольных точек останова. После ввода программы и её отладки необходимо запустить программу в автоматическом режиме.

2.3 Задание.

Необходимо смоделировать работу пульта управления при следующих условиях:

2) На порт микропроцессорной системы управления поступает информация от восьми микропереключателей, входящих в состав пульта управления;
3) Если микропереключатель замкнут, то от него на пульт управления поступает сигнал высокого уровня («1»), низкий уровень («0») соответствует разомкнутому состоянию;
4) Если сигнал поступает с 1-го, 2-го или 3-го микропереключателя, то необходимо занести во внутреннюю память данных DSEG кодовую комбинацию CODE по адресу ADDR;
5) Если сигнал поступает с 4-го, 5-го, 6-го или 7-го микропереключателя, то необходимо в соответствующий порт P4, P5, P6 или P7 выдать двоичный код, равный номеру микропереключателя, подавшему сигнал;
6) Если сигнал подан восьмым микропереключателем, необходимо установить флаг F0.

2.4 Индивидуальное задание.

Кодовую комбинацию CODE вычисляются по формуле: $CODE = i + 2 \cdot j$, где i, j – последняя и предпоследняя цифра номера зачетной книжки. Адрес внутренней памяти данных ADDR вычисляют по формуле: ADDR = 025h + i, где i, j – последняя и предпоследняя цифра номера зачетной книжки. Если последняя цифра зачетки четная – микропереключатели подключены к порту P1, если нечетная – к порту P2.

2.5 Содержание отчета

1) Задание.
2) Графический алгоритм программы.
3) Текст программы на языке Ассемблер.
4) Выводы по работе.

2.6 Контрольные вопросы

1) Как программно определить состояние микропереключателя пульта управления?
2) Чем отличается обмен информацией с портами P3 – P7 от обмена с портами P1, P2 и BUS?
3) Каким образом внешняя память данных определяет обращение к ней?
4) В чем заключается операция маскирования?
5) Нарисуйте структурную схему Intel 8048 и объясните назначение всех функциональных узлов.
6) До каких предельных размеров возможно расширение памяти программ, памяти данных и линий ввода–вывода?
7) Объясните назначение режима пошагового выполнения программы, укажите его преимущества и недостатки.
8) Как задать точки останова программы?

3. РАЗРАБОТКА И ИССЛЕДОВАНИЕ ПРОЦЕДУРЫ СРАВНЕНИЯ ДАННЫХ

3.1 Цель работы

Изучить методы сравнения данных для однокристального микропроцессора Intel 8048, исследовать способы сравнения массива чисел.

3.2 Теоретический раздел

Особенностью системы команд однокристального микропроцессора Intel 8048 является отсутствие команд вычитания. Для реализации поставленной задачи – разработки процедуры сравнения – необходимо разработать подпрограмму, реализующую вычитание. Для начала вспомним, что числа, участвующие в вычитании, называются *уменьшаемым* и *вычитаемым*. Вычитаемое вычитается из уменьшаемого, а результат называется *разностью*. Пусть необходимо вычесть 176 из 253.

$$\begin{array}{rr} & 253 \\ - & 176 \\ \hline & xxx \end{array}$$

Начнем с крайнего правого столбца. Очевидно, что 6 больше 3, поэтому приходится занять 1 у 5 и вычесть 6 из 13. Получается 7. Поскольку мы заняли у пятерки единицу, она превратилась в 4. Эта цифра меньше 7, и мы занимаем 1 у 2 и вычитаем 7 из 14. Получаем 7. Вспоминаем, что заняли 1 у 2, так что вместо 2 имеем 1 и вычитаем 1 из 1. Получаем 0. Окончательный ответ 77.

Чтобы обойтись в вычитании без заимствования, во-первых, вычтем вычитаемое не из уменьшаемого, а из 999:

$$\begin{array}{rr} & 999 \\ - & 176 \\ \hline & 823 \end{array}$$

Мы используем 999, поскольку вычитаемое является трехзначным числом. Если бы оно было четырехзначным, мы вычли бы его из 9999. Результат вычитания числа из строки девяток называется дополнением до девяти (*nines' complement*).

Дополнение до девяти числа 176 есть 823. Верно и обратное: дополнение до девяти числа 823 есть 176. И что приятно: каким бы ни было вычитаемое, вычисление его дополнения до девяти никогда не требует заимствования.

Вычислив дополнение числа до девяти, прибавим к нему уменьшаемое:

$$\begin{array}{rr} & 253 \\ + & 823 \\ \hline & 1076 \end{array}$$

Наконец, сложив результат с 1 и вычтя 1000, получим 77. Мы получили тот же результат, что и раньше, без единого заимствования. Как это получилось? Исходная задача на вычитание выглядит так:

$$253 - 176$$

К этому выражению можно прибавить и вычесть любое число – результат от этого не изменится. Прибавим и вычтем 1000.

$$253 - 176 + 1000 - 1000$$

Это выражение эквивалентно выражению

$$253 - 176 + 999 + 1 - 1000$$

Эти числа можно перегруппировать:

$$253 + (999 - 176) + 1 - 1000.$$

Это как раз и есть тот расчет с использованием дополнение вычитаемого до девяти. Мы заменили одно вычитание двумя вычитаниями и двумя сложениями, попутно освободившись от всех заимствований.

При вычитании двоичных чисел используется точно такой же метод. Более того, он проще, чем в десятичном случае. Вот как выглядит наша задача на вычитание в двоичном представлении:

$$\begin{array}{rr} & 11111101 \\ - & 10110000 \\ \hline & \text{xxxxxxxx} \end{array}$$

Шаг 1. Вычтем вычитаемое из 11111111 (т. е. 255):

$$\begin{array}{rr} & 11111111 \\ - & 10110000 \\ \hline & 01001111 \end{array}$$

Когда мы работали с десятичными числами, вычитаемое вычиталось из строки девяток, а результат назывался дополнением до девяти. При расчетах с двоичными числами вычитаемое вычитается из строки единиц, и результат называется дополнением до единицы (*ones' complement*). Заметьте, чтобы получить дополнение до единицы, вычитать на самом деле не нужно. И вот почему: каждый 0 в исходном числе становится 1 в дополнении, а каждая 1 превращается в 0. Именно поэтому дополнение до единицы иногда также называют отрицанием (*negation*) или инверсией (*inversion*).

Шаг 2. Складываем дополнение вычитаемого до 1 с уменьшаемым:

$$\begin{array}{rr} & 11111101 \\ + & 01001111 \\ \hline & 101001100 \end{array}$$

Шаг 3. Прибавляем к результату 1:

$$\begin{array}{rr} & 101001100 \\ + & 00000001 \\ \hline & 101001101 \end{array}$$

Шаг 4. Вычитаем 100000000 (т. е. 256):

$$\begin{array}{rr} & 101001101 \\ - & 100000000 \\ \hline & 001001101 \end{array}$$

При использовании микропроцессора Intel 8048 шаг 4 является лишним. Микропроцессор является восьмиразрядным, поэтому 9 разряд не попадет в регистр или ячейку памяти, но приведет к установке флага переполнения CY. По флагу можно судить, какой результат (положительный или отрицательный) получился в результате операции. Если CY = 1 – результат положительный, если CY = 0, то отрицательный. Более того, нас интересует сравнение чисел, поэтому установка флага CY позволяет решить данную задачу.

Тот же результат можно получить сложением обратного кода уменьшаемого с вычитаемым с последующей инверсией суммы. В этом случае отсутствие флага переполнения будет свидетельствовать о том, что результат положительный, наличие – что результат отрицательный.

Для чего нужна использовать программа для сравнения чисел? Достаточно часто при разработке оборудования необходимо осуществлять контроль того, что полученные параметры какой-либо физической величины находятся в определенных заданных границах. Например, диаметр вала удовлетворяет заданному допуску на его изготовление, давление пациента не превышает определенных значений и т.п. При этом преобразование исходной физической величины в значение напряжения осуществляется при помощи датчика и преобразуется в цифровой код при помощи аналого-цифровых преобразователей (АЦП). Однако преобразование, которое осуществляет АЦП, не происходит мгновенно. Поэтому прежде, чем производить считывание данных с АЦП, нужно убедиться, что преобразование закончено и данные не изменяются. Для этого во многих АЦП предусмотрен выходной сигнал окончания преобразования, который необходимо подключить к микропроцессору через линию порта ввода-вывода или отдельный тестируемый порт.

3.3 Задание

Необходимо разработать программу, которая считывает данные из порта ввода-вывода, к которому присоединен АЦП, сравнивает полученное значение с двумя заданными границами, и выдает сигналы в зависимости от того, в какой диапазон попадает данное значение. Сравнение двух чисел необходимо оформить в виде подпрограммы.

3.4 Индивидуальное задание

1) Значения границ определить исходя из номера зачетной книжки. Первая граница равна двум первым цифрам зачетной книжки, вторая – двум последним.
2) Значения границ разместить в третьей странице памяти программ. Для этого лучше воспользоваться директивами ORG и DB. Первая директива определяет начальный адрес фрагмента программы, вторая – задает байт (или байты) в памяти. Например,

```
ORG 320h  ; Задание начального адреса фрагмента
DB 23h    ; Число 23 расположено в ячейке ПЗУ 320h
DB 0F1h   ; Число F1 расположено в ячейке ПЗУ 321h
```

3) Если последняя цифра зачетки четная – АЦП подключен к порту P1, а сигнал окончания преобразования ко входу T0, если нечетная – к порту P2 и входу T1.
4) Если данные, полученные с АЦП, меньше нижней границы, записать в порт P4 число Ah, если больше верхней границы, записать в порт P5 число Bh.
5) Для заинтересованных студентов предлагается выяснить, что происходит, если от меньшего числа вычитать большее.

3.5 Содержание отчета.

1) Задание.
2) Графический алгоритм программы.
3) Текст программы на языке Ассемблер.
4) Выводы по работе.

3.6 Контрольные вопросы

1) Чем отличается память программ и память данных?
2) Каким образом осуществляется доступ к внешней памяти программ и внешней памяти данных?
3) Как можно организовать сравнение двух чисел в микропроцессоре Intel 8048?
4) Какие методы адресации к памяти данных существуют?
5) Как получить данные из памяти программ?
6) Нарисуйте формат слова состояния программы и расскажите о флагах, устанавливаемых по результатам выполнения операций обработки данных в АЛУ.

4. ИССЛЕДОВАНИЕ ПЕРЕДАЧИ ИНФОРМАЦИИ В ПОСЛЕДОВАТЕЛЬНОМ КОДЕ

4.1 Цель работы

Исследовать способы передачи информации в последовательном коде посредством микропроцессора Intel 8048, составить и отладить программу, реализующую передачу информации в последовательном коде удаленному получателю в асинхронном режиме, при отладке программы изучить использование контрольных точек.

4.2 Теоретические сведения

Последовательная связь – это передача или прием, при котором данные передаются последовательно бит за битом, в противоположность параллельной передаче, при которой данные передаются побайтно или пословно.

Передать данные от микропроцессора к какому-либо другому устройству системы можно параллельно или последовательно. При параллельной передаче микропроцессор передает к внешнему устройству одновременно 8 бит данных по 8 линиям. При последовательной (не одновременной) передаче данных для передачи 8 бит данных используется одна линия.

Скорость передачи данных. Для передачи данных по одной линии необходимо определить частоту или скорость передачи данных. Скорость передачи v определяется как число битов данных, передаваемых по линии в 1 секунду.

Обычно скорость передачи данных составляет 110 и 300 бит/с (бод) при передаче данных к телетайпу и 1200, 2400, 4800 и 9600 бод при передаче между контроллерами и ЭВМ и наоборот. Предположим, что скорость передачи данных 1200 бод. Длительность импульса, соответствующая передаче 1 бит данных, определяется по формуле:

$$T = \frac{1}{v}.$$

На рисунке 1 показано, что две логические «1», передаваемые последовательно, воспринимаются как логическая «1» длительностью, равной утроенной длительности передачи 1 бит данных.

Рисунок 4.1 – Формат последовательных данных

Длительность передачи 1 байта данных со скоростью 1200 бод составит

$$T = \frac{1}{v} = \frac{8}{1200} = 6{,}7\ мс$$

При параллельной передаче данных эта же информация передается за несколько микросекунд.

Параллельно-последовательное преобразование данных. Для последовательной передачи данных вырабатываемое микропроцессором 8-разрядное слово нужно преобразовать в последовательность из 8 бит, передаваемых, например, со скоростью 9600 бод. Для этого нужно: выработать 8-разрядное слово; записать его в сдвиговый регистр (параллельно 8 бит); сдвигая 8-разрядное слово с выбранной частотой, выдавать с выхода регистра последовательно по одному биту.

Вырабатываемое микропроцессором 8-разрядное слово вводится в сдвиговый регистр, а затем последовательно сдвигается. Биты данных выдаются по одному с тактовой частотой, начиная с *D0* и кончая *D7*. Скорость передачи данных зависит от тактовой частоты сдвигового регистра. Используя микропроцессор, операцию преобразования параллельного кода в последовательный можно организовать, используя его средства.

Биты «старт», «стоп» и «паритет». Рассмотрим, как осуществляется прием и распознавание переданных данных. Предположим, что передается следующая кодовая комбинация:

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	0	0	0	1

Первым к какому-либо приемнику передается бит *D0*. Приемные устройства должны определить начало передачи данных. Предположим, например, что прием первой логической «1» или «0» воспринимается приемным устройством как момент начала передачи данных. Однако при этом не учитываются некоторые условия.

Во-первых, нужно предположить, что линия последовательной передачи данных находится в состоянии логической «1» или «0» тогда, когда на шине данных нет информации. Предположим, что, когда нет передачи данных, эта линия находится в состоянии логической «1». Чтобы определить момент начала передачи, приемное устройство должно зафиксировать изменение состояния этой линии с логической «1» в логический «0» и наоборот.

Однако если первый передаваемый бит *D0* равен логической «1», то момент начала передачи будет пропущен, поскольку состояние линии передачи данных, не изменится после передачи бита *D0*. В худшем случае передаваемая кодовая комбинация может состоять из одних единиц. Тогда ни один бит данных не будет принят приемным устройством.

Решить эту проблему можно, добавив к передаваемому цифровому слову дополнительные стартовые биты. Стартовые биты, вырабатываемые

передающим устройством, информируют принимающее устройство о начале передачи новой кодовой последовательности.

Когда нет передачи данных, последовательная линия связи должна находиться в одном из двух логических состояний. Предположим, что это состояние логической «1». Состояние стартового бита должно быть противоположно состоянию линии в режиме ожидания передачи. Приемное устройство воспринимает стартовый бит и начинает прием данных.

Кроме стартового бита, к передаваемой кодовой комбинации добавляется еще один разряд – бит паритета. Этот бит вырабатывается передающим устройством и используется приемником для контроля правильности приема кодовой комбинации. Передаваемое 8-разрядное цифровое слово может иметь четное или нечетное число логических «1». Например, в слове 1Ah три логических «1», а в слове 3Fh шесть логических «1». Приемное устройство не только принимает данные, но и настроено на прием нечетного или четного числа логических «1».

Предположим, что приемник настроен на прием четного числа логических 1. Передающее устройство всегда должно передавать цифровые слова с четным числом логических «1». Не все 8-разрядные слова содержат четное число «1», поэтому при передаче цифрового слова передающее устройство должно вырабатывать дополнительный бит – бит паритета. Использование бита паритета для контроля правильности приема данных при последовательной передаче позволяет обнаруживать одиночные ошибки, т. е. такие ошибки, когда одна из переданных логических «1» была неверно принята как логический «0» или наоборот.

Бит паритета имеет такое логическое состояние, что передаваемое 8-разрядное слово всегда содержит четное число логических «1». Если передаваемое слово содержит нечетное число единиц, то бит паритета будет установлен в логическую «1», а если четное – в логический «0». Бит паритета дополняет число передаваемых «1» до четного или нечетного, в зависимости от типа контроля, реализованного в приемном устройстве.

Последний бит передаваемой кодовой комбинации называется стоповым битом. Стоповый бит используется для обозначения окончания передачи кодовой комбинации и вырабатывается передающим устройством. Приемное устройство ожидает прихода стопового бита в конце каждого передаваемого цифрового слова. Если этот бит не пришел, приемное устройство может выдать сигнал ошибки передачи данных. Стоповый бит, так же, как и бит паритета, служит для обнаружения ошибок передачи данных.

При различных типах последовательной передачи данных используется различное число стоповых битов: 1; 1,5 или 2. Полное цифровое слово, содержащее один стоповый бит, показано на рисунке 1. Общее число передаваемых битов данных равно 12. Длительность передачи этого слова со скоростью 2400 бод:

$$T = \frac{1}{v} = \frac{12}{2400} = 0{,}055\,мс\ .$$

Поскольку передача битов в последовательном коде должна осуществляется с определенной длительностью, необходимо реализовать подпрограмму задержки времени. Для реализации временной задержки между двумя фрагментами программы можно использовать таймер микропроцессора, работающий в режиме таймера (временные интервалы пропорциональны длительности машинного цикла) или в режиме счетчика (временные интервалы пропорциональны периоду сигналов от внешнего генератора, подаваемым на входы T0 или T1).

Например, для того, чтобы организовать временную задержку длительностью T с использованием таймера, необходимо настроить его на работу в режиме таймера, загрузить байт константы пересчета (K) в таймер, запустить таймер, а затем перейти в режим ожидания (или холостого хода) до появления флажка переполнения таймера TF. Программа обработки прерывания от таймера должна прекращать работу таймера. Эта процедура реализуется следующей последовательностью команд:

```
        ORG 100H        ; Адрес программы задержки времени
DELAY:  MOV A, # 156    ; Предустановка
        MOV T, A        ; таймера
        STRT T          ; Запуск таймера
WAIT:   JTF ENDT        ; Ожидание переполнения таймера
        JMP WAIT        ; таймера
ENDT:   STOP TCNT       ; Останов таймера
        RETR            ; Выход из программы


        ORG 30H         ; начальный адрес программы
        ANL P1, #FEH    ; Установка P1.0 в 0
        CALL DELAY      ; Вызов подпрограммы задержки
        ORL P1, 01H     ; Установка P1.0 в 1
```

Временная задержка определяется временем работы таймера до переполнения (256 – K) отсчетов, а также временем выполнения команд подготовки таймера к работе. Таймер увеличивает своё значение на 1 за 80 мкс. При достижении 255 увеличение значения таймера на 1 вызывает его переполнение и установку флага TF. Если мы записали в таймер число 156, то до его переполнения произойдет 100 отсчетов таймера. Тогда время передачи одного бита составит

$$T = 100 \cdot 80 = 8000 \text{ мкс.}$$

Из этого выражения можно определить значение скорости передачи:

$$T = \frac{1}{v} = \frac{1}{8000 \cdot 10^{-6}} = 125 \text{ (бод).}$$

Поскольку при разработке программы в данной работе будет присутствовать большое количество циклов, и поскольку эмулятор не позволяет исследовать состояние портов ввода-вывода в реальном времени, то необходимо воспользоваться точками прерываний. Процедура пошагового выполнения программы оказывается слишком медленной и бесполезной в случае, когда программа должна выполниться в рабочем режиме. Поэтому желательно иметь возможность безостановочного выполнения программы до тех пор, пока не встретится какая-то определенная команда. При выполнении команды останова программы в заданной точке управление будет передано управляющей программе отладчика, и пользователь сможет проанализировать состояние процессора, что позволит оценить правильность выполнения программы.

Средства обеспечения контрольных остановов в заданных точках программ очень полезны для организации их отладки, поскольку до момента останова участки программы любого размера могут прогоняться с максимальной скоростью. Например, некоторая подпрограмма может быть выполнена целиком, а затем в точке останова можно проанализировать состояние регистров процессора с целью установления правильности выполнения подпрограммы.

И в заключении, немного о согласовании уровня сигналов. Микропроцессор имеет ТТЛ уровни своих выходных сигналов. При необходимости, когда внешнее устройство – приемник сигналов требует более высоких уровней логических сигналов (например, уровней КМОП), то можно использовать стандартные схемы в зависимости от номинала питания КМОП-схем.

Чаще всего дополнительные схемы согласования уровней при построении систем сбора и обработки информации на основе микропроцессоров требуются как раз при обмене данными по последовательному каналу. Такой обмен требует соблюдения требований интерфейса RS-323. Последовательный порт микропроцессора имеет уровни ТТЛ: «1» – высокий уровень, «0» – низкий уровень. RS-232 требует для передачи «1» уровня – 12В, для передачи «0» – уровня +12В (см. рис. 4.2).

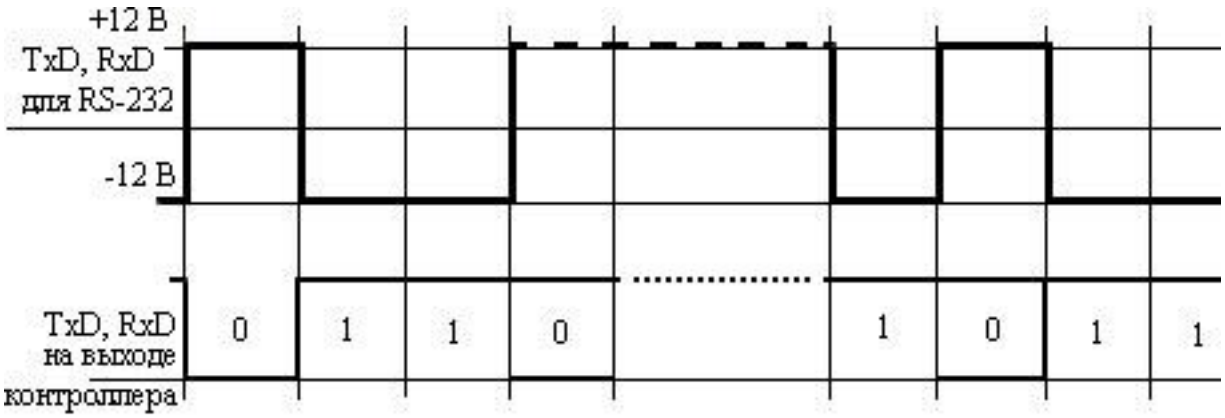

Рисунок 4.2 – ТТЛ и RS-323 сигналы.

Согласование сигналов может быть выполнено в соответствии со схемой на основе элементов из серии К170. В качестве передатчиков и приемников могут использоваться и другие схемы (например, передатчик – SN75150, приемник – SN75154, или схема MAX232 (рисунок 4.3), содержащая и передатчик, и приемник).

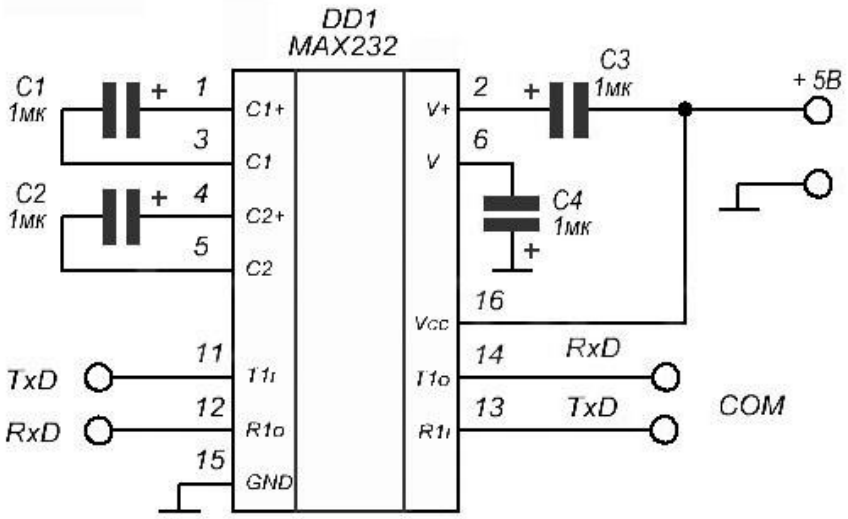

Рисунок 4.3 – Согласование сигналов на основе элемента MAX232A

Если передача осуществляется на большое расстояние, то она должна выполняться в соответствии со стандартом «токовая петля». Для формирования токовых посылок может использоваться схема на основе транзистора и оптрона (рисунок 4.4).

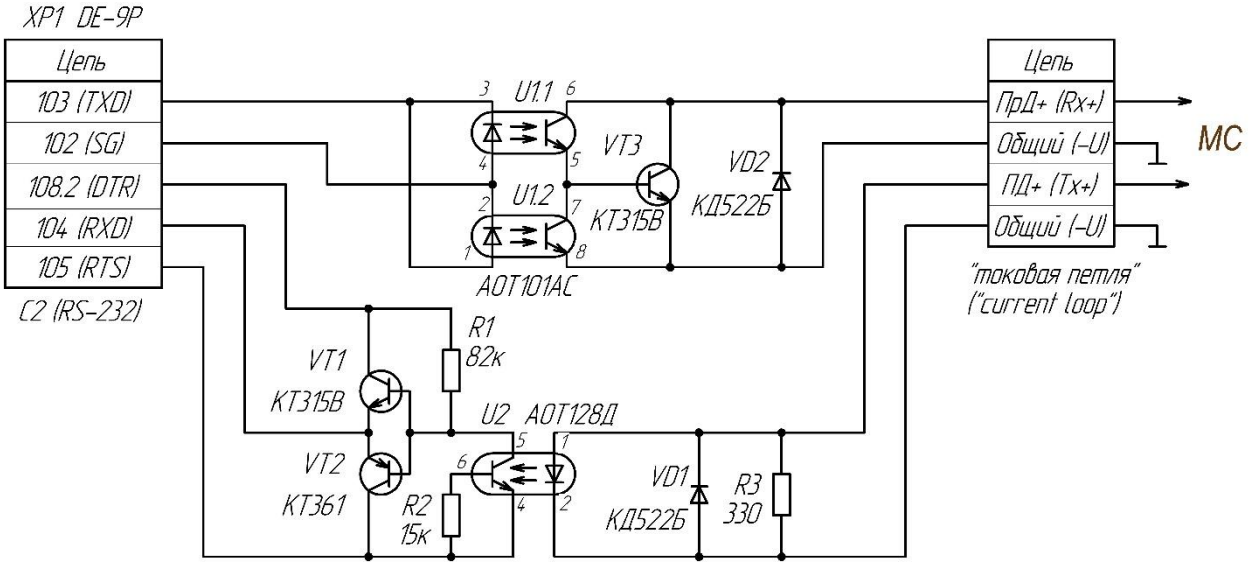

Рисунок 4.4 – Согласование сигналов по стандарту «токовая петля»

4.3 Задание

1. Разработать подпрограмму вычисление арифметического выражения:

$$\mathbf{DW = i + j \cdot 8},$$

где *i, j* – последние цифры номера зачетной книжки.

2. Разработать подпрограмму, реализующую передачу результата удаленному получателю по двухпроводной линии передачи. Передачу информации осуществляется в последовательном коде, при асинхронном режиме обмена. Предполагается, что линия передачи информации подключена к четвертому разряду порта **P2**, а оставшиеся разряды могут быть использованы для подключения датчиков и внешних исполнительных устройств.
3. Измерить действительную скорость передачи информации, определить относительную погрешность действительной скорости относительно теоретической.

4.4 Индивидуальное задание

Скорость передачи информации по линии связи выбрать из таблицы в соответствии с последней цифрой номера зачетной книжки.

Последняя цифра номера зачетки									
1	2	3	4	5	6	7	8	9	0
Скорость передачи, Бод									
600	3200	300	2 400	800	1 200	400	1600	150	750

4.5 Содержание отчета

1. Задание.
2. Текст программы на языке Ассемблер.
3. Временная диаграмма сигнала, поступающего в линию связи.
4. Расчет относительной погрешности скорости передачи.
5. Выводы по работе.

4.6 Контрольные вопросы

1. Расскажите о формате кадра последовательного кода.
2. Каким образом можно реализовать умножение числа на 2 в микропроцессоре Intel 8048?
3. Какова длительность периода увеличения содержимого таймера на единицу?
4. Каким образом можно осуществить преобразование параллельного кода в последовательный?
5. Какой способ останова программы в контрольных точках применяется в используемом эмуляторе?
6. Расскажите о порядке работы с точками прерываний.
7. За счет чего возникает погрешность скорости передачи?

5. ИССЛЕДОВАНИЕ МЕТОДОВ ПРОГРАММИРОВАНИЯ ИНДИКАЦИИ

5.1 Цель работы

Исследовать методы программирования операций ввода-вывода микропроцессора Intel 8048, изучить особенности работы портов P1 и P2, исследовать команды, позволяющие осуществлять одновременный ввод и вывод сигналов через порты, изучить способы подключения 7-сегментных индикаторов к микропроцессору.

5.2 Введение

Специальная схемотехника портов P1 и P2, которая получила название квазидвунаправленной, позволяет выполнять ввод, вывод или ввод-вывод. Каждая линия портов P1 и P2 может быть программным путем настроена на ввод, вывод или на работу с двунаправленной линией передачи. Для того чтобы настроить некоторую линию на режим ввода в микропроцессор, необходимо перед этим в буферный триггер этой линии записать логическую «1». Сигнал RESET автоматически записывает во все линии портов P1 и P2 сигнал «1». Квазидвунаправленная структура портов P1 и P2 для программиста Intel 8048специфична тем, что в процессе ввода информации выполняется операция логического «И» над вводимыми данными и текущими (последними) выведенными данными. Квазидвунаправленные схемы портов P1 и P2 и команды логических операций ANL и ORL предоставляют разработчику эффективное средство маскирования для обработки однобитных входных и выходных переменных.

В системе команд микропроцессора Intel 8048 есть команды, которые позволяют выполнять запись нулей и единиц в любой разряд или группу разрядов порта, но так как в этих командах маска задается непосредственным операндом, то необходимо знать распределение сбрасываемых и устанавливаемых линий на этапе разработки прикладной программы. В том случае, если маска вычисляется программой и заранее не известна, в ОЗУ необходимо иметь копию состояния порта вывода. Эта копия по командам логических операций объединяется с вычисляемой маской в аккумуляторе и затем загружается в порт. Необходимость этой процедуры вызвана тем, что в микропроцессоре отсутствует возможность выполнить операцию чтения значений портов P1 и P2 для определения прежнего состояния порта вывода. Порт P2 отличается от порта P1 тем, что его младшие четыре бита могут быть использованы для расширения микропроцессорных систем по вводу/выводу. Через младшую тетраду порта P2 по специальным командам обращения возможен доступ к четырем внешним четырехбитным портам ввода/вывода P4 – P7. Работа этих внешних портов синхронизируется сигналом PROG.

Многие микропроцессорные устройства требуют наличия только простейшей индикации типа «ДА/НЕТ», «ВКЛ/ВЫКЛ». Такая индикация реализуется на основе отдельных светодиодов. Для отображения цифровой и буквенной информации широко используются 7-сегментные индикаторы. Семь

отображающих элементов позволяют высвечивать десятичные и шестнадцатеричные цифры, некоторые буквы русского и латинского алфавитов, а также некоторые специальные знаки. Скажем, нам нужно организовать вывод какого-то числа на 7-сегментные индикаторы. Ну, например, мы делаем часы, или термометр, или еще чего-нибудь. Давайте посчитаем, сколько понадобится проводов, чтобы подключить индикаторы. Кстати, индикаторы выглядят примерно вот так:

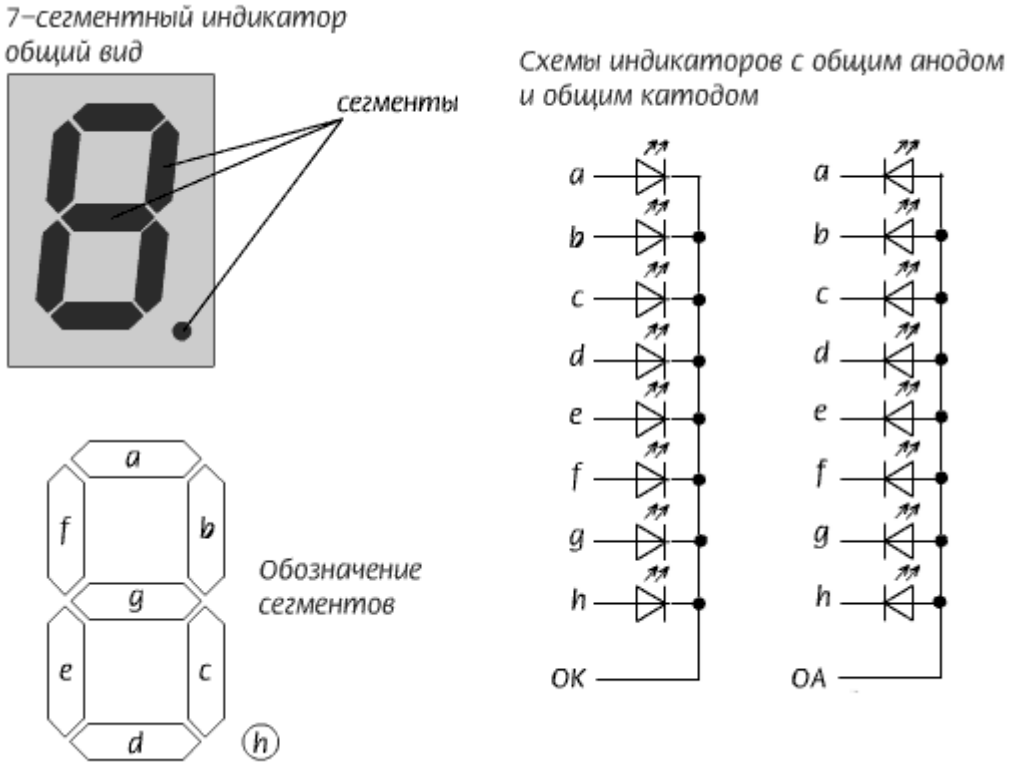

Рисунок 5.1 – 7-сегментный индикатор

Уделим пару слов увиденному. Перед нами – самый обычный светодиодный 7-сегментный индикатор. Вопрос: почему 7-сегментный, когда сегментов на самом деле – восемь? Ответ: потому что 8-й сегмент – это точка, которая не входит в изображение цифры и вообще, является необязательной. Бывают индикаторы и без точек. Для нумерации сегментов используются латинские буквы от A до H. У всех всех-всех 7-сегментных индикаторов в нашем бренном мире, сегменты нумеруются в том порядке, как это показано на рисунке, и никак иначе.

Теперь смотрим на схемы. Во-первых, почему их две? Да потому что индикаторы бывают разные. Бывают – с общим катодом (ОК), бывают – с общим анодом (ОА). Среди забугорных более распространены, как ни странно, индикаторы с ОА. Но встречаются и с ОК. Так что – нужно быть осторожным, чтоб не спутать. Хотя, нам-то все равно. Нашими индикаторами будет управлять микропроцессор. А его можно запрограммировать как на работу с ОК, так и с ОА. Тут сложностей нет.

У каждого индикатора – 9 ножек. С 1 по 8 – выводы сегментов, 9-я – общий. Допустим, мы хотим отображать 4-разрядное число. Надо 4 индикатора. Арифметика маленького Пети: $8 \cdot 4 = 32$. То есть, нам потребуется 32

провода (не считая общие). Неслабо? Ничуть. Тем более, учитывая, что у большинства однокристальных микропроцессоров количество каналов ввода/вывода как раз равно 32 (а в Intel 8048 и вообще всего 24). Как-то меня не очень греет перспектива угрохать все выводы микропроцессора на индикацию. А вас? Ведь надо ж еще куда-то подключить кнопочки, ручечки, датчички и пр. пр. пр., а мало ли чего ещё?

Существует еще одна проблема. Микроконтроллер передает данные внешнему устройству, 7-сегментному индикатору. Подключая к микроконтроллеру внешние микросхемы, всегда нужно проверять, согласуются ли уровни ее входных напряжений с выходными напряжениями микроконтроллера. Для этого находим описание микроконтроллера, желательно оригинальное, в описании с трудом находим раздел электрических характеристик, а там приведены значения выходных напряжений логического нуля V_{ol} (output low voltage) и единицы V_{oh} (output high voltage). Весьма забавно, что данные приведены для определенного напряжения питания и токов, даны минимальные и максимальные значения. Например, в таком виде:

Таблица 5.1 – D.C. and operating characteristics

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V_{OL}	Output Low Voltage (BUS, RD, WR, PSEN, ALE)			0.45	V	I_{OL}= 1.6 mA
V_{OL1}	Output Low Voltage (P1.0-P1.7, P2.0-P2.7)			0.45	V	I_{OL}= 1.2 mA
V_{OL2}	Output Low Voltage (All Other Outputs)			0.45	V	I_{OL}= 0.8 mA
V_{OH}	Output High Voltage (BUS)	2.4			V	I_{OH}= - 0.28 mA
V_{OH1}	Output High Voltage (RD, WR, ALE, PSEN)	2.4			V	I_{OH}= – 0.08 mA
V_{OH2}	Output High Voltage (All Other Outputs)	2.4			V	I_{OH}= – 0.03 mA

Уровни напряжений для других условий можно оценить по графикам «I/O Pin Current vs Output Voltage» (рис. 5.2).

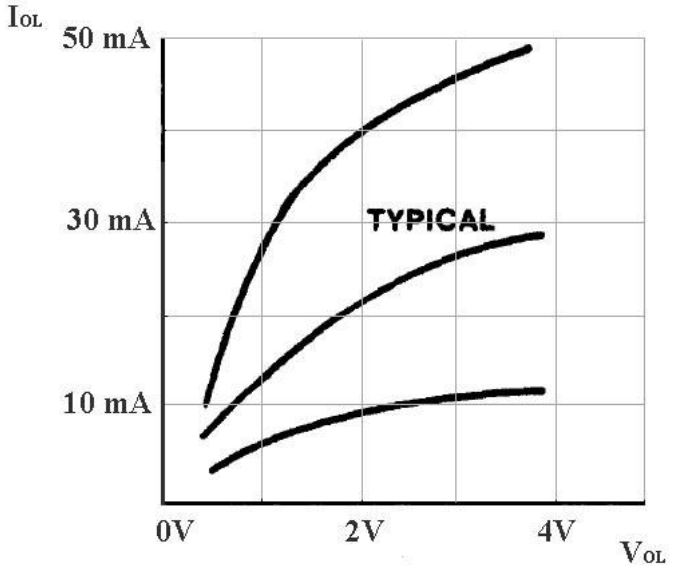

Рисунок 5.2 – Зависимость выходного напряжения от втекающего тока.

Микроконтроллер может быть или источником, или потребителем тока (то есть ток или вытекает из него, или втекает). Это зависит от того, какой логический уровень установлен на выводе – ноль или единица. Если на выводе логический ноль, то любое положительное напряжение, приложенное к выводу, вызовет втекающий ток. Если на выводе логическая единица, то любое внешнее напряжение, меньше напряжения питания микроконтроллера, вызовет вытекающий ток. Протекание тока сопровождается падением напряжения на внутренних цепях вывода микроконтроллера. Величину падения можно оценить по тем же графикам.

Мы видим, что графики приведены для диапазона токов от 0 до 50 мА, но это вовсе не означает, что через каждый из выводов микроконтроллера может протекать такой ток. Absolute Maximum Ratings в разделе электрических характеристик говорит нам, что максимальный ток через один вывод микроконтроллера не должен превышать 40 мА, а ток по цепи питания не должен превышать 200 мА. Это позволяет оценить, какую нагрузку микроконтроллер способен запитать своими силами. Например, мы можем подключить 8 ярких светодиодов при токе 20 мА, и микроконтроллер не сгорит. При этом для расчета ограничительных резисторов нужно будет учитывать падение напряжения (~ 0,6 В) на выходе микроконтроллера.

$$8 \cdot 20 \text{ мА} = 160 \text{ мА}$$

Добавим 10 мА собственного потребления, итого: 170 мА < 200 мА.

$$R = (5 – 0{,}6 – V_d) / 20 \text{ мА},$$

где V_d – прямое падение напряжения на светодиоде.

Подключим 7-сегментный индикатор непосредственно к микропроцессору. Это стандартная схема с раздельным включением/выключением

сегментов. При логической «1» на выходе микропроцессора соответствующий сегмент светится, при логическом «0» – погашен. Сопротивление резисторов R1 ... R8 выбирается так, чтобы суммарный ток через все линии при логической «1» не превышал допустимое по значение 200 мА. Для индикатора с общим катодом схема представлена на рис. 5.3.

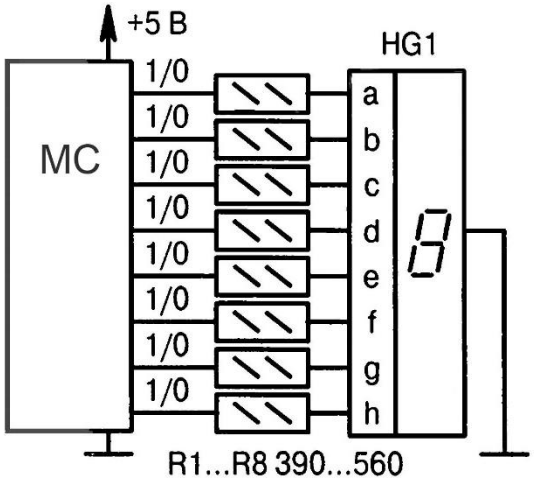

Рисунок 5.3 – Подключение индикатора с общим катодом к микропроцессору

Для индикатора с общим анодом решение будет аналогичным (см. рисунок 5.4).

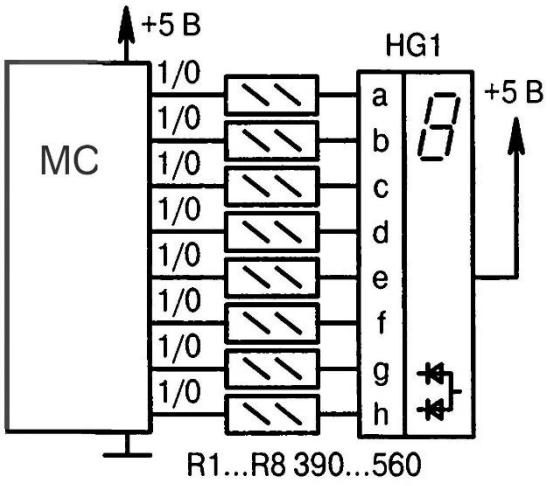

Рисунок 5.4 – Подключение индикатора с общим анодом к микропроцессору

Можно слегка сэкономить на резисторах. Поставим вместо восьми резисторов, как на рис. 5.4. только один, но с меньшим сопротивлением. Однако, как следствие, цифры с большим и малым числом сегментов будут иметь разную яркость, например, «1» и «8». Чтобы выровнять яркости, применяют циклы импульсов с одинаковой длительностью, но с разной частотой. Например, для высвечивания цифры «4» выдаётся «бегущая единица» по сегментам «b», «c», «f», «g» со средней частотой. Для цифры «8» частота будет максимальная, а для цифры «1» — минимальная, но выше 25 Гц, чтобы

глаз не замечал мерцание. Реализация такого решения представлена на рисунке 5.5.

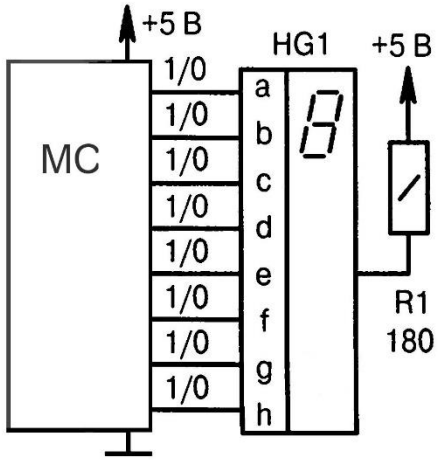

Рисунок 5.5 – Подключение индикатора с общим анодом к микропроцессору

Но можно ли назвать приведенные решения хорошими? Для засветки одного сегмента большинства типов 7-сегментных индикаторов необходимо обеспечить протекание через сегмент тока 10 – 15 мА при напряжении 2,0 – 2,5 В. Низкая нагрузочная способность микропроцессора ограничивает возможность прямого подсоединения с 7-сегментным индикатором. Второй индикатор уже проблематично подключить из-за ограничений по току. Значит, надо использовать промежуточные усилители. В качестве промежуточных усилителей тока могут использоваться логические элементы серии К155. Или для подключение яркого индикатора к источнику повышенного напряжения +9 ... +12В можно использовать БИС К1109КТ63. Способ такого подключения показан на рисунке 5.6.

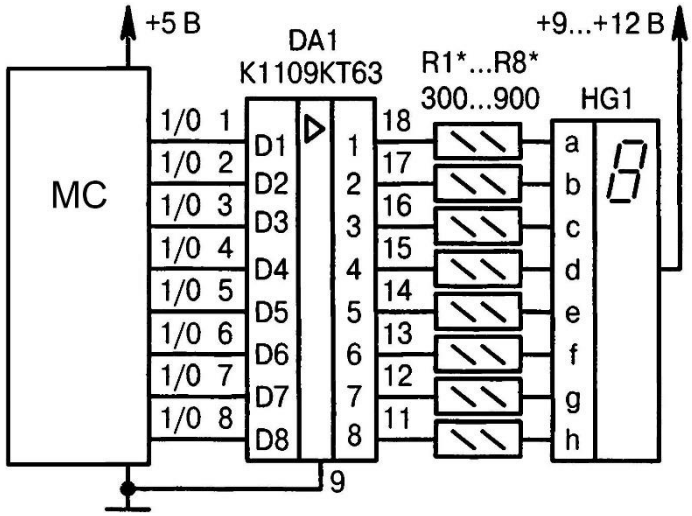

Рисунок 5.6 – Подключение индикатора с помощью БИС К1109КТ63

Если сегменты индикатора потребляют большой ток, например, 50 мА, то следует уменьшить сопротивление резисторов R1... R8 и увеличить их

мощность. Сборка транзисторных ключей DA1 может быть заменена аналогичной зарубежной ULN2003. Или можно заменить это отдельными транзисторами КТ817 с резисторами 1 ... 10 кОм между базой и соответствующей линией.

Но на этом проблемы не кончаются. Нам необходимо осуществить преобразование двоичных кодов в коды для 7-сегментного индикатора. Его можно осуществить либо при помощи программы, либо с использованием преобразователей К514ИД1, К514ИД2, 133ПП4, 564ИД5. Кстати, интегральные схемы преобразователей кодов также позволяют повысить нагрузочную способность портов ввода-вывода микропроцессора. Подключение одного 7-сегментного индикатора к микропроцессору с помощью дешифратора КР514ИД2 показано на рисунке 5.7.

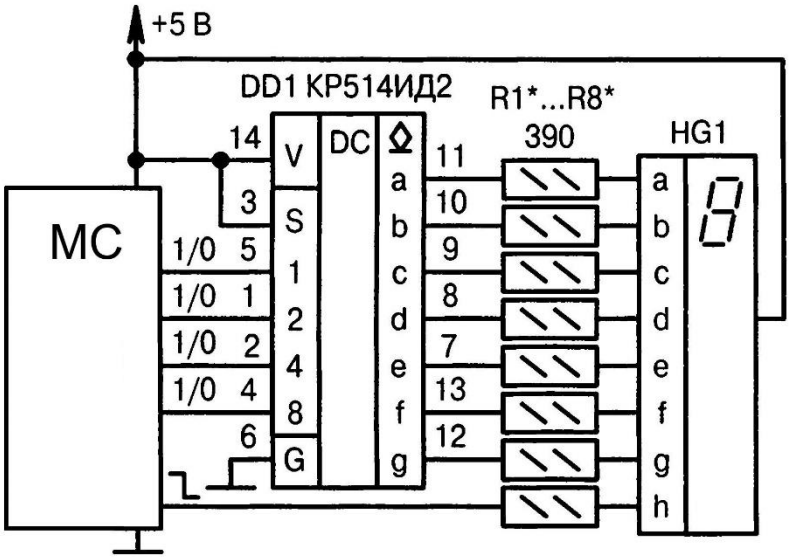

Рисунок 5.7 – Подключение индикатора с помощью дешифратора КР514ИД2

Кроме преобразования кодов мы получим сокращение числа выходных линий микропроцессора за счёт специализированного дешифратора DD1, имеющего выходы с открытым коллектором.

Но что делать, если надо вывести число из нескольких цифр? Для отображения такой информации используются линейные (однострочные) дисплеи. Такие дисплеи представляют собой «линейку», смонтированную из отдельных 7-сегментных индикаторов. Число знакомест дисплея определяется в соответствии с требованиями к микропроцессорной системе. А хватит ли нам выходов микропроцессора? Каким же методом можно добиться уменьшение числа выходов микропроцессора, задействованных для индикации?

Существует два способа организации интерфейса микропроцессора с линейным дисплеем: статический и динамический. Первый (*статический*) требует наличия на входах каждого индикатора специальных буферных регистров для хранения кодов выводимых символов. Естественно, что с увеличением разрядности дисплея возрастает число дополнительных микросхем, следовательно, и стоимость микропроцессорной системы. Но с другой стороны, ис-

пользование буферных регистров разгружает процессор и существенно упрощает программу. Для любителей пользоваться паяльником такая схема подключения для двух индикаторов приводится ниже.

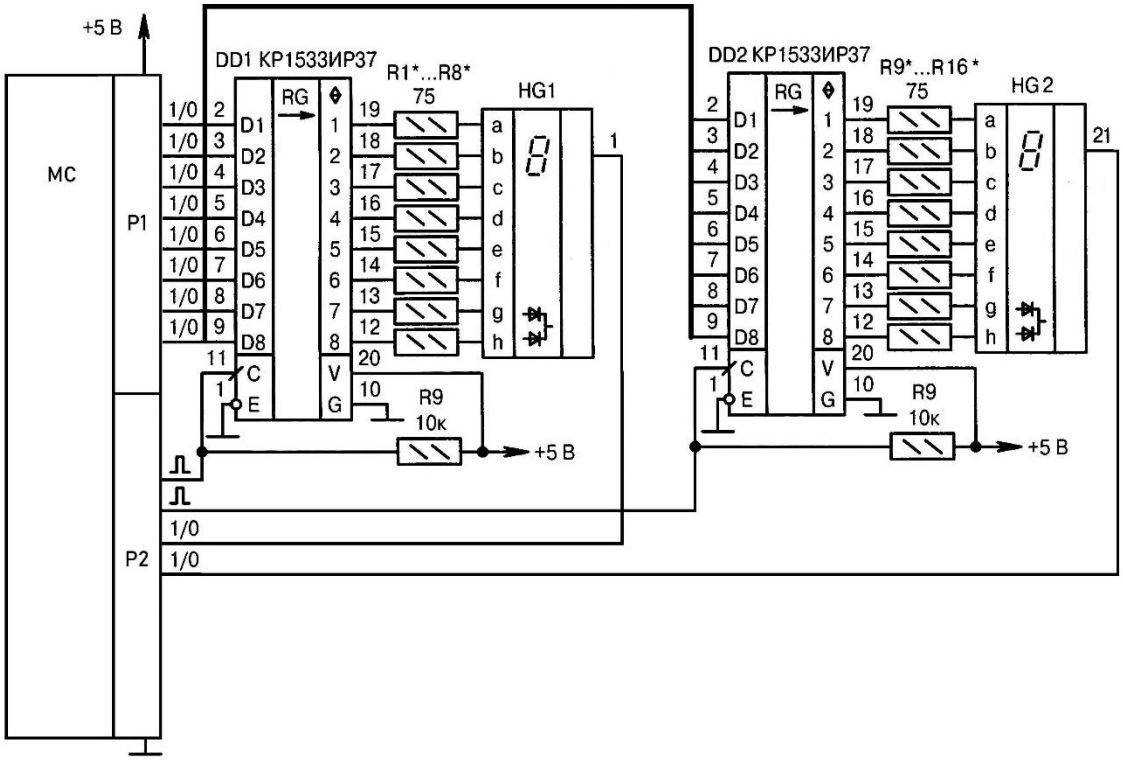

Рисунок 5.8 – Статическая индикация на регистрах КР1533ИР37

Регистры хранения DD1 и DD2 увеличивают допустимый ток через сегменты на индикаторах HG. Логика работы тут такая: выставляем на восьми линиях порта P1 код символа для индикатора HG1. На вход C индикатора подаем импульс длительностью 6...30 мс. Данные с порта P1 запоминаются в регистре, а после окончания импульса регистр становится недоступен для записи, но продолжает выдавать записанные данные на индикатор. Повторить процедуру заново, но уже для индикатора HG2. Резистор R9 важен в момент рестарта микропроцессора. Он принудительно переводит все входы регистров в Z-состояние, предотвращая случайные показания на индикаторах. Про вывод информации на индикаторы можно забыть до следующей процедуры вывода.

Сокращение числа задействованных линий микропроцессора можно реализовать с помощью сдвигового регистра 74HD4094 (или CD4094BE, или КР1561ПР1). На верхних выходах микропроцессора формируется «бегущая единица», но с микросекундными паузами в несколько микросекунд, чтобы успевал смениться код очередной цифры/символа на нижних линиях микропроцессора. Информация в сдвиговый регистр DD1 вводится по линии «D» последовательно во времени за 8 тактов, сообразуясь с сигналами на линиях «S» (строб), «C» (синхро). Например, можно использовать схему П. Гуссенса (см. рис. 9). Сдвиговые регистры DDl ... DDn соединяются последовательно через выходы переноса «P» и входы «D». Информация во все индикаторы HG1 ... HGn заносится последовательно во времени общим блоком «единиц/нулей». Преимущество этой схемы – статическая индикация, не создающая динамических

импульсных помех. Регистры DD1 ... DDn при желании можно заменить дешифраторами 74HC595.

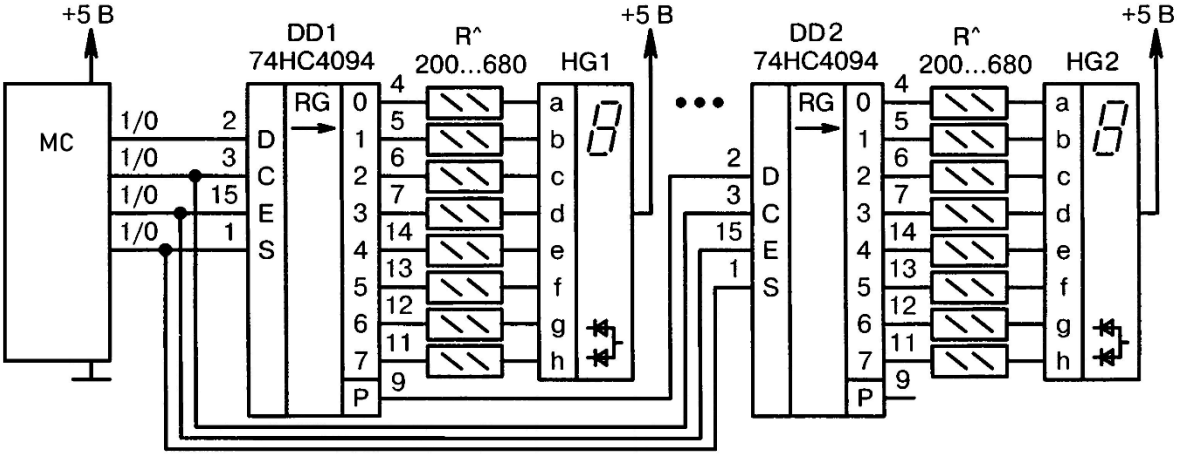

Рисунок 5.9 – Статическая индикация на сдвиговом регистре 74HD4094

Второй способ (*динамический*) основан на том, что любой световой индикатор является инерционным прибором, а человеческому глазу отображаемая на дисплее информация, если ее обновлять с частотой хотя бы 24 раз в секунду, представляется неизменяемой. Делаем очень просто: подключаем все индикаторы параллельно. Точнее, сажаем выводы сегментов на общую шину. А общие аноды подключаем раздельно к порту P1 (рис. 5.10).

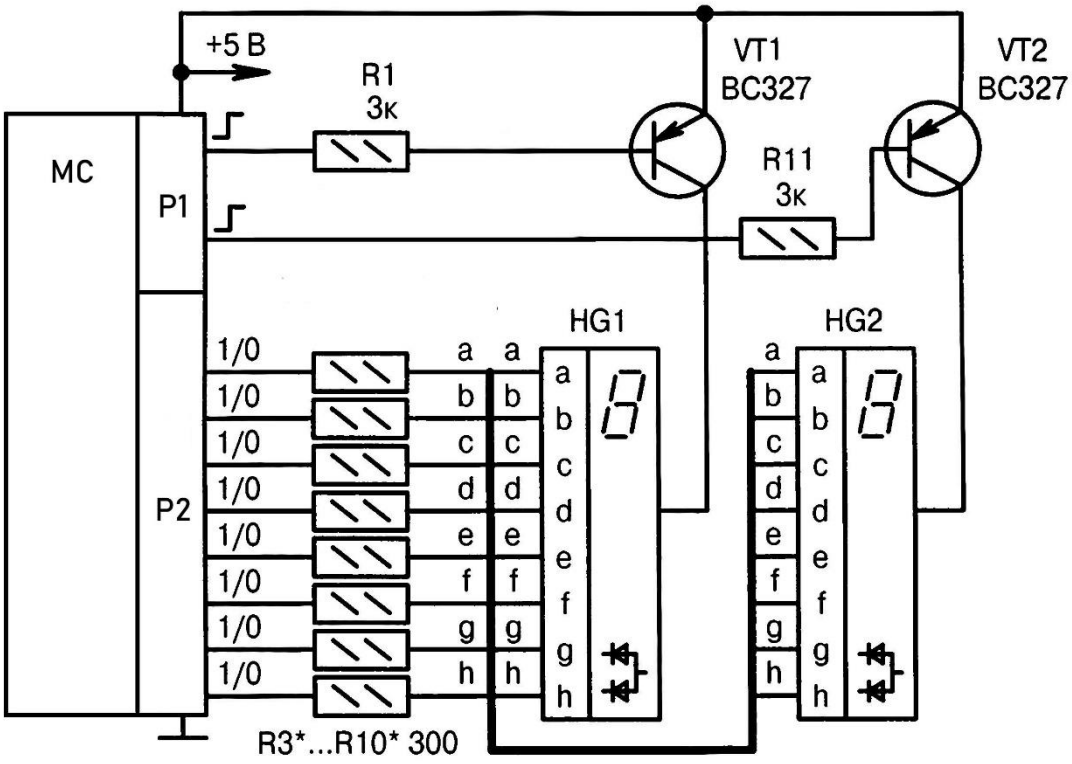

Рисунок 5.10 –Динамическая индикация

Что происходит? Мы последовательно подаем напряжение на адресные входы индикаторов, и одновременно выдаем в шину 7-сегментный код, соот-

ветствующий индикатору, активному в данный момент. Кроме этого, для повышения тока, индикаторы HG1 и HG2 управляются эмиттерными повторителями на транзисторах VT1 и VT2. Динамический способ вывода информации требует значительно меньших аппаратных затрат, но более сложного программного обеспечения. Необходимо, например, использовать подпрограмму задержки времени. И не только… Но именно этот способ организации вывода информации получил преимущественное распространение в микропроцессорных системах. Все. Прекращаю кормить Вас баснями. Сейчас вы пишете программу.

5.3 Задание

В системе управления имеются дискретных четыре датчика, подключенных к младшей тетраде порта P1, датчики могут находиться во включенном состоянии (выдавать сигнал логической «1») и в выключенном состоянии (выдавать сигнал логического «0»). К порту P2 подключены четыре 7-сегментных индикатора, управление которыми осуществляется по линиям P1.4 – P1.7. Разработать электрическую структурную схему. Составить программу, определяющую состояние датчиков, и отображающую порядковый номер датчика на соответствующем индикаторе, если датчик включен, или 0, если датчик выключен.

5.4 Примечание к выполнению задания

Если последняя цифра в номере зачетной книжки четная – индикация статическая, если нечетная – динамическая. Если предпоследняя цифра в номере зачетной книжки четная – использовать двоично-семисегментные дешифраторы, если нечетная – выполнить программное преобразование кодов. Если третья цифра в номере зачетной книжки четная – использовать индикаторы с общим катодом, если нечетная – с общим анодом.

5.5 Содержание отчета

1. Задание.
2. Схема электрическая структурная.
3. Графический алгоритм программы.
4. Текст программы на языке Ассемблер.
5. Выводы по работе.

5.6 Контрольные вопросы

1. Чем отличается порт BUS от портов P1 и P2?
2. Какие команды ввода-вывода существуют для портов P1 и P2?
3. Как в программе определить состояние дискретного датчика?
4. Объясните суть статического способа индикации.
5. Объясните суть динамического способа индикации.
6. Почему 7-сегментные индикаторы нежелательно подключать напрямую к портам микропроцессора?

6. ИССЛЕДОВАНИЕ СИСТЕМЫ ПРЕРЫВАНИЙ

6.1 Цель работы

Изучить систему прерываний микропроцессора Intel 8048, разработать подпрограмму обслуживания прерывания от внешнего источника запроса, исследовать механизм обслуживания прерывания от внешнего источника для однокристального микропроцессора Intel 8048.

6.2 Введение

Вот Вам пример из обыденной жизни – сидите Вы на кухне, чай пьете с малиновым вареньем и вкусняшками всенепременно, и ждете гостей. А как узнать, что кто-то пришел? Тут два варианта: либо мы каждые пять минут будем отвлекаться от варенья, в смысле, чая и бегать проверять, а не стоит ли кто за дверью, поскольку дверь у Вас под током – в них стучишь, они тебя бьют – шуму-грому на весь дом, кто же в такие стучать будет, либо купить дверной звонок и спокойненько ждать на нагретом месте, пока кто-нибудь в него не позвонит.

Так вот, когда гость звонит – это событие. Соответственно, мы прерываемся и кидаемся к двери. Естественно, возникает проблема приоритета. Это как мы все также сидим пьем чай, но звонят нам уже не только в дверь, но и по телефону... И ведь не разорвешься, что-то нужно сделать первым. Поэтому есть таблица векторов прерываний. Чем меньше номер прерывания, тем более оно приоритетно. Но есть несколько тонкостей...

Вот произошло событие – пошел запрос на прерывание, то есть выставляется так называемый «флаг запроса на прерывание». Если все хорошо, прерывание разрешено, то жизнь прекрасна и происходит его обработка.

А вот если прерывание запрещено – например, уже происходит обработка более приоритетного прерывания – тогда этот флаг запроса так и остается висеть до момента разрешения прерываний. После этого микропроцессор проверяет регистр запроса (если он есть в микропроцессоре) в порядке приоритета и, если флаг есть – обрабатывает его.

Однако… Получается, что даже если прерывание обрабатывается, не факт, что событие, которое его вызвало, ещё живо... Это как позвонили в дверь и по телефону одновременно, Вы ответили по телефону, а гости уже решили, что никого нет дома, и ушли. Вроде как событие – звонок в дверь – было, а за дверью никого нет.

Ещё одна проблема: пока обрабатывается другое прерывание и флаг запроса уже установлен, событие может произойти ещё несколько раз. Ответили на звонок по телефону, открываем дверь – а там уже целая куча гостей! Страшно? Страшно...

Ещё одна особенность использования прерываний – да и не только прерываний: реентерабельность (или повторная входимость). Реентерабельная программа – та, которую могут вызвать несколько пользователей (или процессов), и при этом как минимум не возникнет ошибка, а как максимум – не

будет потери вычислений – например, другому пользователю не придется выполнять код снова.

Иными словами, если на кухне, в то время, пока Вы встречаете гостей, никто не утащит себе вкусняшки, то значит все реентерабельно.

Вообще-то, серьёзная штука – если её не учитывать, можно долго мучится с вопросом: «А чего же она не работает?!». Нужно её учитывать, например, если обрабатываются несколько прерываний, и каждое изменяет какую-то глобальную переменную...

Обычно прерывания не реентерабельны. То есть, в момент работы прерывания нельзя повторно вызвать это же прерывание. Именно для защиты от повторного вхождения в обработчик прерывания автоматически запрещаются в момент его обработки (если вам захотелось разрешить прерывания в процедуре обработки прерываний, надо десять, двадцать раз подумать, прежде чем сделать такой опрометчивый шаг).

Встроенная в микропроцессор Intel 8048 одноуровневая система прерываний имеет два источника запросов, имеет фиксированные векторы прерываний и приоритеты. Одноуровневость системы прерываний означает, что никакие новые запросы не воспринимаются до тех пор, пока не будет обслужен текущий запрос.

Итак, в системе определены два источника запросов: по входу INT (стартовый адрес подпрограммы обслуживания 003H) и по переполнению таймера T (стартовый адрес подпрограммы обслуживания 007H). Каждый источник может быть замаскирован независимо от других специальными флажками EX и ET, управляемыми командами:

```
EN I        ; Разрешение внешнего прерывания INT
DIS I       ; Запрет внешнего прерывания INT
EN TCNTI    ; Разрешение прерывания от переполнения таймера
DIS TCNTI   ; Запрет прерывания от переполнения таймера
```

Флажки сбрасываются при подаче сигнала RESET. В случае одновременной фиксации двух запросов приоритет отдается внешнему. Структура системы прерываний от внешнего источника показана на рисунке 6.1. Внешнее прерывание по входу INT воспринимается, когда на нем появляется напряжение низкого уровня. Линия INT контролируется во время сигнала ALE в последнем машинном цикле каждой команды. Результат тестирования запоминается во внутреннем триггере внешнего прерывания IX, установка которого инициирует формирование аппаратного эквивалента команды CALL 003H. Обычно по адресу 003H находится команда безусловного перехода на подпрограмму обслуживания или команда вызова подпрограммы.

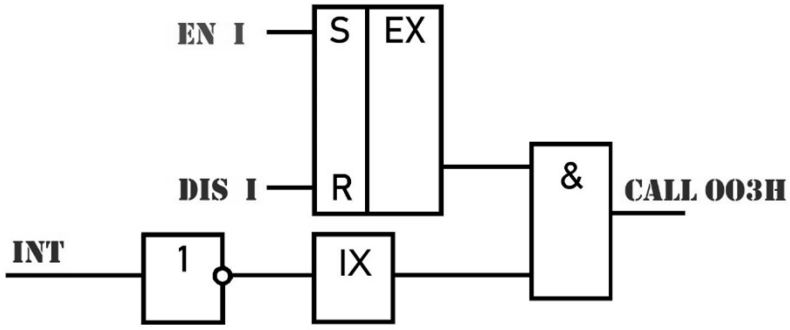

Рисунок 6.1 – Структурная схема системы прерываний

При выполнении аппаратного эквивалента команды CALL 003H текущее содержимое регистра PC и флажки PSW запоминаются в стеке, обеспечивая возможность возврата в точку прерывания.

Программа обслуживания прерывания должна завершаться командой RETR, восстанавливающей PC и PSW. В начале второго цикла команда RETR разрешает прием новых запросов на прерывание. К этому моменту сигнал запроса INT должен быть снят внешними средствами. Обычно это делается автоматически в момент обслуживания устройства, выставившего запрос. Если такая возможность не осуществляется, то любая выходная линия Intel 8048 может быть использована для выдачи специального сигнала подтверждения прерывания.

Если для хранения программы требуется объем памяти, превышающий 2 Кбайт, то при разработке программ обслуживания прерываний пользователю необходимо учитывать особенности управления памятью программ, присущие данному микропроцессору. Микропроцессор INTEL 8048может прямо адресовать только 2048 байт памяти программ. Для расширения объема адресуемой памяти до 4096 байт предусмотрен механизм переключения банков памяти программ с использованием команд SEL MB0, SEL MB1: (0...2047) байт – банк 0, (2048...4096) байт – банк 1. Однако, его нельзя использовать при обработке прерываний, так как при выполнении перехода или вызова подпрограммы в одиннадцать младших разрядов счётчика команд PC производится загрузка адреса из команды, 12-й разряд загружается из триггера переключения банка памяти MB, указывающего номер банка памяти программ.

В микропроцессоре при обслуживании прерываний старший разряд PC, независимо от состояния триггера MB, аппаратно устанавливается в «0» всякий раз, когда происходит переход к подпрограмме обработки прерываний. Поэтому все программы обработки прерываний и все подпрограммы, вызываемые ими, должны располагаться в пределах нулевого банка памяти программ (ячейки 0...2047). До выхода из подпрограммы обслуживания прерываний (то есть до исполнения команды RETR) никакая команда не может установить в «1» старший разряд PC. Выполнение команд SEL MB0 и SEL MB1во время обработки прерываний не рекомендуется, так как эти команды,

изменяя состояние триггера MB, не изменяют значения старшего разряда PC до тех пор, пока происходит обработка текущего прерывания.

6.3 Задание

Разработать программу для системы управления автоматизированного устройства, в состав которого входят восемь цилиндров. Воздух в цилиндры подается при помощи четырехлинейных двухпозиционных пневматических распределителей. При подаче напряжения высокого уровня на катушку распределителя происходит выдвижение штока цилиндра, при подаче напряжения низкого уровня – втягивание штока. Цилиндры должны срабатывать последовательно, после выдвижения штока первого цилиндра должен выдвинуться шток второго и т.д. После того, как сработает восьмой цилиндр, необходимо привести цилиндры в исходное состояние (втянуть штоки) в обратной последовательности (с 8 по 1 цилиндр). Время выдвижения и втягивания штока принять равным T. В случае возникновения аварийной ситуации (появлении сигнала низкого уровня на входе INT) следует приостановить работу устройства. Работа должна быть возобновлена после нажатия кнопки «Продолжить», сигнал от которой поступает на вход T0 микропроцессора. Убедитесь, что сигнал INT после устранения аварии снят.

6.4 Индивидуальное задание

Время T определяется по формуле:

$$T = i + j + 1 \ (мс),$$

где i, j — последние цифры в номере зачетной книжки. В процессе выполнения подпрограммы обслуживания зафиксировать содержимое стека.

6.5 Содержание отчета

1. Задание.
2. Текст программы на языке Ассемблер.
3. Содержимое стека микропроцессора.
4. Выводы по работе.

6.6 Контрольные вопросы

1. Нарисуйте структурную схему системы прерываний микропроцессора Intel 8048 и объясните принцип ее работы.
2. Что такое одноуровневая системы прерываний?
3. Какой из запросов на прерывание имеет высший приоритет?
4. Назовите начальные адреса подпрограмм обслуживания прерываний.
5. Какие команды используются для маскирования прерываний?
6. Каким образом можно использовать вход INT?

7. СИНТЕЗ И ИССЛЕДОВАНИЕ РАБОТЫ АЦП ДВОЙНОГО ИНТЕГРИРОВАНИЯ

7.1 Цель работы

Разработать принципиальную электрическую схему АЦП двойного интегрирования на основе однокристального микропроцессора Intel 8048, разработать программу управления АЦП, исследовать процесс получения информации с АЦП, используя эмулятор для разработки и отладки управляющей программы, исследовать работу встроенного таймера и возможности пошагового режима и точек прерываний эмулятора.

7.2 Введение

Подавляющее большинство датчиков, использующихся в технических системах, выдают сигнал в аналоговом виде. Считывать, да и вообще обрабатывать информацию лучше все-таки в цифровой форме, которая является универсальной и не зависит от природы физической величины. Встает задача преобразования аналоговой величины в дискретную. Если подумать, то термин «аналоговый» не слишком хорошо отражает сущность явления – точнее говорить «непрерывный». Интересно, что непосредственный цифровой способ отображения информации на том же 7-сегментном индикаторе, хотя и значительно более корректен, чем аналоговый, не всегда может оказаться более правильным. Датчик количества топлива в том же автомобиле, который проектировщики традиционно заставляют врать, занижая показания, иначе слишком много водителей оказывалось бы в 100 метрах от ближайшей заправочной станции, является аналоговым. А обладателю часов с цифровым дисплеем приходится в уме производить довольно сложные арифметические действия, вычисляя интервал времени, вместо того чтобы просто мысленно передвинуть стрелки на циферблате. Как видите, проектирование устройств отображения не слишком простое дело, и не может производиться только из соображений компактности схемы или высокой точности измерений, а непременно должно учитывать требования эргономики и удобства пользования.

Основной принцип оцифровки любых сигналов очень прост. В некоторые моменты времени t_1, t_2, t_3 и т.д. мы берем мгновенное значение аналогового сигнала и как бы прикладываем к нему линейку, проградуированную в двоичном масштабе. Обычная линейка содержит крупные деления (метры), поделенные каждое на десять частей (дециметры), каждая из которых также поделена на десять частей (сантиметры), и ещё на десять частей (миллиметры). Двоичная линейка содержала бы деления, поделенные пополам, затем еще раз пополам и т. д. – на сколько хватит разрешающей способности. Если вся длина такой линейки составляет, к примеру, 2,56 м, а самое мелкое деление – 1 см (то есть мы можем померить ей длину с точностью не хуже 1 см), то таких делений будет ровно 256, и их можно представить двоичным числом размером 1 байт или 8 двоичных разрядов. Ничего не изменится, если мы меряем не длину, а напряжение. При выбранной разрешающей способности и числе разрядов мы можем измерить величину не больше некоторого значения, которое

соответствует максимальному числу, в данном случае 255. Иначе придется или увеличивать число разрядов (удлинять линейку) или менять разрешающую способность в сторону ухудшения (растягивать ее). Все изложенное здесь и есть сущность работы аналого-цифрового преобразователя – АЦП.

Нет в жизни совершенства. Наиболее точными и одновременно самыми медленными являются интегрирующие АЦП. Их следует рассмотреть подробно, потому что: во-первых, они достаточно просты схемотехнически, (иногда даже есть смысл самому соорудить такой узел схемы на дискретных элементах, чем подбирать подходящую БИС), и во-вторых, этот тип АЦП часто применяется в радиолюбительской. Интегрирующих АЦП вообще-то не меньше десятка, но мы рассмотрим наиболее простые.

Схема самого простого интегрирующего АЦП показана на рис. 7.1. Это АЦП с однократным интегрированием. В начале преобразования на вход С динамического D-триггера поступает положительный фронт, который устанавливает выход Q в состояние логической единицы. Она является разрешающим уровнем для элемента «И–НЕ», и на вход счетчика поступают импульсы. Одновременно через выход $\overline{Q}$ запирается транзистор VT1. Конденсатор начинает заряжаться от источника стабильного тока. При равенстве значения входного измеряемого напряжения и напряжения на конденсаторе компаратор срабатывает и обнуляет триггер (выход на логическом элементе «И–НЕ» запираются, транзистор открывается и разряжает конденсатор, счетчик обнуляется). Количество импульсов, накопленных в счетчике к этому моменту, пропорционально входному напряжению.

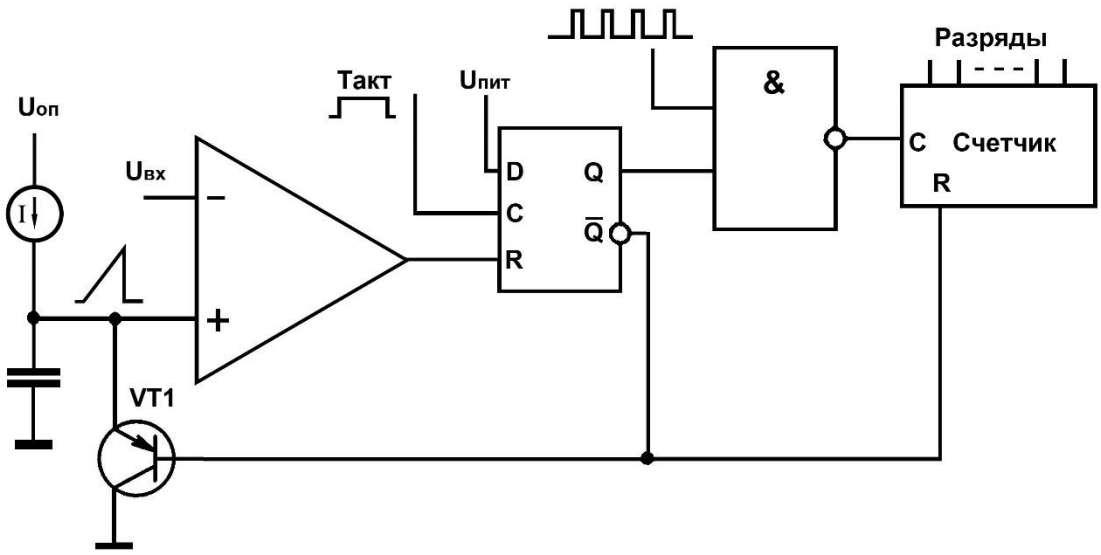

Рисунок 7.1 – АЦП однократного интегрирования

У схемы на рис. 7.1. единственное достоинство – простота, и целая куча недостатков. Главным ее недостатком является то, что результат преобразования тут зависит от всего на свете – от стабильности источника тока и генератора линейного напряжения (и каждого его элемента в отдельности, в первую

очередь – конденсатора), от стабильности порога компаратора, от неидеальности ключа для сброса и т. п. Еще хуже то, что схема в данном варианте срабатывает от мгновенного значения входного сигнала и потому весьма восприимчива к его дребезгу и вообще любым помехам. А если тактовая частота случайно окажется кратной частоте помехи (в первую очередь сетевой с частотой 50 Гц), то мы вообще можем получать каждый раз значения, весьма далекие от истины. Поэтому такая схема годится лишь для измерения сигналов постоянного тока – для контроля напряжения батареек или чего-нибудь в этом роде.

В то же время преобразование длится все равно достаточно долго, так как обычные значения тактовой частоты, при которых схема еще работает приемлемо, лежат в диапазоне максимум десятков килогерц, то есть для достижения разрешающей способности в восемь разрядов (больше все равно не выжать) частота отсчетов составит в лучшем случае 100 Гц, на практике же еще меньше. Может быть, использовать этот факт и измерять не мгновенное, а среднее значение сигнала за время преобразования? Сделать это несложно: подаём измеряемое напряжение на вход генератора линейного напряжения, а опорное – на компаратор. Тогда сигнал будет интегрироваться за время преобразования, причем интегрироваться очень точно, мы будем получать истинное среднее арифметическое значение сигнала за это время. Но легко увидеть, что сама функция преобразования при этом окажется обратной – то есть время заряда (и значение выходного кода на счетчике) окажется обратно пропорциональным значению входного напряжения. Это неудобно, так как сильно усложняет обработку результата.

Можно использовать какой-нибудь хитрый метод деления частоты с использованием реверсивного счетчика, можно попробовать инвертировать входной сигнал и затем сдвинуть его в положительную область, но все это приводит к усложнению схемы, причем неоправданному – сама по себе точность преобразования в любом случае не увеличится, избавляемся мы только от помехи. По всем этим причинам АЦП с однократным интегрированием, несмотря на его простоту, в настоящее время не употребляют вообще и даже не выпускают в виде специализированных микросхем. Единственная область, где можно было бы рекомендовать использовать такой метод – использование микропроцессоров, имеющих встроенный компаратор. В этом случае с помощью одного внешнего резистора и конденсатора можно получить простейший преобразователь аналогового сигнала в код. Но и эта рекомендация потеряла в настоящее время всякий смысл, так как доступны микроконтроллеры со встроенными «нормальными» АЦП, без всяких внешних элементов, причем мультиканальными, с гарантированной точностью, и разрешением до 10-ти и даже 12-ти разрядов, чего для большинства практических нужд более чем достаточно.

Однако можно преодолеть чуть ли не все перечисленные выше недостатки одним махом, и притом не слишком усложняя схему. Интегрирующие

АЦП не получили бы такого распространения и заслуженной репутации «самых стабильных», если бы не это обстоятельство. Идея метода, который называется двойным интегрированием, показана на рис. 7.2.

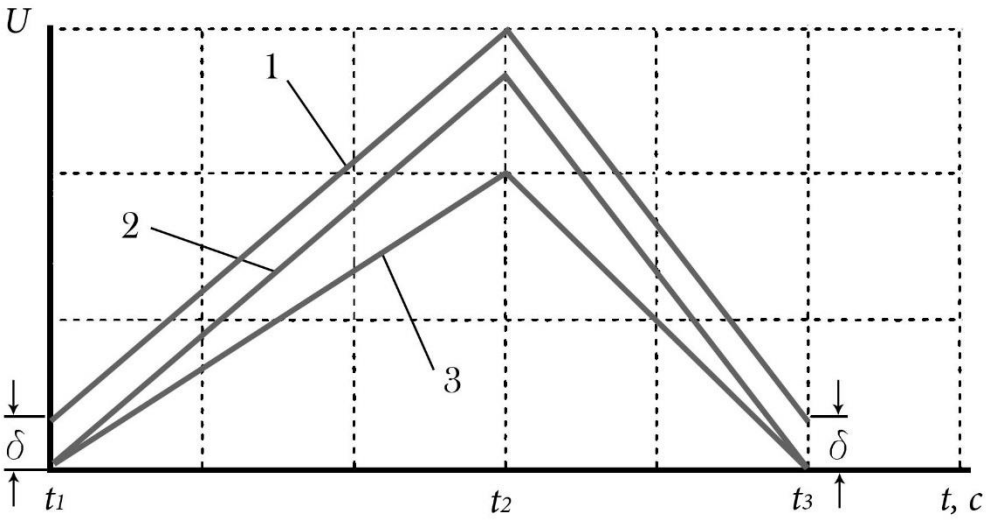

Рисунок 7.2 – Цикл работы АЦП двойного интегрирования
(1 – при сдвиге порога компаратора; 2 – идеальный случай; 3 – при изменении емкости конденсатора)

Посмотрим сначала на график, обозначенный цифрой 2. В первую стадию цикла работы за фиксированное время такта $t_2 - t_1$ конденсатор интегратора заряжается током, который определяется входным (измеряемым) напряжением $U_{вх}$. Во второй стадии этот конденсатор разряжается точно известным током, определяющимся опорным напряжением $U_{оп}$, до момента равенства напряжения нулю (t_3). Чем больше входное напряжение, тем до большей величины зарядится конденсатор в первой стадии, и тем дольше он будет разряжаться во второй. Легко показать, что отношение интервала времени $t_3 - t_2$ к известному времени такта $t_2 - t_1$ будет равно отношению входного напряжения $U_{вх}$ к опорному $U_{оп}$. Таким образом, измерив полученный интервал времени $t_3 - t_2$ обычным методом с помощью счетчика, как это сделано в схеме на рис. 7.1, мы получим на выходе код, пропорциональный входному напряжению.

На самом деле напряжение, до которого разряжается конденсатор, задается порогом компаратора и может в общем случае быть отличным от нуля на величину δ за счет отклонения порога, например, при изменении температуры. Но так как в начале цикла измерения напряжение определялось тем же значением порога, то, как вы видите из графика 1 на рис. 7.2, в данном случае имеет значение только изменение порога за время преобразования. А оно даже в самых неповоротливых АЦП такого типа не превышает долей секунды, потому это изменение можно не принимать в расчет. На результате не скажется и изменение емкости конденсатора, так как при этом наклон прямой и заряда, и разряда изменится в одинаковой степени (график 3).

В самых точных АЦП такого типа дополнительно проводят цикл «коррекция нуля», когда на вход подают нулевое напряжение и результат потом вычитают из значения кода, полученного в рабочем цикле. Мало того, всю схему можно заводить от любого RC-генератора при условии, что время такта $t_3 - t_2$ и частота заполнения «ворот» для подсчета длительности результирующего интервала $t_2 - t_1$ задаются от одного и того же генератора. Но чудес не бывает – точность и стабильность преобразования здесь полностью определяются точностью и стабильностью значения U_{on}. От этого никуда не денешься, это общее условие для всех без исключения конструкций АЦП и ЦАП.

Сконструированное нами АЦП относится к типу ПНВ: преобразователей напряжение – время. Ранее широко использовались ПНЧ: преобразователи напряжение – частота (в основном на основе микросхемы 555), однако большинство их реализаций обладает тем же недостатком, что и однократный интегратор, то есть в них точность напрямую зависит от качества компонентов.

Схема, реализующая метод двойного интегрирования, представлена на рисунке 7.3.

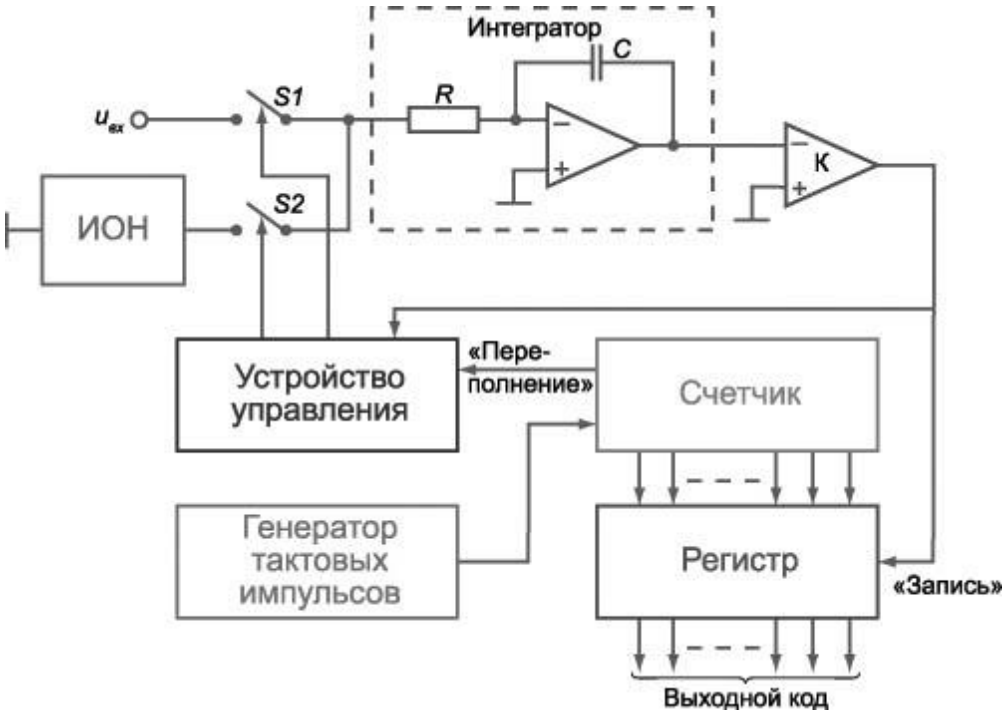

Рисунок 7.3 – Структурная схема АЦП двойного интегрирования

При наличии микропроцессора все функциональные блоки, за исключением интегратора, компаратора и электронных ключей можно реализовать внутренними средствами этого микропроцессора, разработав соответствующую программу, которую Вы и должны реализовать.

7.3 Задание

Разработать структурную схему АЦП двойного интегрирования, построенную на базе микропроцессора Intel 8048, и описать алгоритм ее работы. Считаем, что реализации переключателей S1 – S2 использованы транзисторные

ключи, управление которыми осуществляется по линиям P1.0 и P1.1 соответственно. Выход компаратора подключен ко входу T0 микропроцессора.

Разработать и отладить программу управления АЦП. Пользуясь эмулятором, исследовать работу АЦП, получить периоды T_2 и T_3, записать их значения в регистры R6 и R7, по значению опорного напряжения U_{on} определить значение входного напряжения. Для заинтересованных студентов – предложить алгоритм вычисления входного напряжения.

7.4. Индивидуальное задание

Следует заметить, что эмулятор не позволяет моделировать работу подключённых внешних устройств. При срабатывании ключа S2 (за счет установки P1.1 в единицу) запускаем таймер. Срабатывание компаратора эмулируем изменением состояния входа T0. При этом необходимо остановить таймер, занести полученное значение $t_2 - t_1$ в регистр R6, очистить содержимое таймера, и после отключения S2 и включения S1 снова запустить таймер. При новом изменении состояния на входе T0 остановить таймер, занести полученное значение $t_3 - t_2$ в регистр R7. Для расчета входного напряжения опорное напряжение определяются из таблицы 7.1.

Таблица 7.1 – Значение опорного напряжения

i	0	1	2	3	4	5	6	7	8	9
U_{on} , B	3	4	5	6	7	8	9	10	12	15

где i – последняя цифра номера зачетной книжки.

7.4 Содержание отчета.

1. Цель работы.
2. Структурная схема АЦП на основе микропроцессора.
3. Описание работы АЦП.
4. Текст программы на языке Ассемблер.
5. Выводы по работе.

7.5 Контрольные вопросы.

1. Нарисуйте формат слова состояния программы и расскажите о флагах, устанавливаемых по результатам выполнения операций обработки данных в АЛУ.
2. Как измерить период времени между двумя событиями?
3. Как организовать задержку времени для микропроцессора Intel 8048?
4. Опишите работу системы прерываний микропроцессора Intel 8048.
5. Какие команды работы с таймером существуют?

8. СИНТЕЗ И АНАЛИЗ ШИРОТНО-ИМПУЛЬСНОГО РЕГУЛЯТОРА

8.1 Цель работы

Разработать программу для микропроцессора, реализующую широтно-импульсное управление двигателем постоянного тока. Измерить частоту широтно-импульсной модуляции.

8.2 Теоретический раздел

Как Вы наверняка помните, в цифровой электронике все сигналы могут быть представлены всего двумя уровнями напряжения: «логической единицей» и «логическим нулём». Это всего лишь условные величины напряжения. «Логической единице» приписывается высокий уровень напряжения, обычно около 3 – 5 вольт, «логическим нулём» считается близкое к нулю напряжение. Цифровые импульсы графически изображаются прямоугольными или трапециевидными по форме.

Главной величиной одиночного импульса является его длительность. Длина импульса – это отрезок времени, в течение которого рассматриваемый логический уровень имеет одно устойчивое состояние. На рисунке латинской буквой *t* отмечена длина импульса высокого уровня, то есть логической «1». Длина импульса измеряется в секундах, но чаще в миллисекундах (мс), микросекундах (мкс) и даже наносекундах (нс). А одна наносекунда – это очень короткий отрезок времени!

Чем больше импульсов, тем больше информации можно ими передать. У множества импульсов появляется много характеристик. Самая простая – частота следования импульсов. Частота следования импульсов – это количество полных импульсов в единицу времени.

Существуют ещё несколько характеристик, которые проявляются только при участии двух и более импульсов. Одним из таких важных параметров импульсной последовательности является период.

Период импульсов – это промежуток времени, между двумя характерными точками двух соседних импульсов. Обычно период измеряют между двух фронтов или двух спадов соседних импульсов и обозначают заглавной латинской буквой *T*.

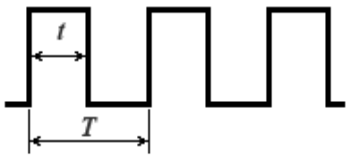

Рисунок 8.1 – Последовательность импульсов

Период следования импульсов напрямую связан с частотой импульсной последовательности, и его можно вычислить по формуле:

$$T = 1/F$$

Если длина импульса *t* точно равна половине периода *T*, то такой сигнал часто называют «*меандр*».

Скважностью импульсов называется отношение периода следования импульсов к их длительности и обозначается буквой *S*:

$$S = T/t$$

Скважность – безразмерная величина, не имеет единиц измерения, но может быть выражена в процентах. Часто в англоязычных текстах встречается термин «*Duty cycle»*, это так называемый коэффициент заполнения. Коэффициент заполнения *D* является величиной, обратной скважности. Коэффициент заполнения обычно выражается в процентах и вычисляется по формуле:

$$D = 1/S$$

Если Вы внимательно читали, то могли сообразить, что если увеличивать или уменьшать длину импульса и при этом на столько же уменьшать или увеличивать паузу между импульсами, то период следования импульсов и частота останется неизменной!

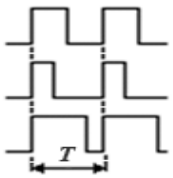

Рисунок 8.2 – Импульсы с различной сважностью

Это очень важный факт, который нам ещё не раз понадобится в будущем. Но сейчас ещё хочется добавить другие способы передачи информации с помощью импульсов. Например, можно несколько импульсов объединить в группы. Такие группы с паузами определённой длины между ними называют пачками или пакетами. Генерируя разное число импульсов в группе и варьируя его, можно также передавать какую-либо информацию.

Рисунок 8.3 – Пакет импульсов

Для передачи информации в цифровой электронике (ещё её называют дискретной электроникой) можно использовать два и более проводников или каналов с разными импульсными сигналами. При этом информация передаётся с учётом определённых правил. Такой метод позволяет заметно увеличить скорость передачи информации или добавляет возможность управлением потоком информации между различными схемами.

Перечисленные возможности передачи информации с помощью импульсов могут быть использованы как сами по себе раздельно, так и в комбинации между собой. Существуют также множество стандартов передачи информации с помощью импульсов, например, I2C, SPI, CAN, USB, LPT.

Именно этот эффект положен в основу принципа ***Широтно-Импульсной Модуляции (ШИМ)***, английское сокращение – ***PWM (Pulse-Width Modulation)***.

Широтно-импульсная модуляция – это способ кодирования аналогового сигнала путём изменения ширины (длительности) прямоугольных импульсов несущей частоты.

На рис. 4 представлены типичные графики ШИМ сигнала. Так как при ШИМ частота импульсов, а значит, и период *T*, остаются неизменными, то при уменьшении ширины импульса *t* увеличивается пауза между импульсами (рис. 4, б) и наоборот: при расширении импульса пауза сужается (рис. 4, в).

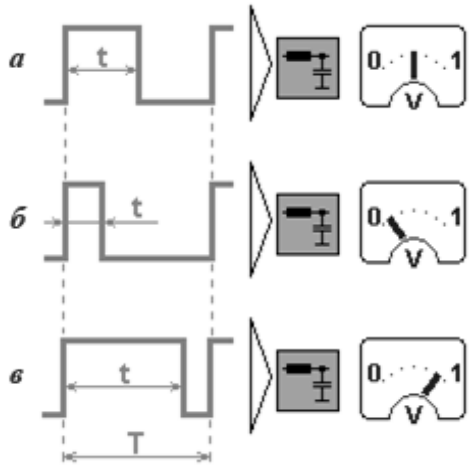

Рисунок 8.4 – Зависимость напряжения от скважности ШИМ

Если сигнал ШИМ пропустить через фильтр низких частот, то уровень постоянного напряжения на выходе фильтра будет определяться скважностью импульсов ШИМ. Назначение фильтра – не пропускать несущую частоту ШИМ. Сам фильтр может состоять из простейшей интегрирующей RC-цепи, или же может отсутствовать вовсе, если оконечная нагрузка имеет достаточную инерцию.

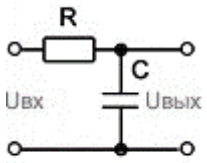

Рисунок 8.5 – Фильтр низких частот

Таким образом, имея в расположении лишь два логических уровня, «единицу» и «ноль», можно получить любое промежуточное значение аналогового сигнала. Часто в схемах с ШИМ применяют обратную связь для управления длительностью импульса по той или иной закономерности, например, в схемах PID-регуляторов.

Во многих современных микроконтроллерах имеются встроенные средства формирования одного или более независимых каналов ШИМ. Очень гибкие возможности конфигурации ШИМ-формирователя микроконтроллера позволяют использовать их в разнообразных схемах электронного управления и автоматики.

Пожалуй, наиболее часто ШИМ применяется для управления различными типами моторов. Тут основным достоинством ШИМ-управления является то,

что электронный ключ (обычно транзистор) работает в ключевом режиме, что заметно повышает экономичность схемы, так как потери на активных элементах сводятся к минимуму. Кроме того, широтно-импульсная модуляция применяется для формирования аналоговых сигналов сложной формы, в импульсных стабилизаторах, для кодирования и помехозащищённой передачи аналоговой информации.

Таким образом, принцип ШИМ заключается в изменении ширины импульса при постоянстве частоты следования импульса. Амплитуда импульсов при этом неизменна.

Аналоговая широтно-импульсная модуляция

Как уже упоминалось выше, частота сигнала и его амплитуда при ШИМ всегда постоянны. Одним из важнейших параметров сигнала ШИМ является коэффициент заполнения. Так, если мы имеем сигнал ШИМ с длительностью импульса 300 мкс и периодом импульса 1 мс, коэффициент заполнения составит 300/1000 = 0,3. Коэффициент заполнения также выражается в процентах, для чего коэффициент заполнения умножается на 100%. По примеру выше процентный коэффициент заполнения составляет 0,3 х 100% = 30%.

Смысл ШИМ заключается в регулировании среднего значения напряжения путем изменения коэффициента заполнения. Среднее значение напряжения равно произведению коэффициента заполнения и амплитуды напряжения. Так, при коэффициенте заполнения 0,3 и амплитуде напряжения 12 В среднее значение напряжения составит 0,3 х 12 = 3,6 (В). При изменении коэффициента заполнения в теоретически возможных пределах от 0% до 100% напряжение будет изменяться от 0 до 12 В, т.е. широтно-импульсная модуляция позволяет регулировать напряжение в пределах от 0 до амплитуды сигнала. Что и используется для регулирования скорости вращения электродвигателя постоянного тока или яркости свечения светильника.

Сигнал ШИМ формируется микроконтроллером или аналоговой схемой. Этот сигнал обычно управляет мощной нагрузкой, подключаемой к источнику питания через ключевую схему на биполярном или полевом транзисторе. В ключевом режиме полупроводниковый прибор либо разомкнут, либо замкнут, промежуточное состояние исключается. В обоих случаях тепловая мощность, рассеиваемая на ключе, ничтожна. Поскольку эта мощность равна произведению тока через ключ на падение напряжения на нем, а в первом случае ток, во втором – напряжение через ключ близко к нулю.

В переходных состояниях на ключе присутствует значительное напряжение с прохождением значительного тока. Поэтому в качестве ключа необходимо применение малоинерционных полупроводниковых приборов с быстрым временем переключения, порядка десятков наносекунд.

Если ключевая схема управляет светодиодом, то при малой частоте сигнала светодиод будет мигать в такт с изменением напряжения сигнала ШИМ. При частоте сигнала выше 50 Гц мигания сливаются вследствие инерции человеческого зрения. Общая яркость свечения светодиода начинает зависеть от

коэффициента заполнения – чем ниже коэффициент заполнения, тем слабее светится светодиод.

При управлении посредством ШИМ скорости вращения двигателя постоянного тока частота ШИМ должна быть очень высокой, и лежать за пределами слышимых звуковых частот, т.е. превышать 20 кГц, в противном случае двигатель будет «звучать», издавая раздражающий слух писк с частотой ШИМ. От частоты зависит и стабильность работы двигателя. Низкочастотный сигнал ШИМ с невысоким коэффициентом заполнения приведет к нестабильной работе двигателя и даже возможной его остановке.

Следовательно, при управлении двигателем желательно повышать частоту сигнала ШИМ, но и здесь существует предел, определяемый инерционными свойствами полупроводникового ключа. Если ключ будет переключаться с запаздываниями, схема управления начнет работать с ошибками. Чтобы избежать потерь энергии и добиться высокого КПД импульсного преобразователя, полупроводниковый ключ должен обладать высоким быстродействием и низким сопротивлением проводимости.

Сигнал с выхода ШИМ можно также усреднять посредством простейшего фильтра низких частот. Иногда можно обойтись и без этого, поскольку электродвигатель обладает определенной электрической индуктивностью и механической инерцией. Сглаживание сигналов ШИМ происходит естественным путем в том случае, когда частота ШИМ превосходит время реакции регулируемого устройства.

Цифровая широтно-импульсная модуляция

Существует разновидность ШИМ, называемая цифровой ШИМ. В этом случае период модулирующего сигнала заполняется прямоугольными импульсами несущей, и регулируется уже количество импульсов несущей в периоде, что и определяет среднюю величину сигнала за период (рис. 6).

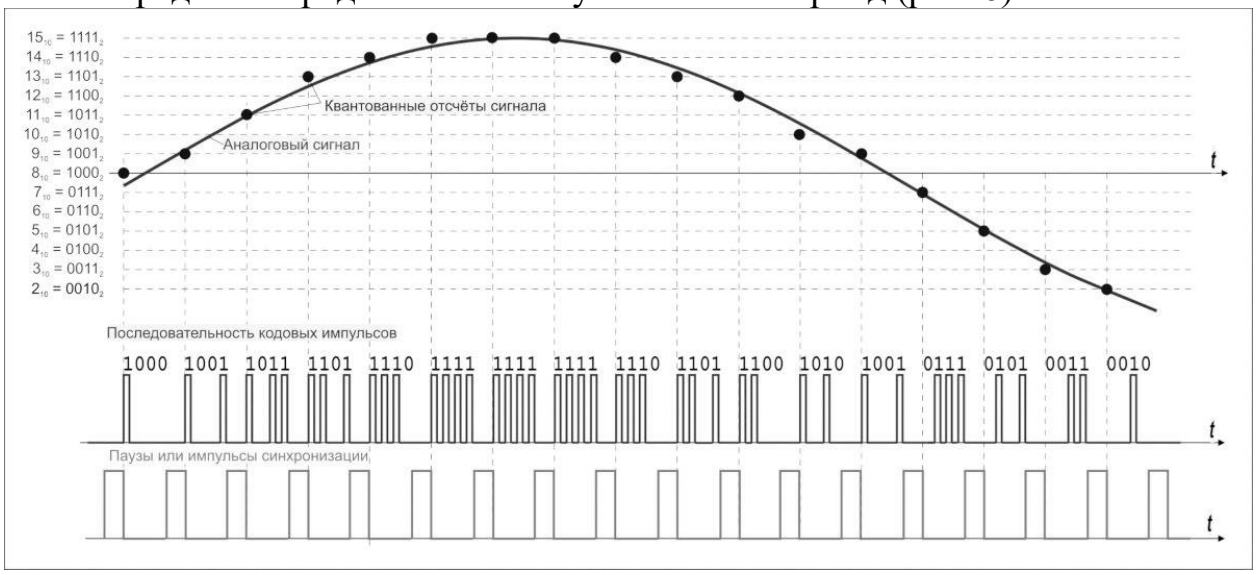

Рисунок 8.6 – Цифровая ШИМ

В цифровой ШИМ заполняющие период импульсы несущей («единицы») могут стоять в любом месте периода. Среднее значение напряжения за период

определяется только их количеством, при этом импульсы могут следовать один за другим и сливаться. Отдельно стоящие импульсы приводят к ужесточению режима работы ключа.

В качестве источника сигнала цифровой ШИМ можно использовать COM-порт компьютера с 10-битовым сигналом на выходе. С учетом 8 информационных битов и 2 битов старт/стоп, в сигнале COM-порта присутствует от 1 до 8 «единичек», что позволяет регулировать напряжение в пределах от 0 до 100% напряжения питания с шагом в 12,5%.

А теперь немного теории, истории и приложений в других областях.

Теоретические основы цифровой передачи звука были разработаны еще на заре двадцатого века, когда ученые попытались передать звуковой сигнал на большое расстояние, но не по телефону, а довольно странным для того времени способом.

Разделив звуковую волну на небольшие части, ее можно было отправлять получателю в некоем математическом представлении. Получатель, в свою очередь, мог восстановить исходную волну и прослушать запись. Также перед учеными стояла задача увеличить пропускную способность «эфира».

В 1933 году увидела свет теорема В.А. Котельникова. В западных источниках ее называют теоремой Найквиста – Шеннона. Да, Гарри Найквист был первым, кто затронул эту тему: в 1927 году он рассчитал минимальную частоту дискретизации для передачи формы волны, впоследствии названную в его честь «частотой Найквиста», – но мы–то знаем, что теорема Котельникова была издана на 16 лет раньше. В работе «О пропускной способности эфира и проволоки в электросвязи», в частности, была сформулирована одна из теорем следующим образом: *«Любую функцию F(t), состоящую из частот от 0 до* f_c*, можно непрерывно передавать с любой точностью при помощи чисел, следующих друг за другом через* $1/2f_c$ *секунд»*. Независимо от него эту теорему в 1949 году (через эти самые 16 лет) доказал Клод Шеннон, поэтому в западной литературе эту теорему часто называют теоремой Шеннона.

Кстати, в 1999 году Международный научный фонд Эдуарда Рейна (Германия) признал приоритет Котельникова, наградив его премией в номинации «За фундаментальные исследования» за «впервые математически точно сформулированную и доказанную в аспекте коммуникационных технологий теорему отсчётов».

Суть теоремы проста: непрерывный сигнал можно представить в виде интерполяционного ряда, состоящего из дискретных отчетов, по которым можно заново восстановить сигнал. Чтобы была возможность восстановить приблизительно исходное состояние сигнала, частота дискретизации должна равняться как минимум удвоенной верхней граничной частоте этого сигнала.

Много лет теорема не была востребована – вплоть до прихода цифровой эпохи. Тут-то ей и нашлось применение. В частности, теорема пригодилась при разработке формата CDDA (*Compact Disc Digital Audio*), в простонародье его называют Audio CD или Red Book. Формат был выпущен инженерами

Philips и Sony в 1980 году и стал стандартом для аудио-компакт-дисков. Характеристики формата: частота дискретизации – 44,1 кГц, разрядность квантования – 16 бит.

Частота дискретизации 44,1 кГц была рассчитана из теоремы Котельникова. Считается, что слух среднестатистического человека не способен уловить звук за пределами 19–22 кГц. Вероятно, частота 22 кГц и была выбрана в качестве верхней граничной. Считали так:

$$22\,000 \times 2 = 44\,000 + 100 = 44\,100 \text{ (Гц)}$$

Откуда взялось 100 Герц? Есть версия, что это небольшой запас на случай ошибок или передискретизации. На самом же деле такую частоту в Sony выбрали из соображений совместимости со стандартом телевещания PAL.

Модуляция плотности импульсов (PDM – Pulse Density Modulation)

После разработки формата CDDA Philips и Sony заключили инженерный союз для создания нового формата хранения цифрового аудио, который обеспечил бы существенно меньшее количество шумов и искажений в записи, нежели исходные студийные ленты или тот же CDDA. Проще говоря, соотношение SNR нового формата должно было быть не ниже 110 дБ.

Следует учесть, что на момент разработки формата разрядность квантования выше 16 бит и частоты семплирования мультибитного аудиопотока выше 44,1 КГц были, мягко говоря, с технической точки зрения, научной фантастикой. Создание «железа», способного на такое, в реалиях технологий тех лет, было абсолютно экономически не целесообразно.

Потому было решено, как завещал классик, пойти другим путём. И сделать формат… 1-битным! Таким образом, кодирование сигнала осуществлялось теперь не за счет его разрядности, а за счет импульсов высокой частоты. То есть был применен метод кодирования импульсно–плотностной модуляции.

Звучит, как абракадабра? На самом деле, все довольно просто. Такой метод формирования сигнала (модуляция) записывает повышение амплитуды единицами. А понижение – нулями. А нулевая точка (то есть тишина) кодируется сменой числа. А значение амплитуды сигнала (то есть частота звучания) в каждый момент времени представлено плотностью значений одного типа. То есть рядами нулей и единиц соответственно. Именно этот принцип и отражен в названии метода.

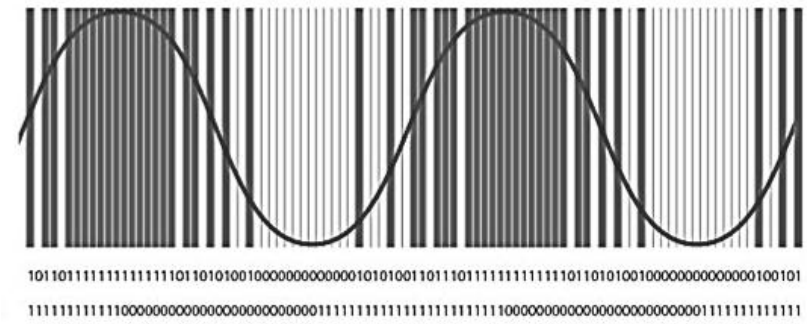

Рисунок 8.7 – Наглядный пример кодирования двухканального аудио

Частота дискретизации, учитывая назначение формата и его разрядность, должна быть очень высокой. Таковой она и является. Отправной точкой является частота дискретизации в 2,8 МГц! Что в 64 раза больше, чем частота у CD, равная 44,1 КГц. Казалось бы, это дичайший бред! Ведь многократное превышение частоты верхнего порога слышимых частот, то есть 20 кГц, не дает ничего для качества звучания. Ведь никакой полезной акустической информации оно добавить в запись уже не может…

Но это так только на первый взгляд! Этот подход называется oversampling, он же передискретизация, и призван он существенно уменьшить количество шумов, вносимых в запись при квантовании (изначальной дискретизации). Это достигается за счет того, что большая частота заставляет любые два соседних участка сигнала коррелировать друг с другом. То есть обеспечивает их устойчивую взаимосвязь, не допуская бесконтрольного изменения одного участка записи без соответствующего изменения соседнего. Что же получается в итоге? Однобитный формат, обладающий очень ценными свойствами: простотой декодирующего оборудования (ведь обработать одноразрядный сигнал, пусть и с высокой частотой, значительно проще и дешевле, нежели 16-разрядный); помехоустойчивостью; практически полным отсутствием шумов в слышимом акустическом диапазоне.

После всего вышеизложенного может возникнуть вопрос: чем хорош данный метод для регулирования мощности двигателя? Ведь каких-то принципиальных отличий от ШИМ здесь нет…

Можно назвать два преимущества: отсутствие шума при невозможности увеличить частоту переключения и более стабильную работу двигателя при малых мощностях. Но при этом надо реализовать специальные алгоритмы.

Простейший алгоритм – использование шаблонов. Идею этих алгоритмов легко понять из рис. 8.8.

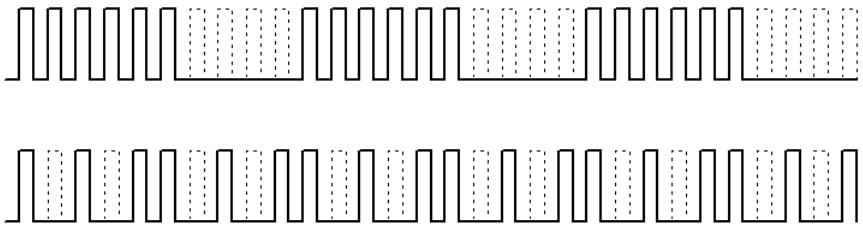

Рисунок 8.8 – Простейший алгоритм реализации PDM

Шаблон задается на некотором количестве тактов (кадр) внешнего генератора меандра. Количество ненулевых импульсов, отнесенное к количеству тактов на кадре, задает относительную плотность импульсов. Количество тактов в кадре (длина кадра) определяет разрешение, с которым задается плотность импульсов (и, соответственно, мощность).

Например, на кадре длиной 10 тактов можно задать 11 уровней мощности. Именно такой случай показан на рис.8.8.

Самый простейший шаблон, который можно задать на кадре, представлен на рис.8 сверху. В первой части кадра модулятор пропускает подряд какую-то часть импульсов (в зависимости от требуемой мощности), а остальные (до конца кадра) не пропускает. На рисунке показано три кадра с шаблоном, соответствующим мощности 60%.

Этот простейший шаблон легко реализовать на практике для любой длины кадра (чем длиннее кадр, тем меньше шаг регулирования мощности).

Но у него есть один серьезный недостаток, который, как легко видеть из рис. 9, заключается в сильной низкочастотной модуляции выходного сигнала с частотой $F_m = F_0/N$, где F_0 – частота генератора, а N – длина кадра.

В результате при работе, например, на частоте 100 кГц и кадрами длиной 10 мы можем услышать звук с частотой 10 кГц.

Уменьшить такую низкочастотную модуляцию можно за счет применения более сложных шаблонов. В них импульсы идут уже не подряд, а «в разбивку». Пример такого шаблона для тех же 60% мощности показан в нижней части рис. 8.9.

Понятно, что в общем случае разбивка – неоднозначный процесс. Ее можно сделать несколькими способами. Один из вариантов полного набора шаблонов приведен на рис. 8.9. Пунктиром обозначены импульсы, не пропущенные модулятором.

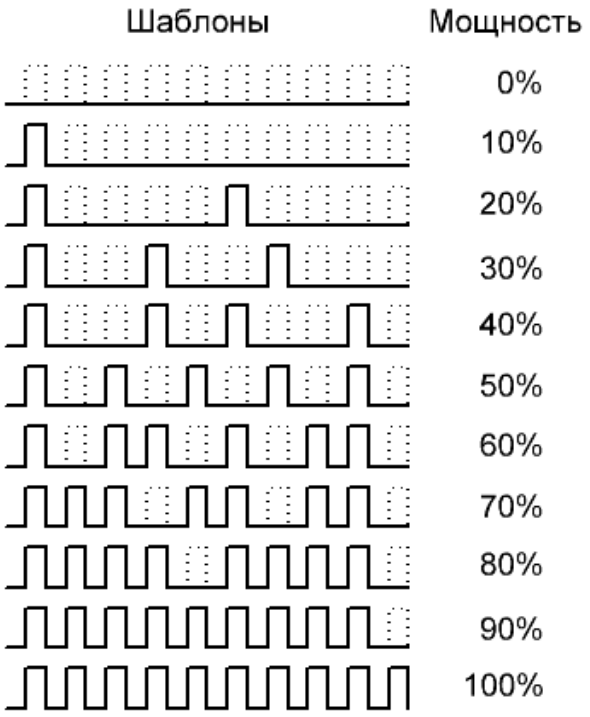

Рисунок 8.9 – Вариант набора шаблонов

На рис.8.9 видно, что полностью избежать низкочастотной модуляции все равно не удается, но характер этой модуляции уже другой. В частности, частота модуляции быстро растет с ростом мощности. Например, $F_0/10$ для 10%, $F_0/5$ для 20% и $F_0/2$ для 50%. В принципе, используя несколько шаблонов для каждого уровня мощности и выбирая их случайным образом, можно «размазать» спектр низкочастотной модуляции по широкому частотному диапазону и, тем самым, снизить среднюю мощность каждой гармоники.

Шаблоны удобны при сравнительно малой длине кадра. Но если мы захотим регулировать мощность с большей точностью (например, изменять ее с шагом 1%), то возникнет проблема, связанная с необходимостью хранить большое количество длинных шаблонов (например, для шага в 1% мы должны использовать 100 шаблонов, длиной по 100 бит). Это уже 1250 байт. А это уже ну очень существенная величина. В этом случае используются другие алгоритмы. Можно позаимствовать алгоритмы, применяемые при цифровой обработки сигналов или задач растровой компьютерной графики (к примеру, алгоритм Флойда–Стейнберга, алгоритм Брезенхема и др.). Но это совсем другая история.

8.3 Задание и порядок выполнения теоретических расчетов

Разработать программу, реализующую широтно-импульсное управление двигателем постоянного тока. Для задания модулирующей частоты использовать прерывание от таймера. Исходный данные выбрать из таблицы 8.1. Если номер зачетной книжки четный, двигатель подключить к линии P1.0, если нечетный – к линии P2.0. Исходя из значения периода модулирующей частоты рассчитать число, которое необходимо занести в таймер.

Таблица 8.1 – Варианты заданий на разработку программы

Последняя цифра зачетки	0	1	2	3	4	5	6	7	8	9
Скважность ШИМ, %	12	50	38	25	75	25	75	62	12	88
Период модулирующей частоты, мкс	80	160	240	320	240	160	80	160	160	80

8.4 Порядок выполнения экспериментальных исследований

После выполнения программы измерить частоту несущей, скважность и определить погрешность заданной и получившейся скважности. В случае необходимости, для уменьшения погрешности скорректировать программу.

8.5 Содержание отчета

1. Задание.
2. Текст программы на языке Ассемблер.
3. Значение частоты несущей, скважность и скважности.
4. Выводы по работе.

8.6 Контрольные вопросы

1. Что такое одноуровневая системы прерываний?
2. Какой из запросов на прерывание имеет высший приоритет?
3. Назовите начальные адреса подпрограмм обслуживания прерываний.
4. Опишите принцип широтно-импульсной модуляции.
5. Что такое скважность?
6. Что такое несущая и модулирующая частота?

9. РАЗРАБОТКА ПРОГРАММЫ УПРАВЛЕНИЯ ОБОРУДОВАНИЕМ НА ОСНОВЕ ЦИКЛОГРАММЫ

9.1 Цель работы

Исследовать возможности использования эмулятора для разработки и отладки управляющей программы системы управления технологическим оборудованием на основе циклограммы.

9.2 Теоретический раздел

Циклограмма – это диаграмма, показывающая время и последовательность выполнения операций в процессе. Считается, что в основе циклограммы лежит диаграмма Ганта.

Для информационного потока циклограмма строится на основе диаграмм потока. Для материального потока в производстве циклограмма строиться на основе технологических карт.

Для построения циклограммы необходимо определить длительность выполнения операций и действий, определенных в диаграмме. Длительность работы – это продолжительность времени, необходимого для выполнения всех операций. Оценка длительности операций зависит от их содержания, операции, требуемых ресурсов и даже от доступности ресурсов. Длительность всего цикла определяется длительностью отдельных операций. Длительность определяется на основе установленных нормативов или экспертной оценки, которая подтверждается на основе хронометража рабочего времени.

Циклограмма последовательности работы механизмов и узлов оборудования является практически алгоритмом его работы и служит исходной информацией для создания системы управления.

Для построения циклограммы необходимо:

1) проанализировать компоновку и определить все движения (переходы) основного и вспомогательного оборудования, необходимые для выполнения заданного цикла технологического процесса;
2) определить (составить перечень) всех механизмов основного и вспомогательного оборудования, участвующих в формировании заданного цикла;
3) задать исходное положение всех механизмов;
4) составить последовательность движений оборудования (механизмов) за цикл в виде таблицы;
5) определить (или принять) время выполнения каждого движения оборудования.

Следует обратить внимание, что сигнал с системы управления в большинстве случаев подаётся не на механизм, осуществляющий движение, а на устройство, обеспечивающее движение этого механизма. Так, например, для выдвижения штока пневмоцилиндра сигнал подаётся на электропнемоклапан, для включения мощного двигателя – на реле и т.п. Пример построения такой диаграммы для теста стиральной машины приведён на рисунке 9.1.

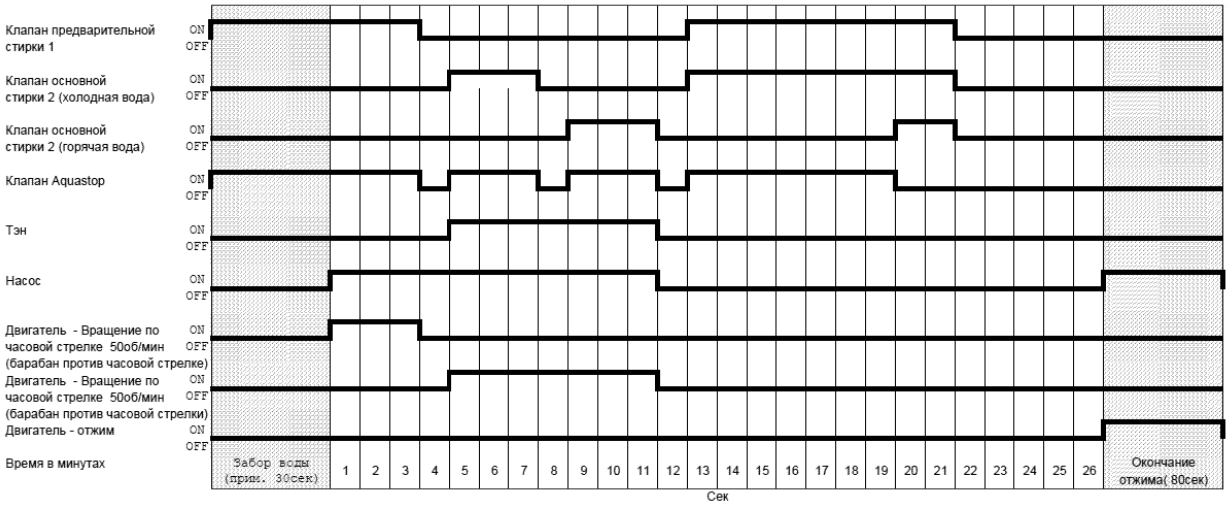

Рисунок 9.1 – Циклограмма быстрого теста стиральной машины

В случае необходимости на циклограмме отображаются датчики (например, датчики конечных положений), которые информируют систему управления о достижении конечной точки движения механизма.

С точки зрения программиста задача разработки программы для системы управления при наличии циклограммы является тривиальной. Необходимо всего лишь подавать на соответствующие линии соответствующего порта последовательность логических нулей и единиц, что уже отображается на циклограмме.

9.3 Задание и порядок выполнения теоретических расчетов

Согласно заданию, для решения предложенной задачи подобрать оборудование, выбрать порты и линии порта для подключения оборудования. Исходный данные выбрать из таблицы 8.1. Составить циклограмму, исходя из которой разработать программу для системы управления. Задержку времени, если это не указанно особо, не учитывать.

Таблица 9.1 – Варианты заданий на разработку программы

Последняя цифра зачетки	Задание
0	Упаковка: банки движутся по конвейеру. Сформировать 4 ряда банок по 3 в каждом ряду.
1	Упаковка: манипулятор опускается, захватывает 12 банок (4x3), переносит, опускает к коробке, отпускает банки, поднимается и возвращается в исходное положение.
2	Упаковка: закрыть попарно клапаны коробки, заклеить лентой.
3	Измерение: произвести измерение диаметра вала в 3 местах измерительной скобой.
4	Измерение: произвести измерение конусности детали.
5	Измерение: для определения радиального биения вала необходимо произвести измерение диаметра вала 6 раз, поворачивая его на 60^{o}.

6	Робототехника: захватить заготовку манипулятором из лотка и установить в патрон токарного станка. Патрон зажимает заготовку, манипулятор перемещается в исходное положение.
7	Робототехника: манипулятор захватывает деталь, поворачивается на 90^{o}, выдвигается, осуществляется маркировка заготовки, задвигается, после чего поворачивается еще на 90^{o} и сбрасывается в бункер.
8	Робототехника: взять заготовку вала из лотка, развернуть на 180^{o} и положить обратно.
9	Медицинские технологии: для стерилизации инструментов в автоклаве подать пар давлением в 2 атмосферы, нагреть до 125^{o} С, подождать, после чего отключить нагрев и снять давление.

9.4 Порядок выполнения экспериментальных исследований

Запустить программу и убедиться в соответствии временных диаграмм соответствующего порта разработанной циклограмме. В случае необходимости скорректировать программу.

9.5 Содержание отчета

1. Задание.
2. Циклограмма работы оборудования.
3. Текст программы на языке Ассемблер.
4. Выводы по работе.

9.6 Контрольные вопросы

1. Для чего применяют циклограмму?
2. Опишите порядок разработки циклограммы.
3. Какие порты имеются у микропроцессора Intel 8048?
4. Можно ли расширить число портов микропроцессора Intel 8048?

АППАРАТНЫЕ И ПРОГРАММНЫЕ РЕШЕНИЯ ПРИ РАЗРАБОТКЕ УПРАВЛЯЮЩИХ УСТРОЙСТВ

Методические указания
к выполнению лабораторных работ

Составители: **Карташов** Леонид Евгеньевич,
Чаленков Никита Игоревич

В авторской редакции

Изд. № 107/2023. Объем 3,75 п.л. Усл. печ. л. 3,49. Уч.-изд. л. 3,675.
РИИЦМ ФГАОУ ВО «Севастопольский государственный университет»