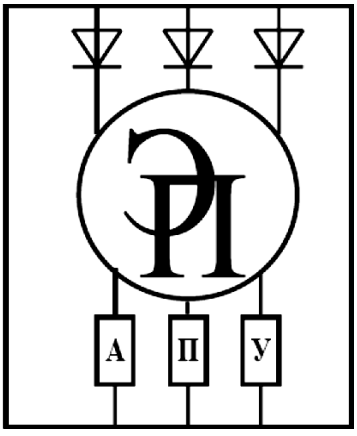
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Электропривод и автоматизация промышленных установок»

МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА УПРАВЛЕНИЯ РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

Методические рекомендации к лабораторным работам для студентов направления подготовки 15.03.06 «Мехатроника и робототехника» очной формы обучения



Могилев 2023

УДК 621.865.8
ББК 34.9
М59

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Электропривод и АПУ» «14» февраля 2023 г., протокол № 6

Составитель ст. преподаватель А. В. Янкович

Рецензент канд. техн. наук, доц. С. В. Болотов

Методические рекомендации к лабораторным работам для студентов направления подготовки 15.03.06 «Мехатроника и робототехника» очной формы обучения.

Учебное издание

МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА УПРАВЛЕНИЯ РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

Ответственный за выпуск	А. С. Коваль
Корректор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 36 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

© Белорусско-Российский
университет, 2023

Содержание

1 Лабораторная работа № 1. Изучение архитектуры микроконтроллера и системы программирования

Цель работы: изучить особенности структурной схемы и системы команд микроконтроллеров семейства MCS-51, порядок работы с учебным стендом СУ-МК и системой программирования ProView; приобретение навыков составления и отладки простейших программ в среде ProView.

1.1 Микроконтроллеры семейства MCS-51

В микропроцессорной технике выделился самостоятельный класс интегральных схем – микроконтроллеры, которые предназначены для встраивания в приборы различного назначения. От класса однокристальных микропроцессоров их отличает наличие встроенной памяти, развитые средства взаимодействия с внешними устройствами.

Микроконтроллеры семейства MCS-51 [1] выполнены на основе высокоуровневой n–МОП-технологии. Через четыре программируемых параллельных порта ввода/вывода и один последовательный порт микроконтроллеры взаимодействуют с внешними устройствами. Основу структурной схемы (рисунок 1.1) образует внутренняя двунаправленная 8-битная шина, которая связывает между собой основные узлы и устройства микроконтроллера: резидентную память программ, резидентную память данных, арифметико-логическое устройство, блок регистров специальных функций, устройство управления, блок прерываний таймеров, последовательного порта и порты ввода/вывода.

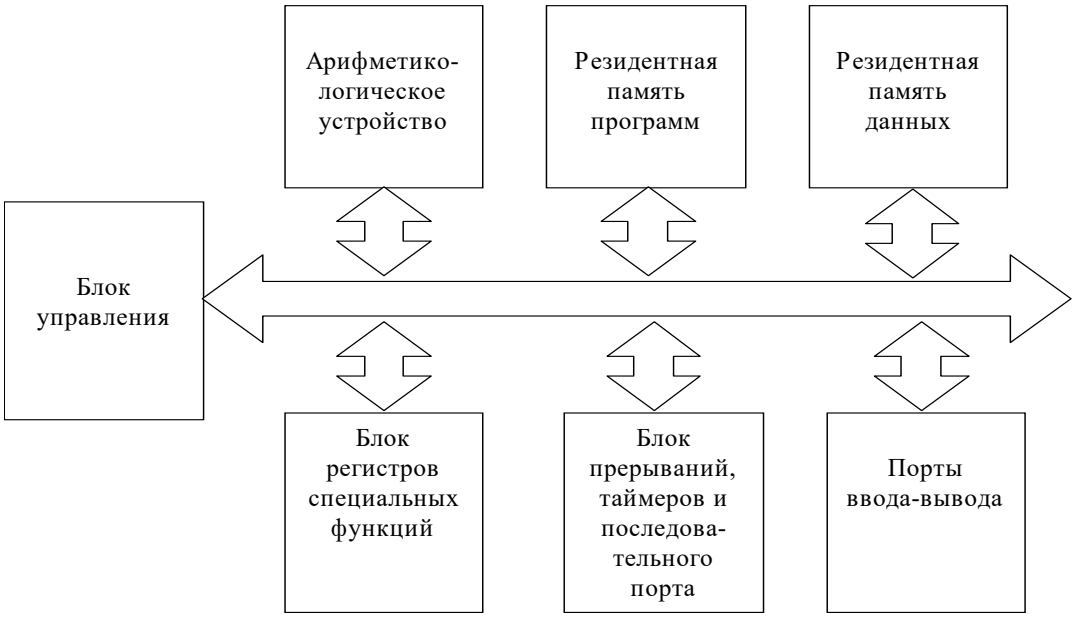


Рисунок 1.1 – Структурная схема микроконтроллеров семейства MCS-51

Система команд микроконтроллеров MCS-51 содержит 111 базовых команд, которые по функциональному признаку могут быть разделены на пять групп:

1) команды передачи данных;
2) арифметические операции;
3) логические операции;
4) операции с битами;
5) команды передачи управления.

Большинство команд имеют формат в 1 или 2 байта и выполняются за один или два машинных цикла. При тактовой частоте 11,059 МГц длительность машинного цикла составляет 1,085 мкс. Первый байт команды любых типов и форматов всегда содержит код операции (КОП). Второй и третий байты содержат либо адреса операндов, либо непосредственные операнды.

1.2 Структура учебного стенда СУ-МК

Учебный стенд СУ-МК состоит из следующих структурных элементов (рисунок 1.2).

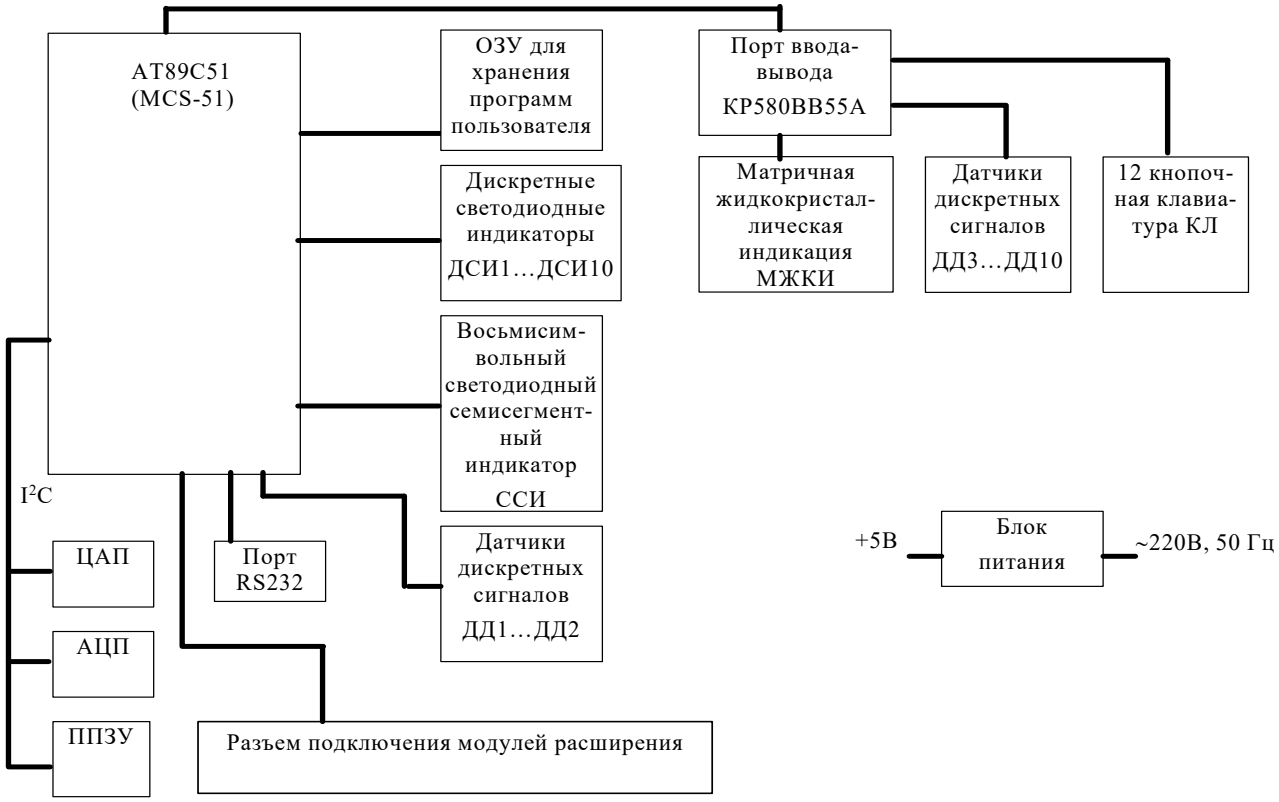


Рисунок 1.2 – Структурная схема учебного стенда СУ-МК

Стенд построен на базе микроконтроллера AT89C51 семейства MCS-51.

К микроконтроллеру подключено оперативное запоминающее устройство (ОЗУ) для хранения программ пользователей. Для исследования вывода дискретных сигналов используются дискретные светодиодные индикаторы ДСИ1…ДСИ10. Для исследования применения динамической семисегментной индикации используется восьмисимвольный семисегментный светодиодный

индикатор (ССИ). Для исследования ввода дискретных сигналов используются датчики дискретных сигналов ДД1…ДД2.

К микроконтроллеру подключен порт ввода-вывода КР580ВВ55А. К порту подключены: устройство матричной жидкокристаллической индикации МЖКИ, датчики дискретных сигналов ДД3…ДД10 и двенадцатикнопочная клавиатура КЛ.

В стенде организована шина I2C, по которой к микроконтроллеру подключены: аналого-цифровой преобразователь (АЦП), цифроаналоговый преобразователь (ЦАП) (цифровой резистор), и электрически стираемое программируемое постоянное запоминающее устройство (ППЗУ).

Для связи с другими устройствами, например с персональным компьютером, стенд оснащен последовательным портом RS232.

Стенд оснащен разъемом подключения внешних модулей расширения.

Все устройства, входящие в состав стенда, кроме ОЗУ программ пользователя, являются программно-доступными.

Для питания стенда используется внешний блок питания на 5 В.

Внешний вид передней панели стенда СУ-МК приведен на рисунке 1.3.

На передней панели учебного стенда СУ-МК расположены:

– дискретный светодиодный индикатор (набор из 10 светодиодов ДСИ1…ДСИ10) (*1*);

– восьмисимвольный семисегментный светодиодный индикатор ССИ (*2*);

– матричный жидкокристаллический индикатор МЖКИ (*3*);

– линейка светодиодов – индикатор аналогового сигнала на выходе ЦАП – ЛСИ (*4*);

– имитаторы аналогового сигнала на входах АЦП АД1…АД3 (*5*);

– датчики дискретных сигналов (набор из 10 переключателей ДД1…ДД2) (*6*);

– кнопка сброса (*7*);

– двенадцатикнопочная клавиатура КЛ (*8*).

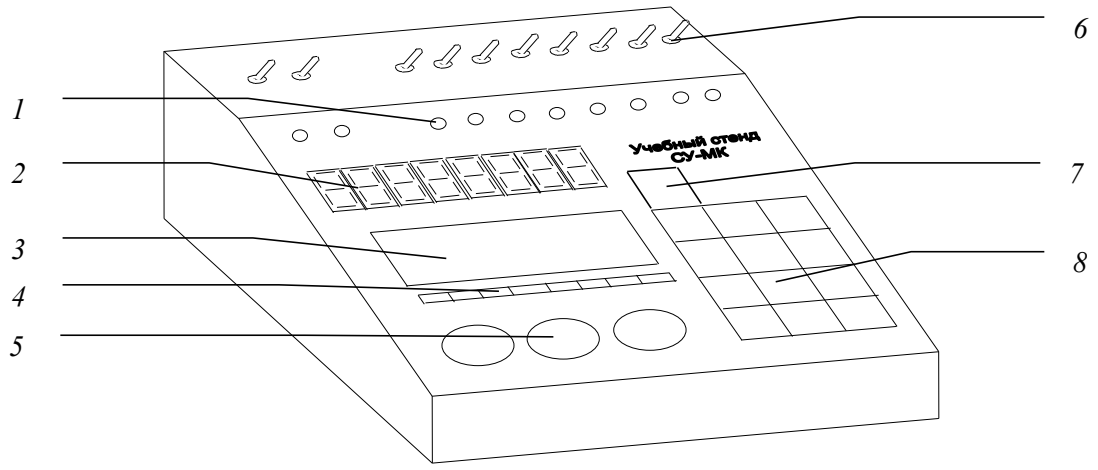


Рисунок 1.3 – Внешний вид передней панели учебного стенда СУ-МК

Задняя панель лабораторного стенда приведена на рисунке 1.4.

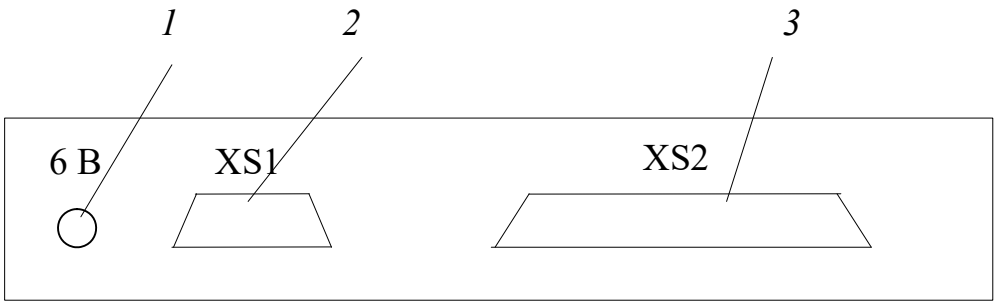


Рисунок 1.4 – Вид задней панели учебного стенда СУ-МК

На задней панели лабораторного стенда расположены:

– разъем подключения питания (*1*);

– разъем RS232 для подключения стенда к персональному компьютеру (*2*);

– разъем расширения для подключения дополнительных внешних устройств (*3*).

Неотъемлемой частью учебного стенда является программа «Загрузчик СУ-МК», предназначенная для управления режима работы стенда и для загрузки пользовательской программы в память программ стенда.

Программа имеет традиционный для Windows интерфейс. Для соединения лабораторного стенда с программатором используется последовательный порт, работа с которым производится с использованием стандартных функций API Windows, что обеспечивает полную совместимость программы со всеми версиями Windows, начиная с Windows 95.

Программа позволяет выполнять следующие операции:

– чтение файла с пользовательской программой с диска в буфер;

– запись данных из буфера в стенд;

– сравнение данных из буфера с данными со стенда;

– переключение режимов работы стенда программирование/выполнение пользовательской программы;

– перезапуск программы стенда;

– работа в режиме терминала;

– выбор порта.

Поддерживаются файлы следующих форматов:

BIN-файлы (чистый машинный код);

HEX-файлы (файлы формата HEX от фирмы INTEL).

Тип файла можно выбрать в диалоге при открытии файла. Максимальный размер загружаемого файла – 32 КБ.

Открыть файл можно:

– через меню: «Файл» –> «Открыть»;

– нажатием клавиш Ctrl+O;

– нажатием кнопки на панели инструментов.

Программирование стенда заключается в загрузке программы пользователя в память стенда.

Включить программирование можно:

– через меню «Программирование» -> «Запись в стенд»;

– нажатием клавиш Ctrl + W;

– нажатием кнопки на панели инструментов.

После программирования стенд автоматически переходит в режим выполнения программы.

Лабораторный стенд подключается к компьютеру через последовательный порт. Программа поддерживает до четырех COM-портов. При запуске программы проверяется наличие портов и возможность их использования. Если по каким-то причинам порт использовать невозможно (он физически отсутствует или занят другим устройством), то в разделе меню «Порт» он отображается серым цветом. Активный порт обозначается кружком. При выборе другого порта программа пытается его инициализировать. Если инициализация не удалась, то активным остается старый порт.

При запуске программы осуществляется попытка инициализации порта, используемого в предыдущем сеансе. Если выясняется, что ни один COM-порт использовать невозможно, то выдается сообщение «Нет доступных COM-портов» и выполнение программы завершается. Для корректной работы лабораторного стенда MCS-51 программа осуществляет следующую настройку COM-порта:

– скорость работы – 9600 бод;

– длина посылки – 8 бит;

– контроль четности – НЕТ;

– количество стоп-бит – 1.

Порядок работы с учебным стендом СУ-МК:

– подключить стенд к персональному компьютеру, для этого необходимо подключить прилагаемый к стенду кабель к разъему RS232 стенда и к одному из COM-разъемов персонального компьютера в соответствии со схемой, приведенной на рисунке 1.5;

– подключить с помощью адаптера к сети 220 В в соответствии со схемой (см. рисунок 1.5);

– подготовить откомпилированный HEX-файл с кодом пользовательской программы;

– загрузить программу управления работой стенда «Загрузчик СУ-МК» («ПУСК» → «Программы» → «Учебный стенд СУ-МК» → «Загрузчик СУ-МК»);

– при необходимости, выбрать порт (COM1…COM4), к которому подключен стенд СУ-МК;

– загрузить откомпилированную программу в виде HEX- или BIN-файла в программу загрузчика;

– загрузить пользовательскую программу в память программ стенда при помощи программы загрузчика;

– после загрузки стенд автоматически переходит в режим выполнения пользовательской программы.

1.3 Интегрированная среда ProView

ProView фирмы Franklin Software Inc. – интегрированная среда разработки программного обеспечения для однокристальных микроконтроллеров семейст-

ва Intel 8051 и его клонов. Она включает в себя всё, что нужно для создания, редактирования, компиляции, трансляции, компоновки, загрузки и отладки программ:

– стандартный интерфейс Windows;

– полнофункциональный редактор исходных текстов с выделением синтаксических элементов цветом;

– организатор проекта;

– транслятор с языка С;

– ассемблер;

– отладчик;

– встроенную справочную систему.

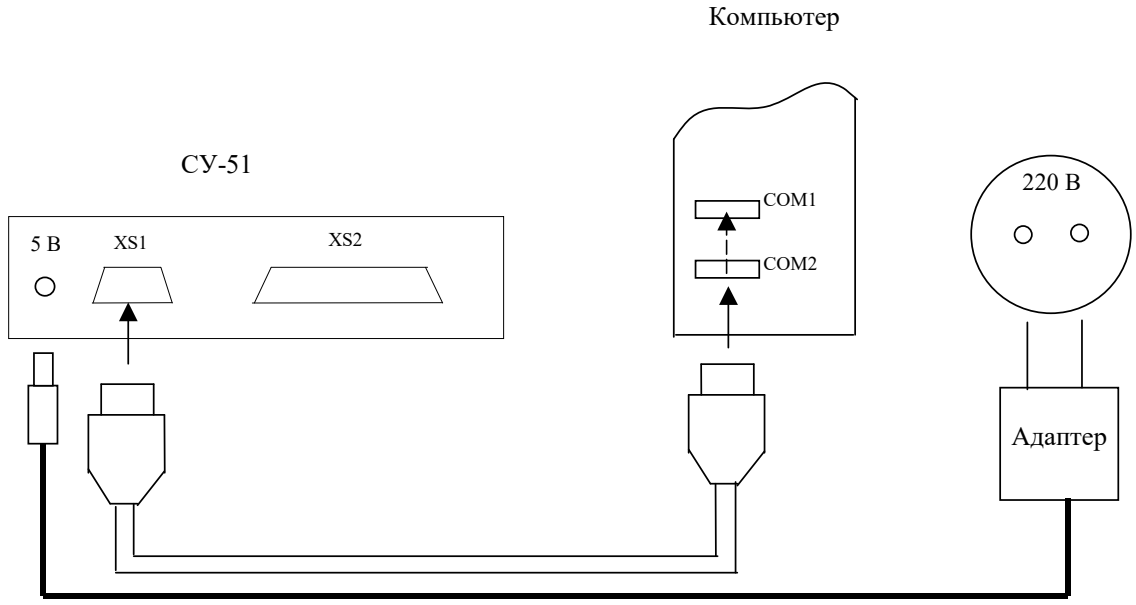


Рисунок 1.5 – Схема подключения учебного стенда СУ-МК

Первый этап разработки программы – запись её исходного текста на каком-либо языке программирования.

Затем производится компиляция или трансляция его в коды из системы команд микроконтроллера, используя транслятор или ассемблер. Трансляторы и ассемблеры – прикладные программы, которые интерпретируют текстовый файл, содержащий исходный текст программы, и создают объектные файлы, содержащие объектный код.

После компоновки объектных модулей наступает этап отладки программы, устранения ошибок, оптимизации и тестирования программы.

ProView объединяет все этапы разработки прикладной программы в единый рекурсивный процесс, когда в любой момент времени возможен быстрый возврат к любому предыдущему этапу.

ProView запускается из стартового меню Windows подобно остальным приложениям. Если необходимо запустить программу из командной строки, её синтаксис имеет вид: PV32 [projectfile], где projectfile – имя файла проекта с расширением [.PRJ].

Любая новая работа в ProView, как и во всех современных компиляторах, начинается с создания нового файла проекта. Файл проекта содержит имена всех исходных файлов, связанных с проектом, а также установки компиляции, трансляции и связывания файлов, чтобы генерировать выполняемую программу.

Для того чтобы создать новый файл проекта, выберите New из меню Project. Откроется диалоговое окно New Project. Используйте кнопку Browse, чтобы войти в свою папку. Найдите свою папку и нажмите кнопку [OK]. Затем выберите «8051» как тип проекта.

Когда менеджер проекта открывает файл проекта, окно проекта показывает включенные исходные файлы.

1.4 Порядок выполнения лабораторной работы

1 Составить программу в соответствии с заданием. Программа должна выполнять все действия и расчеты, приведенные в задании, в том числе и промежуточные.

2 Отладить программу в среде ProView.

3 Выполнить программу в пошаговом режиме.

4 Составить отчет.

Содержание отчета

1 Цель работы.

2 Постановка задачи.

3 Текст программы.

4 Таблица хода выполнения программы.

5 Выводы по работе.

1.5 Варианты индивидуальных заданий

Варианты индивидуальных заданий приведены в таблице 1.1.

Таблица 1.1 – Варианты задания к лабораторной работе

Номер варианта	Задание
1	Загрузить в регистр R0 число 15, сложить его с 25 и результат поместить на вершину стека. Поместить по адресу 020h внутренней памяти данных младшую десятичную цифру результата, а по адресу 021h – старшую
2	Найти разницу чисел 4836 и 232. Младший байт результата поделить на 2. Поместить по адресу 025h внутренней памяти данных младший байт результата, а по адресу 030h – старший байт
3	Найти адрес ячейки внешней памяти данных путем перемножения двух чисел 0Ch и 0Eh. В эту ячейку записать результат логической операции «исключающее или» между текущим содержимым аккумулятора и числа 09h

Окончание таблицы 1.1

Номер варианта	Задание
4	Найти частное чисел 236 и 59. Результат умножить на 23, используя операции сдвига. По вычисленному таким образом адресу ячейки внутренней памяти данных разместить результат двойного декремента полученного числа
5	В младшую тетраду порта P1 вывести число десятков от числа 044h. Старшую тетраду необходимо оставить без изменений
6	Загрузить регистр R0 числом 023h. Найти сумму R0+PC. В ячейку внутренней памяти данных, расположенную по вычисленному таким образом адресу, загрузить число десятичных единиц результата сложения
7	Найти сумму чисел 9701 и 32. Младший байт результата умножить на 4. Поместить старший байт в порт P1. Сбросить младшую тетраду байта порта
8	Вычислить значение выражения (81 + 64)·(112 – 25) OR 10011010b, сохраняя промежуточные результаты в стеке
9	Загрузить в регистр R0 число 112, вычесть из него число 18 и результат поместить на вершину стека. Поместить по адресу 020h внутренней памяти данных младшую десятичную цифру результата, а по адресу 021h – старшую
10	Найти разницу чисел 4801 и 209. Число десятичных единиц старшего байта результата поместить в старшую тераду порта P0. Младшую тетраду оставить без изменений
11	Загрузить регистр R0 числом 0Ah. Найти произведение 8·R0. Результат разделить на 4, используя операции сдвига. В две ячейки внутренней памяти данных, начиная с ячейки, расположенной по вычисленному таким образом адресу, загрузить число 023E8h
12	В ячейки внешней памяти данных 01A00h, 01A01h, 01A02h занести число сотен, десятков, единиц числа 080h
13	Вычислить младший байт адреса ячейки внешней памяти данных 0A1XXh как частное 0A1h и 7, поместить по этому адресу значение выражения NOT (0101001b OR 74)
14	Вычислить значение выражения (72 + 56)·(122 – 15) XOR 01010011b, сохраняя промежуточные результаты в стеке
15	Найти среднее арифметическое чисел 012h, 033h, 0Ah. Результат умножить на 22, используя операции сдвига. Младшую тераду полученного числа разместить в старшей тетраде порта P1, а старшую в младшей

1.6 Пример выполнения задания

Необходимо найти значение выражения (25 + 13)·(23 – 8), сохраняя промежуточные результаты в стеке. Младшую тетраду полученного результата необходимо поместить в старшую тетраду порта P0.

Программа для выполнения задания.

```
MOV   A,  #25
ADD   A,  #13          ;Выполнение первого действия
PUSH  ACC              ;Сохранение результата
MOV   A,  #23
SUBB  A,  #8           ;Выполнение второго действия
```

```
POP     B               ;Вызов результата первого действия
MUL     AB              ;Третье действие
SWAP    A               ;Обмен тетрад
ANL     A,  #11110000b  ;Нам нужна старшая тетрада
PUSH    ACC             ;Сохранение результата
MOV     A,  P0
ANL     A,  #00001111b  ;Сохраняем младшую тетраду порта P0
MOV     R0, A
POP     ACC
ORL     A,  R0
MOV     P0, A           ;Результат в порт

STOP:   JMP             STOP ; Программа должна завершаться именно так

END
```

Создание проекта и тестирование программы в среде ProView.

Загрузить программу ProView. Если после загрузки программы открывается предыдущий проект, с которым производилась работа, то необходимо закрыть его через меню.

Создать новый проект (рисунок 1.6).

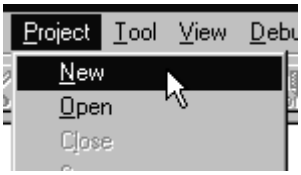


Рисунок 1.6 – Создание нового проекта

Укажите имя файла проекта в вашей личной папке (рисунок 1.7).

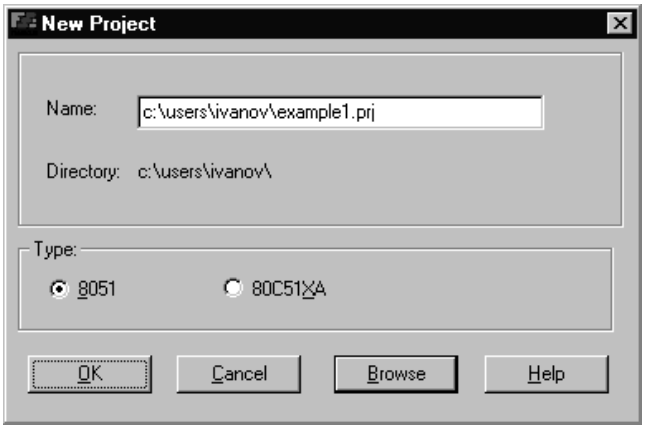


Рисунок 1.7 – Имя и путь файла проекта

Создать новый файл для последующего ввода текста программы (рисунок 1.8) и указать тип файла (рисунок 1.9).

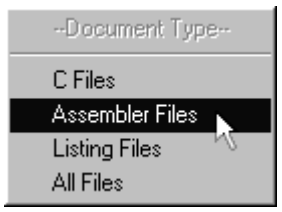


Рисунок 1.8 – Новый файл

Рисунок 1.9 – Тип файла

Откроется пустое окно для ввода ассемблерного текста программы. Записать этот файл (рисунок 1.10).

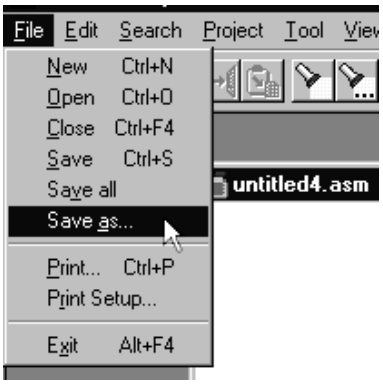
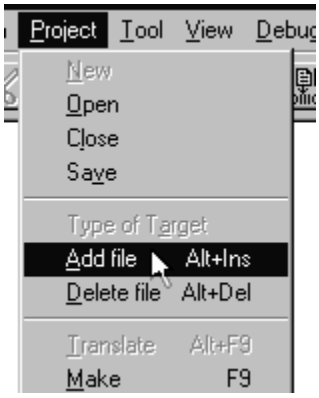


Рисунок 1.10 – Запись файла

Ввести текст программы.

Подключить этот файл к проекту, предварительно активизировав окно проекта (рисунок 1.11).



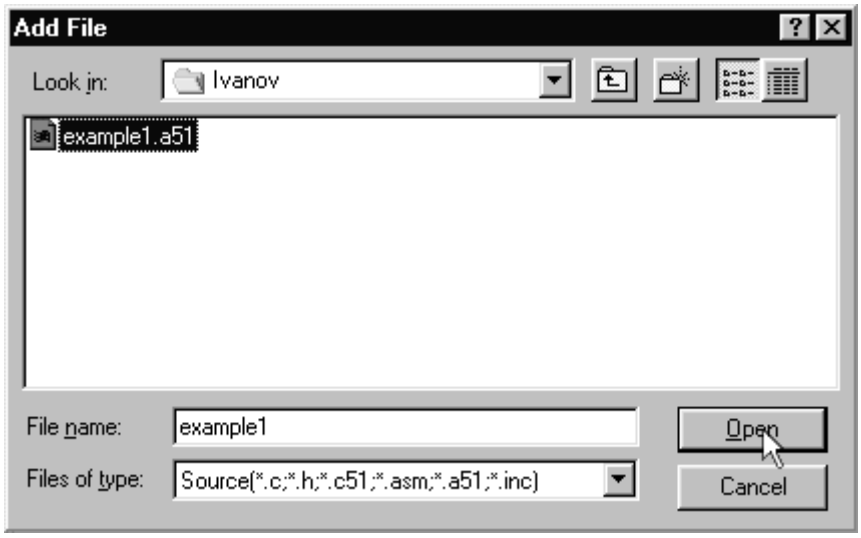


Рисунок 1.11 – Выбор файла для подключения

Выполнить компиляцию проекта (рисунок 1.12).

Для того чтобы в результате компиляции генерировался Intel HEX-файл, включите эту опцию с помощью команды Project меню Options (рисунок 1.13).

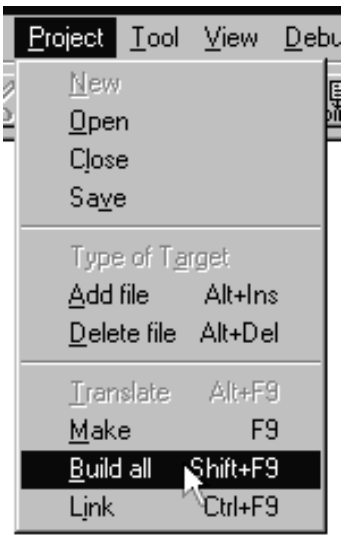


Рисунок 1.12 – Компиляция

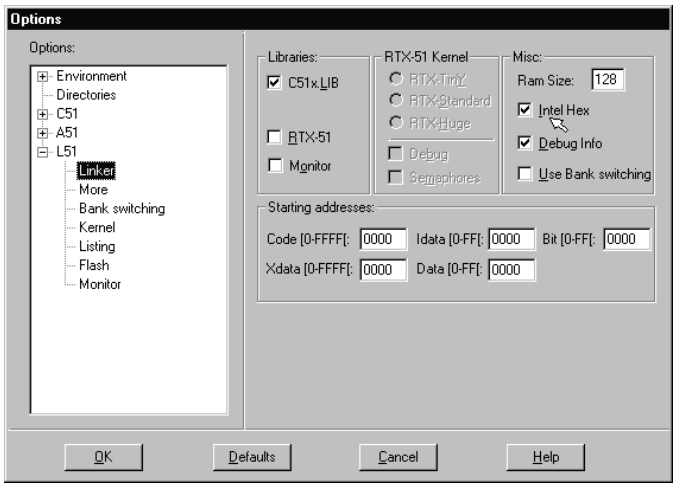


Рисунок 1.13 – Настройка сборки проекта

Запустить программу на выполнение с помощью команды Start из меню Debug, указав необходимые опции отладчика (рисунок 1.14).

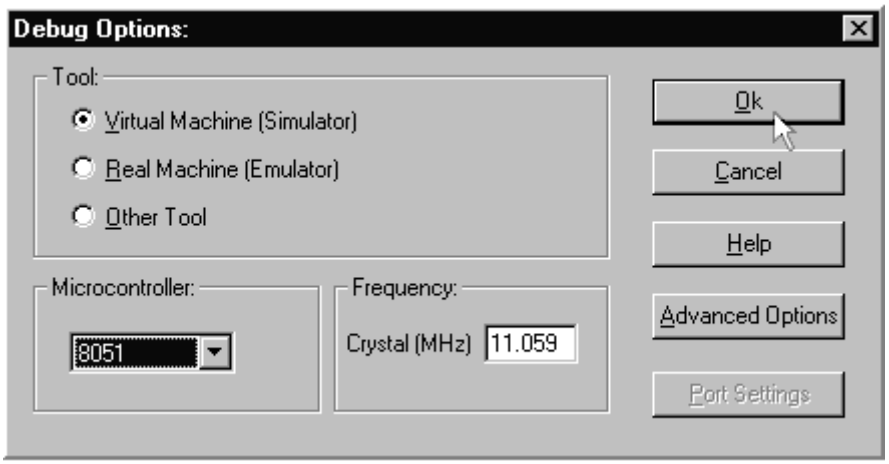


Рисунок 1.14 – Окно опций отладчика

Выполнить отладку программы, используя информационные окна отладчика, в пошаговом режиме.

Контрольные вопросы

1 Перечислите характерные черты архитектуры однокристальных микроконтроллеров.

2 Какие структурные элементы входят в состав микроконтроллеров семейства MCS-51?

3 Каковы основные структурные элементы учебного стенда СУ-МК?

4 Опишите органы управления лабораторного стенда.

5 Каков порядок работы со стендом?

2 Лабораторная работа № 2. Разработка и отладка программы управления устройствами ввода-вывода

Цель работы: изучить структуру и особенности работы портов микроконтроллера, схему подключения входных и выходных дискретных сигналов к микроконтроллеру, особенности программирования ввода-вывода дискретных сигналов на языке программирования микроконтроллеров; составить программу ввода, обработки по заданному алгоритму и вывода дискретных сигналов, записать в память программ микроконтроллера и выполнить.

2.1 Параллельные порты ввода/вывода информации микроконтроллера MCS-51

Все четыре порта (P0…P3) предназначены для ввода или вывода информации побайтно. Каждый порт содержит управляемые регистр-защёлку, входной буфер и выходной драйвер [1].

Выходные драйверы портов 0 и 2, а также входной буфер порта 0 используются при обращении к внешней памяти. При этом через порт 0 в режиме временного мультиплексирования сначала выводится младший байт адреса, а затем выдается или принимается байт данных. Через порт 2 выводится старший байт адреса в тех случаях, когда разрядность адреса равна 16 бит.

Все выводы порта 3 могут быть использованы для реализации альтернативных функций, перечисленных в таблице 2.1. Эти функции могут быть задействованы путем записи 1 в соответствующие биты регистра-защёлки (P3.0…P3.7) порта 3.

Порт 0 является двунаправленным, а порты 1–3 – квазидвунаправленными. Каждая линия портов может быть использована независимо для ввода или вывода.

По сигналу RST в регистры-защёлки всех портов автоматически записываются единицы, настраивающие их тем самым на режим ввода.

Таблица 2.1 – Альтернативные функции порта P3

Символ	Разряд	Имя и назначение
RD	P3.7	Чтение. Активный сигнал низкого уровня формируется аппаратно при обращении к внешней памяти данных
WR	P3.6	Запись. Активный сигнал низкого уровня формируется аппаратно при обращении к внешней памяти данных
T1	P3.5	Вход таймера/счётчика 1 или тест-вход
T0	P3.4	Вход таймера/счётчика 0 или тест-вход
INT1	P3.3	Вход запроса прерывания 1. Воспринимается сигнал низкого уровня или срез
INT0	P3.2	Вход запроса прерывания 0. Воспринимается сигнал низкого уровня или срез
TXD	P3.1	Выход передатчика последовательного порта в режиме UART. Выход синхронизации в режиме регистра сдвига
RXD	P3.0	Вход приёмника последовательного порта в режиме UART. Ввод/вывод данных в режиме регистра сдвига

Все порты могут быть использованы для организации ввода/вывода информации по двунаправленным линиям передачи. Однако порты 0 и 2 не могут быть использованы для этой цели в случае, если система имеет внешнюю память, связь с которой организуется через общую разделяемую шину адреса/данных, работающую в режиме временного мультиплексирования.

Обращение к портам ввода/вывода возможно с использованием команд, оперирующих с байтом, отдельным битом, произвольной комбинацией битов. При этом в тех случаях, когда порт является одновременно операндом и местом назначения результата, устройство управления автоматически реализует специальный режим, который называется «чтение – модификация – запись». Этот режим обращения предполагает ввод сигналов не с внешних выводов порта, а из его регистра-защёлки, что позволяет исключить неправильное считывание ранее выведенной информации.

2.2 Ввод дискретных сигналов

Для ввода дискретной информации в микроконтроллер широко применяются различные переключатели, кнопки и клавиатуры. Пример простейшей схемы подключения источника дискретного сигнала в виде кнопки к порту микроконтроллера приведен на рисунке 2.1.

Типовая процедура ожидания события [2] состоит из следующих действий: ввода сигнала от датчика, анализа значения сигнала и передачи управления в зависимости от состояния датчика. Конкретная программная реализация процедуры зависит от того, каким образом датчик подключен к микроконтроллеру. Например, при подключении датчика к линии бита 3 порта 1 программа ожидания замыкания контакта будет иметь вид:

```
#INCLUDE <reg51.h>
SBIT SB_1 = P1^3;
…
WHILE (SB_1) ;     ожидание замыкания контакта датчика
WHILE (! SB_1) ;   ожидание размыкания контакта датчика
```

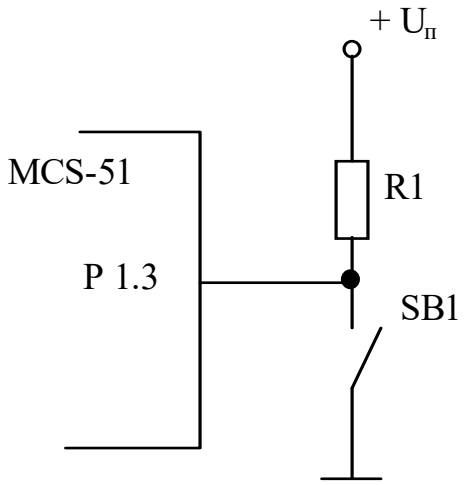


Рисунок 2.1 – Подключение источника дискретных сигналов к порту микроконтроллера

Особенность процедуры ожидания импульсного сигнала состоит в том, что микроконтроллер должен обнаружить не только факт появления, но и факт окончания сигнала.

Для программирования этой процедуры удобно воспользоваться рассмотренными выше примерами, смонтировав их последовательно в линейную программу.

Программная реализация цикла ожидания накладывает ограничения на длительность импульса: импульсы длительностью меньше времени выполнения цикла ожидания могут быть «не замечены» микроконтроллером. Для обнаружения кратковременных импульсов обычно используют способ фиксации импульса на внешнем триггере флага. На вход в этом случае поступает не кратковременный сигнал с датчика, а флаг, формируемый триггером. Триггер устанавливается по фронту импульса, а сбрасывается программным путем – выдачей специального управляющего воздействия. Длительность импульса при этом будет ограничена снизу только быстродействием триггера.

При работе с датчиками, имеющими механические или электромеханические контакты (кнопки, клавиши, реле и клавиатуры), возникает явление, называемое дребезгом. Он заключается в том, что при замыкании контактов возможно появление отскока (BOUNCE) контактов, которое приводит к переходному процессу. При этом сигнал с контакта может быть прочитан микроконтроллером как случайная последовательность нулей и единиц. Подавить это нежелательное явление можно схемотехническими средствами, но чаще это делается программным путем.

Наибольшее распространение получили два программных способа ожидания установившегося значения:

1) подсчет заданного числа совпадающих значений сигнала;
2) временная задержка.

Суть первого способа состоит в многократном считывании сигнала с контакта. Подсчет удачных опросов, обнаруживших, что контакт устойчиво замкнут, ведется программным счетчиком. Если после серии удачных опросов встречается неудачный, то подсчет начинается сначала. Контакт считается устойчиво замкнутым, если последовало N удачных опросов. Число N подбирается экспериментально для каждого типа используемых датчиков и лежит в пределах от 5 до 50. Пример программного подавления дребезга контакта приводится для случая, когда датчик импульсного сигнала подключен к входу P3.4, счет удачных опросов ведется до N = 20:

```
#INCLUDE <reg51.h>
sbit testport = P3^4;

void main(void)
{
  unsigned char Count;
  ...
  for (Count = 0; Count < 20; Count++)
  {
    if (! testport) Count = 0;
  }
  ...
}
```

Устранение дребезга контакта путем введения временной задержки заключается в следующем [3].

Программа, обнаружив замыкание контакта, запрещает опрос его состояния на время, заведомо большее длительности переходного процесса. Программа написана для случая подключения датчика к входу T0 и программной реализации временной задержки. Временная задержка в пределах 1…10 мс подбирается экспериментально для каждого типа датчиков и реализуется подпрограммой DELAY.

```
#INCLUDE <reg51.h>

void delay10 (void)     //Задержка на 10 мс на частоте 11,059
{
 unsigned short register x;
 x = 1151;
 while (--x!=0);
}
```

```
void main(void)
{
...
  while (!T0);
  delay10();
...
}
```

Еще одним способом организации сбора информации с дискретных датчиков является их периодический опрос. При этом, если подобрать соответствующий период опроса, можно устранить и нежелательный эффект дребезга контактов.

Чаще всего такой опрос организуется посредством прерываний от таймеров.

Следующий фрагмент программы на С организует периодический опрос состояния датчика, подключенного к входу P1.1 и выполнение определенных действий в зависимости от состояния датчика.

```
#INCLUDE <reg51.h>
sbit testport = P1^1;
sbit outport = P3^2;

unsigned char count;

void timer0 (void) interrupt 1 using 2
{
  if (++count = 40)          /* Организация опроса датчиков */
                             /* каждые 10 мс */
  {
   if (testport) {outport = 1;}      /*Вывод сигнала по линии P3.2*/
   else {outport = 0;}               /*в зависимости от состояния линии P1.1*/
   count = 0;
  }
}

void main (void)
{
  /* начальные установки таймера 0 и прерывания */
 TH0 = 25;           /* Установка периода таймера */
 TL0 = 25;           /* ~250 мкс на частоте 11,059 МГц */
 TMOD = 0x02;              /* Выбор режима 2 */
 TR0 = 1;                  /* Запуск таймера 0 */
 ET0 = 1;                  /* Разрешение прерывания от таймера 0 */
 EA = 1;                        /* Глобальное разрешение прерываний */
}
```

Микроконтроллеры чаще всего имеют дело не с одним датчиком, как в рассмотренных выше примерах, а с группой автономных, логически независимых или взаимосвязанных, формирующих двоичный код датчиков. При этом микроконтроллер может выполнять процедуру опроса датчиков и передачи управления отдельным фрагментам прикладной программы в зависимости от принятого кода.

Программную реализацию процедуры ожидания заданного кода рассмотрим для случая подключения группы из восьми взаимосвязанных статических датчиков к входам порта 1:

```
while (P1!=10);
```

При опросе двоичных датчиков передачу управления удобно осуществлять по таблице переходов. Ниже приводится текст программы, осуществляющей передачу управления одной из восьми прикладных программ PROG0…PROG7. Передача производится в зависимости от кодовой комбинации на входах P1.0…P1.2:

```
unsigned char a;

a = P1 & 7;
switch (a)
{
 case 0: prog0(); break;
  ...
 case 7: prog7();
}
```

2.3 Вывод дискретных сигналов

Для управления исполнительным устройством, работающим по принципу включено/выключено, на соответствующей выходной линии порта необходимо сформировать статический сигнал 0 или 1, что реализуется командами вывода непосредственного операнда, содержащего в требуемом бите значение 0 или 1. На рисунке 2.2 приведен пример простейшего исполнительного устройства в виде светодиода, подключенного к порту микроконтроллера [3].

В случае параллельного управления группой автономных исполнительных устройств, подключенных к выходному порту, формируется не двоичное управляющее воздействие, а управляющее слово, каждому из разрядов которого ставится в соответствие 1 или 0 в зависимости от того, какие исполнительные устройства должны быть включены, а какие выключены.

Управляющие слова удобно формировать командами логических операций над содержимым порта. Команда ANL используется для сброса тех битов, которые в маске заданы нулём. Команда ORL используется для установки битов управляющего слова. Командой XRL осуществляется инверсия бита.

Для формирования сложных последовательностей управляющих слов

обычно используют табличный способ, при котором все возможные слова упакованы в таблицу, а прикладная программа вычисляет адрес требуемого слова, выбирает его из таблицы и передаёт в порт.

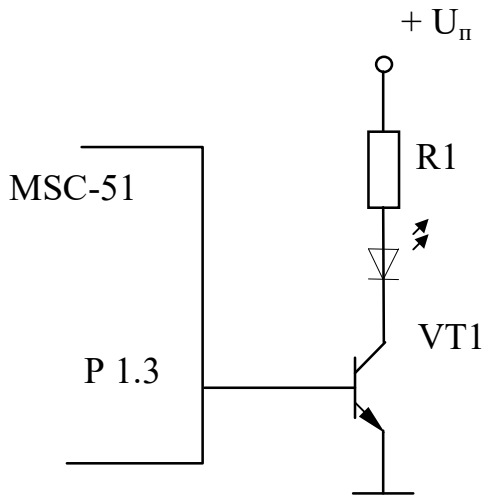


Рисунок 2.2 – Подключение исполнительного устройства к порту микроконтроллера

Импульс можно получить последовательной выдачей сигналов включения и отключения с промежуточным вызовом подпрограммы временной задержки:

```
#INCLUDE <reg51.h>
sbit outport = P1^3;
void delay100 (void)     //Задержка на 100 мс на частоте 11,059
{
 unsigned short register x;
 x = 11514;
 while (--x!=0);
}
void main(void)
{
   outport = 0;
   delay100();
   outport = 1;
   delay100();
   outport = 0;
…}
```

Длительность импульса определяется временной задержкой, реализуемой подпрограммой DELAY.

2.4 Схема к лабораторной работе

На рисунке 2.3 изображена электрическая принципиальная схема к лабораторной работе.

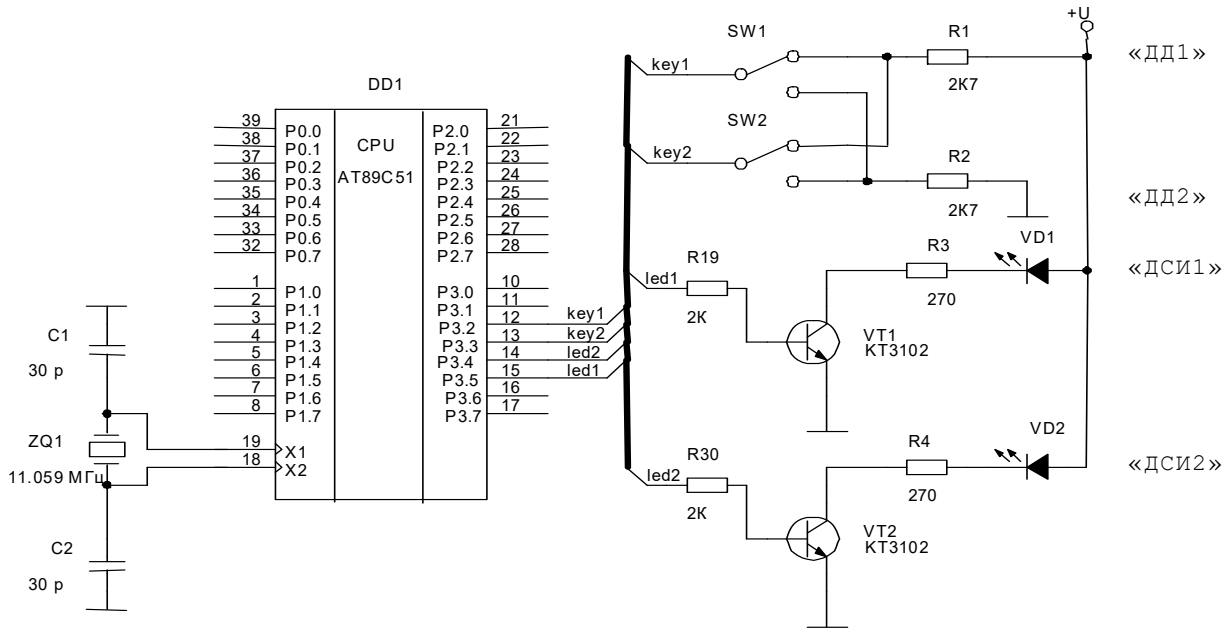


Рисунок 2.3 – Электрическая принципиальная схема к лабораторной работе

В схеме два дискретных датчика ДД1 и ДД2 оформлены в виде двух переключателей SW1 и SW2, подключенных к выводам P3.2 и P3.3 микроконтроллера. Два дискретных индикатора оформлены в виде двух светодиодов ДСИ1 и ДСИ2, подключенных через транзисторные ключи к выводам P3.5 и P3.4 микроконтроллера.

2.5 Варианты индивидуальных заданий к лабораторной работе

Разработать программу для учебного стенда СУ-МК, позволяющую выполнять следующие действия:

1 Если ДД1 = 0, то ДСИ1 = 0 и ДСИ2 = 0; иначе, если ДД2 = 0, то ДСИ1 = 1, ДСИ2 = 0, если ДД2 = 1, то ДСИ1= 0, ДСИ2 = 1.

2 Если ДД1 = 1 и ДД2 = 0, то ДСИ1 = 0 и ДСИ2 = 0, если ДД1 = 0 и ДД2 = 1, то ДСИ1 = 1, ДСИ2 = 1, если ДД1 = ДД2, то ДСИ1 = 0, ДСИ2 = 1.

3 Если ДД1 = 1, то ДСИ1 = 0 и ДСИ2 = 0; иначе, если ДД2 = 1, то ДСИ1 = 1, ДСИ2 = 0, если ДД2 = 0, то ДСИ1 = 0, ДСИ2 = 1.

4 Если ДД1 = 1 и ДД2 = 1, то ДСИ1 = 0 и ДСИ2 = 0, если ДД1 = 0 и ДД2 = 0, то ДСИ1=1, ДСИ2 = 1, если ДД1 ≠ ДД2, то ДСИ1 = 1, ДСИ2 = 0.

5 Если ДД1 = 1, то ДСИ1 = 0 и ДСИ2 = 0; иначе, по приходу на ДД2 положительного импульса длительностью более 1 с, ДСИ1 = 1 и ДСИ2 = 1.

6 Если ДД2 = 0, то ДСИ1 = 0 и ДСИ2 = 0; иначе, по приходу на ДД1 отрицательного импульса длительностью более 1,5 с, ДСИ1 = 1 и ДСИ2 = 1.

7 Если ДД2 = 1, то ДСИ1 = 0 и ДСИ2 = 0; иначе, если ДД1 = 1, то ДСИ1 = 1 и ДСИ2 = 0, если ДД1 = 0, то ДСИ1 = 0 и ДСИ2 = 1; опрос ДД1 организовать с устранением дребезга контактов путем введения временной задержки.

8 Если ДД1 = 0, то ДСИ1 = 0 и ДСИ2 = 0; иначе, если ДД2 = 1, то ДСИ1 = 0 и ДСИ2 = 1, если ДД2 = 0, то ДСИ1 = 1 и ДСИ2 = 0; опрос ДД2 организовать с

устранением дребезга контактов путем подсчета заданного числа совпадений значений сигнала.

9 Если ДД1 = 0, то ДСИ1 = 0 и ДСИ2 = 0; иначе, если ДД2 = 1, то ДСИ1 = 1 и ДСИ2 = 1, если ДД2 = 0, то ДСИ1 = 0 и ДСИ2 = 0; организовать периодический опрос ДД2 с применением таймера; период опроса 10 мс.

10 Если ДД1 = 1 и ДД2 = 1, то ДСИ1 = 0 и ДСИ2 = 0, если ДД1 = 0 и ДД2 = 0, то ДСИ1=1, ДСИ2 = 1, если ДД1 ≠ ДД2, то ДСИ1=1, ДСИ2 = 0; организовать периодический опрос ДД1 и ДД2 с применением таймера; период опроса 20 мс.

11 Если ДД2 = 1, то ДСИ1 = 1 и ДСИ2 = 1; иначе, если ДД1 = 1, то ДСИ1 = 1 и ДСИ2 = 0, если ДД1 = 0, то ДСИ1 = 0 и ДСИ2 = 1; опрос ДД1 организовать с устранением дребезга контактов путем введения временной задержки.

12 Если ДД1 = 1, то ДСИ1 = 0 и ДСИ2 = 0; иначе, если ДД2 = 1, то ДСИ1 = 0 и ДСИ2 = 1, если ДД2 = 0, то ДСИ1 = 1 и ДСИ2 = 0; опрос ДД2 организовать с устранением дребезга контактов путем подсчета заданного числа совпадений значений сигнала.

13 Если ДД1 = 0, то ДСИ1 = 0 и ДСИ2 = 0; иначе, по приходу на ДД2 положительного импульса длительностью менее 3 с, ДСИ1 = 1 и ДСИ2 = 1.

14 Если ДД2 = 1, то ДСИ1 = 0 и ДСИ2 = 0; иначе, по приходу на ДД1 отрицательного импульса длительностью более 4,5 с, ДСИ1 = 1 и ДСИ2 = 1.

15 Если после прихода на ДД1 положительного импульса длительностью более 3 с на ДД2 придет отрицательный импульс длительностью более 2,5 с, ДСИ1 = 1 и ДСИ2 = 1, иначе ДСИ1 = 0 и ДСИ2 = 0.

2.6 Пример выполнения работы

Разработать программу для учебного стенда СУ-МК, позволяющую отображать на ДСИ1 состояние ДД1 и на ДСИ2 состояние ДД2.

Блок-схема алгоритма решения задачи представлена на рисунке 2.4.

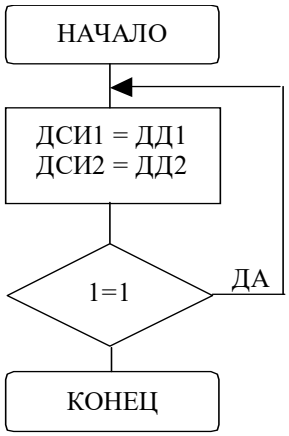


Рисунок 2.4 – Блок-схема алгоритма решения задачи

Листинг программы для решения задачи

```
#include <reg51.h>
#include <lab5.h> //Обязательно включайте в свои программы модуль lab5.h

sbit DD1 = P3^2;
sbit DD2 = P3^3;
sbit DSI1 = P3^5;
sbit DSI2 = P3^4;

void main (void)
{
 init_stend(); //перед началом выполнения своих действий необходимо проинициализировать
            //учебный стенд СУ-МК
 while (1)
{
  DSI1 = DD1;
  DSI2 = DD2;
 }
}
```

Контрольные вопросы

1 Каким образом организуется ожидание статического сигнала?

2 Каким образом организуется ожидание импульсного сигнала?

3 Каковы ограничения на длительность импульса? Как эти ограничения можно преодолеть?

4 В чем сущность явления дребезга контактов? Какие есть способы для преодоления этого явления?

5 Как реализуется способ устранения дребезга путем подсчета числа совпадающих значений сигнала?

6 Как реализуется способ устранения дребезга путем временной задержки?

7 Как организуется периодический опрос датчиков?

8 Как организуется опрос группы дискретных датчиков?

9 Как организуется вывод статических дискретных сигналов?

10 Как организуется вывод импульсных дискретных сигналов?

3 Лабораторная работа № 3. Разработка типовых программ управления

Цель работы: изучить связь между аналоговыми регуляторами и цифровыми фильтрами, формы представления цифровых фильтров, особенности их программной реализации, реализацию с помощью языка программирования; составить программу реализации заданного цифрового фильтра, используя язык программирования микроконтроллера, перевести ее в машинные коды, записать в память программ микроконтроллера и проверить работу, используя аналоговые входы и выходы.

3.1 Цифровые регуляторы

В непрерывных системах широко используются ПИД-регуляторы (пропорционально-интегрально-дифференциальные регуляторы) [2] (рисунок 3.1), которые представляются идеализированным уравнением

$$u(t) = K_p \cdot \left[x(t) + \frac{1}{T_I^x} \int_0^t x(\tau) d\tau + T_D^x \frac{dx(t)}{dt} \right],$$

где $x(t)$ – сигнал рассогласования между управляющим воздействием и реакцией объекта управления;

K_P – коэффициент усиления пропорционального канала;

T_I^x – постоянная времени сопрягающего полюса интегрального канала;

T_D^x – постоянная времени сопрягающего полюса дифференциального канала.

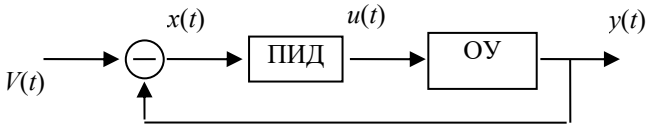


Рисунок 3.1 – Структура ПИД-регулятора

Для малых периодов дискретизации $T_ц$ уравнение может быть преобразовано в разностное без существенной потери в точности. Непрерывное интегрирование может быть представлено с помощью метода прямоугольников, или метода трапеций.

Используем метод прямоугольников для аппроксимации непрерывного интеграла и запишем ПИД-закон в дискретном виде:

$$u[n] = K_p \cdot \left[x[n] + \frac{T_Ц}{T_I^x} \sum_{i=0}^{n-1} x[i] + \frac{T_D^x}{T_Ц} \left(x[n] - x[n-1] \right) \right],$$

где n – номер дискретного отсчета.

В результате получен нерекуррентный (позиционный) алгоритм управления, который требует сохранения всех предыдущих значений сигнала ошибки $x[i]$, и в котором каждый раз заново вычисляется управляющий сигнал $u[n]$.

Для реализации программ закона регулирования на цифровой вычислительной машине (ЦВМ) более удобным является рекуррентный алгоритм. Он характеризуется тем, что для вычисления текущего значения сигнала $u[n]$ используется его предыдущее значение $u[n-1]$ и поправочный коэффициент, не требующий существенных вычислительных затрат.

$$u[n] = u[n-1] + b_0\,x[n] + b_1\,x[n-1] + b_2\,x[n-2], \tag{3.1}$$

где $b_0 = K_P \cdot \left(1 + \dfrac{T_D^x}{T_Ц}\right)$;

$b_1 = K_P \cdot \left(\dfrac{T_Ц}{T_I^x} - 1 - 2 \cdot \dfrac{T_D^x}{T_Ц}\right)$;

$b_2 = K_P \cdot \dfrac{T_D^x}{T_Ц}$.

Если для аппроксимации непрерывного интеграла использовать метод трапеций, то разностное уравнение будет иметь вид:

$$u[n] = K_p \cdot \left[x[n] + \frac{T_Ц}{T_I^x} \cdot \left(\frac{x[n] - x[0]}{2} + \sum_{i=0}^{n-1} x[i] \right) + \frac{T_D^x}{T_Ц} \left(x[n] - x[n-1] \right) \right].$$

Для рекуррентного соотношения выявляются отличия только для коэффициента b_0:

$$b_0 = K_P \cdot \left(1 + \frac{T_D^x}{T_Ц} + \frac{T_Ц}{2 \cdot T_D^x}\right).$$

Запишем рекуррентное соотношение (3.1) для изображений в z-домене:

$$U\,[z]\,(1 - z^{-1}) = (b_0 + b_1\,z^{-1} + b_2\,z^{-2})\,X\,[z],$$

и представим его в виде дискретной передаточной функции

$$W_{PID}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - z^{-1}}. \tag{3.2}$$

Анализ коэффициентов показывает, что если $b_2 = 0$, то получим ПИ-регулятор.

Если $b_0 = 0$, а $b_1 = (1 + b_2)$, то получим ПД-регулятор.

3.2 Алгоритмы программ цифровых фильтров

Существует три основных алгоритма программной реализации дискретных передаточных функций (z-ПФ) [3]:

1) непосредственный с двумя буферами и с одним буфером;
2) последовательный;
3) параллельный.

Дискретную передаточную функцию можно представить в любой из форм:

1) стандартная форма для дискретных ПФ

$$W(z) = \frac{X(z)}{Y(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + ... + b_m z^{-m}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + ... + a_k z^{-k}};$$

2) разложение на z-множители

$$W(z) = \frac{X(z)}{Y(z)} = \frac{K}{1 + d_1 z^{-1}} \frac{1 + e_2 z^{-2}}{1 + d_2 z^{-2}} \frac{1 + e_k z^{-k}}{1 + d_k z^{-k}};$$

3) разложение на z-элементарные дроби

$$W(z) = \frac{X(z)}{Y(z)} = \frac{P_1}{1 + d_1 z^{-1}} + \frac{P_2}{1 + d_2 z^{-2}} + \frac{P_k}{1 + d_k z^{-k}},$$

где e_i – нули z-ПФ;

d_i – полюса z-ПФ;

a_0 – не равно нулю;

P_i – коэффициенты разложения.

Разложения 2 и 3 делают параметры z-ПФ независимыми, позволяют контролировать ряд дополнительных фазовых координат: $x_1[n]$, $x_2[n]$, ..., $x_{k-1}[n]$; или $y_1[n]$, $y_2[n]$, ..., $y_k[n]$ – что удобно при отладке систем.

Последовательная структура 2 удобна при синтезе дискретной коррекции.

Параллельная структура 3 удобна для построения цифровых регуляторов.

Разложение z-ПФ на элементарные дроби 3 позволяет реализовать z-ПФ на параллельно работающих ЦВМ для повышения быстродействия.

Перечисленные факторы определяют выбор алгоритма программы для ЦВМ.

После разложений каждый из множителей в форме 2 или каждую из элементарных дробей в форме 3 следует представить в стандартной форме 1 (с отрицательными степенями оператора z). Переход к разностным уравнениям будет един. z-ПФ в форме 1 соответствует разностное уравнение (РУ)

$$y[n] = \frac{(\sum_{i=0}^{m} b_i x[n-i] - \sum_{j=1}^{k} a_j y[n-j])}{a_0},$$

по которому и составляется программа. Поскольку текущее значение выходной координаты $y[n]$ рассчитывается по предыдущим значениям $y[n-1]$, $y[n-2]$, $y[n-k]$ – данное РУ называется рекурсивным.

Изобразим структурную схему цифрового фильтра для этого уравнения (рисунок 3.2). Ее можно преобразовать, объединив два буфера (рисунок 3.3). Цепочки элементов z^{-1} в программах будут соответствовать буферам из ячеек памяти, данные в которых сдвигаются на каждом такте дискретизации.

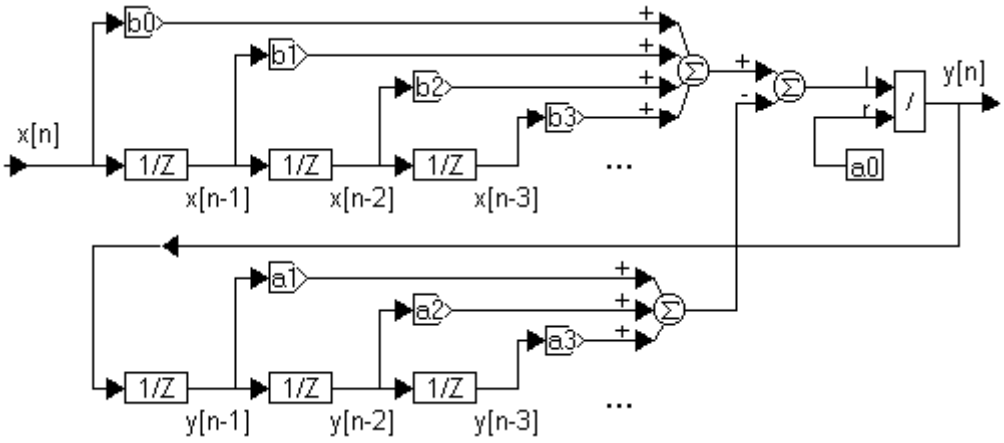


Рисунок 3.2 – Структурная схема по алгоритму 1 с двумя буферами. Условие физической реализуемости – $a_0 \neq 0$

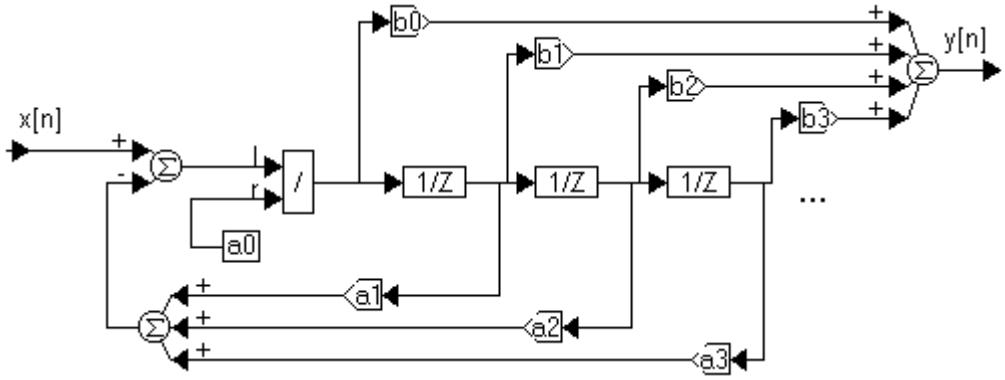


Рисунок 3.3 – Структурная схема по алгоритму 1 с одним буфером. Условие физической реализуемости – $a_0 \neq 0$

3.3 Варианты индивидуальных заданий к лабораторной работе

Разработать программу для учебного стенда СУ-МК, позволяющую вырабатывать управляющее воздействие в соответствии с заданными параметрами ПИД-регулятора. Входное воздействие считывается с одного из аналоговых датчиков АД1…АД3 через АЦП. Управляющее воздействие следует выводить

на выход аналогового сигнала через ЦАП. Вывести числовые значения, соответствующие входному воздействию и управляющему воздействию на ССИ. Предусмотреть возможность генерации тестового воздействия в соответствии с состоянием Д1.

1 ПИД-регулятор со следующими параметрами: K_P = 0,4; $T_Ц$ = 0,2 с; T_D = 0,8 с; T_I = 12 с. Входное воздействие снимается с аналогового датчика АД1.

2 ПИ-регулятор со следующими параметрами: K_P = 0,3; $T_Ц$ = 0,4 с; T_I = 28 с. Входное воздействие снимается с аналогового датчика АД1.

3 ПИД-регулятор со следующими параметрами: K_P = 0,8; $T_Ц$ = 0,1 с; T_D = 0,4 с; T_I = 24 с. Входное воздействие снимается с аналогового датчика АД2.

4 ПД-регулятор со следующими параметрами: K_P = 1; $T_Ц$ = 0,4 с; T_D = 1,2 с. Входное воздействие снимается с аналогового датчика АД1.

5 ПИ-регулятор со следующими параметрами: K_P = 1; $T_Ц$ = 0,5 с; T_I = 21 с. Входное воздействие снимается с аналогового датчика АД3.

6 ПИД-регулятор со следующими параметрами: K_P = 1; $T_Ц$ = 0,4 с; T_D = 1 с; T_I = 10 с. Входное воздействие снимается с аналогового датчика АД3.

7 ПД-регулятор со следующими параметрами: K_P = 0,8; $T_Ц$ = 0,5 с; T_D = 0,8 с. Входное воздействие снимается с аналогового датчика АД2.

8 ПИ-регулятор со следующими параметрами: K_P = 0,4; $T_Ц$ = 0,4 с; T_I = 45 с. Входное воздействие снимается с аналогового датчика АД1.

9 ПИД-регулятор со следующими параметрами: K_P = 0,5; $T_Ц$ = 0,5 с; T_D = 1,2 с; T_I = 36 с. Входное воздействие снимается с аналогового датчика АД1.

10 ПД-регулятор со следующими параметрами: K_P = 0,3; $T_Ц$ = 0,2 с; T_D = 0,5 с. Входное воздействие снимается с аналогового датчика АД2.

11 ПИ-регулятор со следующими параметрами: K_P = 0,5; $T_Ц$ = 0,6 с; T_I = 60 с. Входное воздействие снимается с аналогового датчика АД2.

12 ПД-регулятор со следующими параметрами: K_P = 0,5; $T_Ц$ = 0,6 с; T_D = 0,9 с. Входное воздействие снимается с аналогового датчика АД3.

13 ПИ-регулятор со следующими параметрами: K_P = 0,7; $T_Ц$ = 0,2 с; T_I = 12 с. Входное воздействие снимается с аналогового датчика АД2.

14 ПИД-регулятор со следующими параметрами: K_P = 1; $T_Ц$ = 0,6 с; T_D = 0,9 с; T_I = 45 с. Входное воздействие снимается с аналогового датчика АД2.

15 ПД-регулятор со следующими параметрами: K_P = 0,2; $T_Ц$ = 0,1 с; T_D = 0,4 с. Входное воздействие снимается с аналогового датчика АД1.

3.4 Пример выполнения работы

Разработать программу для учебного стенда СУ-МК, реализующую цифровой ПИД-регулятор со следующими параметрами: K_P = 0,7; $T_Ц$ = 0,3 с; T_D = 0,9 с; T_I = 10 с. Входное воздействие снимается с аналогового датчика АД1 через АЦП. Вывести числовые значения, соответствующие входному воздействию и управляющему воздействию на ССИ. Предусмотреть возможность генерации тестового воздействия, если Д1 = 0. Управляющее воздействие выводить на выход аналогового сигнала через ЦАП. Предусмотреть масштабирование аналогового сигнала таким образом, чтобы величина воздействия

в +508 единиц соответствовала максимальному уровню выходного аналогового сигнала, величина воздействия в –508 единиц соответствовала нулевому уровню выходного аналогового сигнала, величина воздействия в 0 единиц соответствовала уровню выходного аналогового сигнала 0,5 максимального.

Рассчитаем коэффициенты дискретной передаточной функции:

$$b_0 = 0{,}7 \cdot \left(1 + \frac{0{,}9}{0{,}3}\right) = 2{,}8;$$

$$b_1 = 0{,}7 \cdot \left(\frac{0{,}3}{10} - 1 - 2\frac{0{,}9}{0{,}3}\right) = -4{,}879;$$

$$b_2 = 0{,}7 \cdot \frac{0{,}9}{0{,}3} = 2{,}1.$$

Для реализации используем алгоритм с двумя буферами. Для ПИД-регулятора (3.2) глубина буфера входного воздействия равна двум (необходимо фиксировать два предыдущих значения входного воздействия); глубина буфера выходного воздействия равна единице (необходимо фиксировать одно предыдущее значения выходного воздействия).

Текст программы:

```
#include<reg51.h>
#include<lab13.h>
#include<vv55.h>
#include<dac.h>
#include<i2c.h>
#include<adc.h>
#include<ssi.h>

sbit at 0xB3 DD_1;

float b0 = 2.8, b1 = -4.879, b2 = 2.1;     //Коэффициенты дискретной ПФ
float xz_1, xz_2, yz_1;                    //Буферы входного и управляющего
воздействия
void pause(void){                  //Пауза для реализации
unsigned int i1, i2;               //времени дискретизации
 for (i2 = 0; i2 < 2; i2++){               //Тц = 0,3 с
  for (i1 = 0; i1 < 64500; i1++){}
 }
}

float y_zW( unsigned int x)                //Функция расчета управляющего
воздействия
{                              //ПИД-регулятора
```

```
 float y;
 y=x*b0+xz_1*b1+xz_2*b2 + yz_1;
 xz_2=xz_1; xz_1=x;
 yz_1=y;
 return y;
}

void main (void)
{
 unsigned char step;
 signed int y_dac;
 signed int x;
 float fy;
 init_stend();          //перед начало выполнения своих действий необходимо проинициализировать
          //учебный стенд СУ-МК
 init_ssi ();    //Инициализация ССИ
 clr_ssi ();     //Очистка ССИ
 i2c_init();     //Инициализация шины i2c
 adc_init();     //Инициализация АЦП
 adc_select_channel(1);    //Выбор канала АЦП
 step = 0;
 xz_2=xz_1 = 0;      //Инициализация буфера
 yz_1 = 0;
 while (1)      //Бесконечный цикл
 {
  if (DD_1 == 0) //Если ДД1 = 0
  {
   if (++ step > 20) step = 0;   //то формируется тестовое воздействие
   if (step < 10) {x = 0;} else {x = 127;} //в виде ступенчатой функции амплитудой 0,5 макс
  }
  else {x = adc_read();}     //иначе считывается входное воздействие с АЦП
  fy = y_zW(x);          //Расчет управляющего воздействия
  if (fy>510) {y_dac = 508;}     //Ограничение управляющего воздействия по уровню
  else
  {
   if (fy<-510) {y_dac = -508;}
   else y_dac = fy;}
  ssi_write_4dig (x, 4); //Вывод входного воздействия на ССИ
  ssi_write_s3dig (y_dac , 0); //Вывод значения управляющего воздействия на ССИ
  set_dac(y_dac / 4 + 127);      //Установка ЦАП в соответствие с выходным воздействием
```

```
  pause();                    //Пауза 0,3 с
 }
}
```

Контрольные вопросы

1 Где применяются цифровые системы управления?
2 Какую роль играют АЦП и ЦАП в цифровых системах управления?
3 Опишите структуру ПИД-регулятора.
4 Какие существуют схемы алгоритмов реализации цифровых фильтров?
5 Опишите преимущества и недостатки последовательной и параллельной схем построения алгоритмов цифровой фильтрации.

4 Лабораторная работа № 4. Разработка и отладка программы управления в реальном времени

Цель работы: изучить особенности программной и аппаратной реализации временных функций, режимы работы и порядок программирования таймеров микроконтроллера, реализацию временных функций с помощью языка программирования С; составить программу заданной временной функции, перевести ее в машинные коды, записать в память программ микроконтроллера и выполнить.

4.1 Реализация функций времени

Процедура реализации временной задержки использует метод программных циклов. При этом в некоторый рабочий регистр загружается число, которое затем в каждом проходе цикла уменьшается на единицу. Так продолжается до тех пор, пока содержимое рабочего регистра не станет равным нулю, что интерпретируется программой как момент выхода из цикла. Время задержки при этом определяется числом, загруженным в рабочий регистр, и временем выполнения команд, образующих программный цикл.

Предположим, что в управляющей программе необходимо реализовать временную задержку 99 мкс. Фрагмент программы, реализующей временную задержку, требуется оформить в виде подпрограммы, т. к. предполагается, что основная управляющая программа будет производить к ней многократные обращения для формирования выходных импульсных сигналов, длительность которых кратна 99 мкс:

```
DELAY:   MOV R2, #X          ;(R2)←X
COUNT:   DJNZ R2, COUNT;декремент R2 и цикл, если не нуль
RET              ;возврат
```

Для получения требуемой временной задержки необходимо определить число X, загружаемое в рабочий регистр. Определение числа X выполняется на основе расчёта времени выполнения команд, образующих данную подпрограмму. При этом необходимо учитывать, что команды MOV и RET выполняются однократно, а число повторений команды DJNZ равно числу X. Кроме того, обращение к подпрограмме временной задержки осуществляется по команде CALL DELAY, время исполнения которой также необходимо учитывать при подсчете временной задержки. В описании команд микроконтроллера указывается, за сколько машинных циклов (МЦ) исполняется каждая команда. На основании этих данных определяется суммарное число машинных циклов в подпрограмме: CALL – 2 МЦ, MOV – 1 МЦ, DJNZ – 2 МЦ, RET – 2 МЦ.

При тактовой частоте 11,059 МГц каждый машинный цикл выполняется за 1 мкс. Таким образом, подпрограмма выполняется за время 2 + 1 + 2X + 2 = = 5 + 2X мкс. Для реализации временной задержки 99 мкс число X = (99 – 5)/2 = 47.

В данном случае при загрузке в регистр R2 числа 47 требуемая временная задержка (99 мкс) реализуется лишь приблизительно. Для более точной подстройки в подпрограмму могут быть включены команды NOP, время выполнения каждой из которых равно 1 мкс.

Минимальная временная задержка, реализуемая подпрограммой DELAY, составляет 7 мкс (X = 1). Временную задержку меньшей длительности программным путем можно реализовать, включая в программу цепочки команд NOP.

Максимальная длительность задержки, реализуемая подпрограммой DELAY, составляет 515 мкс (X = 255).

На языке С такая подпрограмма выглядит следующим образом:

```
void delay (void)
{
 char register x;
 x = 47;
 while (--x!=0);
}
```

Число, загружаемое в переменную x для реализации временной задержки на языке С, определить сложнее. Дело в том, что временная задержка, задаваемая операторами на языке С будет зависеть от кода, который получается в результате компиляции. При использовании среды ProView32, код можно просмотреть в режиме отладки. Для данной подпрограммы код будет следующим:

```
                MOV    R7,#064H
?WHILE1:        DEC    R7
                MOV    A,R7
                JNZ    ?WHILE1
                RET
```

Для реализации задержки большей длительности можно рекомендовать увеличить тело цикла включением дополнительных команд или использовать метод вложенных циклов. Так, например, если в подпрограмму DELAY перед командой DJNZ вставить дополнительно две команды NOP, то максимальная задержка составит 5 + X(2 + 1) = 5+3 · 255 = 770 мкс (т. е. почти в 1,5 раза больше).

Задержка большой длительности может быть реализована методом вложенных циклов. Числа X и Y выбираются из соотношения T = 2 + 1 + X(1 + + 2Y + 2) + 2, где T – реализуемый временной интервал в микросекундах. Максимальный временной интервал, реализуемый таким способом, при X = Y = 255 составляет 130.82 мс, т. е. приблизительно 0,13 с.

В качестве примера рассмотрим подпрограмму, реализующую временную задержку 100 мс:

```
DELAY:   MOV R1, #195        ;загрузка X
LOOPEX:  MOV R2, #254        ;загрузка Y
LOOPIN:  DJNZ R2, LOOPIN     ;декремент R2 и внутренний цикл,
                             ;если (R2)≠0
         DJNZ R1, LOOPEX     ;декремент R1 и внешний цикл,
                             ;если (R1)≠0
         MOV R3, #174        ;точная подстройка
LOOPAD:  DJNZ R3, LOOPAD     ;временной задержки
         NOP                 ;задержки
         RET
```

Здесь два вложенных цикла реализуют временную задержку длительностью 5 + 195(3 + 2·254) = 99 650 мкс, а дополнительный цикл LOOPAD и команда NOP реализует задержку 350 мкс и тем самым обеспечивает точную настройку временного интервала.

На языке С такая процедура реализуется проще:

```
void delay100ms (void)
{
 unsigned short register x;
 x = 11514;
 while (--x!=0);
}
```

Здесь длительность задержки задается числом в переменной x типа unsigned short. Переменная этого типа может принимать значения от 0 до 65535. Длительность задержки в этом примере вычисляется следующим образом: 9 + 8·X. Для реализации задержки в 100 мс это число должно быть 11514 на частоте 11,059 МГц.

Из рассмотренного примера видно, что секунда является очень большим интервалом времени по сравнению с тактовой частотой микроконтроллера.

Такие задержки сложно реализовать методом вложенных циклов, поэтому их обычно набирают из точно подстроенных задержек меньшей длительности. Например, задержку в 1 с можно реализовать десятикратным вызовом подпрограммы, реализующей задержку 100 мс:

```
void delay1s (void)
{
 unsigned char i;
 for (i = 10, i>0, i--)
 {
  delay100ms();
 }
}
```

Погрешность подпрограммы составляет 21 мкс. Для очень многих применений это достаточно высокая точность, хотя реализованные на основе этой программы часы астрономического времени за сутки «убегут» примерно на 1,8 с.

В составе микроконтроллера [1] имеются регистровые пары с символическими именами TH0, TL0, TH1, TL1, на основе которых функционируют два независимых программно-управляемых 16-битных таймера/счётчика событий (T/C0 и T/C1). При работе в качестве таймера содержимое T/C инкрементируется в каждом машинном цикле, т. е. через каждые 12 периодов резонатора. При работе в качестве счётчика содержимое T/C инкрементируется под воздействием перехода из 1 в 0 внешнего входного сигнала, подаваемого на соответствующий (T0, T1) вход микроконтроллера. Опрос сигналов выполняется в каждом машинном цикле. Т. к. на распознавание перехода требуется два машинных цикла, то максимальная частота подсчёта входных сигналов равна 1/24 частоты резонатора. На длительность периода входных сигналов ограничений сверху нет. Для гарантированного прочтения входного считываемого сигнала он должен удерживать значение 1 как минимум в течение одного машинного цикла.

Для управления режимами работы и для организации взаимодействия таймеров с системой прерывания используются два регистра специальных функций TMOD и TCON, описание которых приводится в таблицах 4.1 и 4.3.

Для обоих T/C режимы работы 0, 1 и 2 одинаковы. Режимы 3 для T/C0 и T/C1 различны.

Установка бита GATE в 1 позволяет использовать таймер для измерения длительности импульсного сигнала, подаваемого на вход запроса прерывания.

Таблица 4.1 – Регистр режима работы таймера/счётчика

Символ	Разряд	Имя и назначение
GATE	TMOD.7 для T/C1 TMOD.3 для T/C0	Управление блокировкой. Если бит установлен, то таймер/счётчик «x» разрешен до тех пор, пока на входе «INT x» высокий уровень и бит управления «TRx» установлен. Если бит сброшен, то T/C разрешается, как только бит управления «TRx» устанавливается
C/T	TMOD.6 для T/C1 TMOD.2 для T/C0	Бит выбора режима таймера или счётчика событий. Если бит сброшен, то работает таймер от внутреннего источника сигналов синхронизации. Если бит установлен, то работает счётчик от внешних сигналов на входе «Tx»
M1	TMOD.5 для T/C1 TMOD.1 для T/C0	Режим работы (таблица 4.2)
M0	TMOD.4 для T/C1 TMOD.0 для T/C0	

Таблица 4.2 – Режимы работы таймера/счётчика

M1	M0	Режим работы
0	0	«TLx» работает как 5-битный предделитель
0	1	16-битный таймер/счётчик. «THx» и «TLx» включены последовательно
1	0	8-битный автоперезагружаемый таймер/счётчик. «THx» хранит значение, которое должно быть перезагружено в «TLx» каждый раз по переполнению
1	1	Таймер/счётчик 1 останавливается. Таймер/счётчик 0: TL0 работает как 8-битный таймер/счётчик, и его режим определяется управляющими битами таймера 0. TH0 работает только как 8-битный таймер, и его режим определяется управляющими битами таймера 1

Таблица 4.3 – Регистр управления/статуса таймера

Символ	Разряд	Имя и назначение
TF1	TCON.7	Флаг переполнения таймера 1. Устанавливается аппаратно при переполнении таймера/счётчика. Сбрасывается при обслуживании прерывания аппаратно
TR1	TCON.6	Бит управления таймера 1. Устанавливается/сбрасывается программой для пуска/останова
TF0	TCON.5	Флаг переполнения таймера 0. Устанавливается аппаратно. Сбрасывается при обслуживании прерывания
TR0	TCON.4	Бит управления таймера 0. Устанавливается/сбрасывается программой для пуска/останова таймера/счётчика
IE1	TCON.3	Флаг фронта прерывания 1. Устанавливается аппаратно, когда детектируется срез внешнего сигнала INT1. Сбрасывается при обслуживании прерывания
IT1	TCON.2	Бит управления типом прерывания 1. Устанавливается/сбрасывается программно для спецификации запроса INT1 (срез/низкий уровень)
IE0	TCON.1	Флаг фронта прерывания 0. Устанавливается по срезу сигнала INT0. Сбрасывается при обслуживании прерывания
IT0	TCON.0	Бит управления типом прерывания 0. Устанавливается/сбрасывается программно для спецификации запроса INT0 (срез/низкий уровень)

Режим 0. Перевод любого Т/С в этот режим делает его 8-разрядным таймером, на вход которого подключен 5-битный предделитель частоты на 32. В этом режиме таймерный регистр имеет разрядность 13 бит. При переходе из состояния «все единицы» в состояние «все нули» устанавливается флаг прерывания от таймера TF1. Входной синхросигнал таймера 1 разрешен (поступает на вход Т/С), когда управляющий бит TR1 установлен в 1 и либо управляющий бит GATE (блокировка) равен 0, либо на внешний вход запроса прерывания INT1 поступает уровень 1.

Режим 1. Работа любого Т/С в этом режиме такая же, как и в режиме 0, за исключением того, что таймерный регистр имеет разрядность 16 бит.

Режим 2. В этом режиме работа организована таким образом, что переполнение (переход из состояния «все единицы» в состояние «все нули») 8-битного счётчика TL1 приводит не только к установке флага TF1, но и автоматически перезагружает в TL1 содержимое старшего байта (TH1) таймерного регистра, которое предварительно было задано программным путем. Перезагрузка оставляет содержимое TH1 неизменным. В режиме 2 Т/С0 и Т/С1 работают совершенно одинаково.

Режим 3. В этом режиме Т/С0 и Т/С1 работают по-разному. Т/С1 сохраняет неизменным своё текущее содержимое. Иными словами, эффект такой же, как и при сбросе управляющего бита TR1 в нуль. В этом режиме TL0 и TH0 функционируют как два независимых 8-битных счётчика. Работу TL0 определяют управляющие биты Т/С0 (C/T, GATE, TR0), входной сигнал INT0 и флаг переполнения TF0. Работу TH0, который может выполнять только функции таймера (подсчёт машинных циклов микроконтроллера), определяет управляющий бит TR1. При этом TH0 использует флаг переполнения TF1.

Внешние прерывания INT0 и INT1 могут быть вызваны уровнем или переходом сигнала из 1 в 0 на входах микроконтроллера в зависимости от значений управляющих битов IT0 и IT1 в регистре TCON. От внешних прерываний устанавливаются флаги IE0 и IE1 в регистре TCON, которые инициируют вызов соответствующей подпрограммы обслуживания прерывания. Сброс этих флагов выполняется аппаратно только в том случае, если прерывание было вызвано по переходу (срезу) сигнала. Если же прерывание вызвано уровнем входного сигнала, то сбросом флага IE управляет соответствующая подпрограмма обслуживания прерывания путем воздействия на источник прерывания с целью снятия им запроса.

Флаги запросов прерывания от таймеров TF0 и TF1 сбрасываются автоматически при передаче управления подпрограмме обслуживания. Флаги запросов прерывания RI и TI устанавливаются UART аппаратно, но сбрасываться должны программой. Прерывания могут быть вызваны или отменены программой, т. к. все перечисленные флаги программно доступны.

В блоке регистров специальных функций есть два регистра, предназначенных для управления режимом прерываний и уровнями приоритета. Форматы этих регистров, имеющих символические имена IE и IP описаны в таблицах 4.4 и 4.5 соответственно.

Возможность программной установки/сброса любого управляющего бита в этих двух регистрах делает систему прерываний исключительно гибкой.

Флаги прерываний опрашиваются в каждом машинном цикле. Ранжирование прерываний по приоритету выполняется в течение следующего машинного цикла. Система прерываний сформирует аппаратно вызов LCALL соответствующей подпрограммы обслуживания, если она не заблокирована одним из условий:

– в данный момент обслуживается запрос прерывания равного или более высокого уровня приоритета;

– текущий машинный цикл – не последний в цикле выполняемой команды;

– выполняется команда RETI или любая команда, связанная с обращением к регистрам IE или IP.

Таблица 4.4 – Регистр масок прерывания IE

Символ	Разряд	Имя и назначение
EA	IE.7	Снятие блокировки прерываний. Сбрасывается программно для запрета всех прерываний независимо от состояний IE4-IE0
–	IE.6, 5	Не используются
ES	IE.4	Бит разрешения прерывания от UART. Установка/сброс программой для разрешения/запрета прерываний от флагов TI, RI
ET1	IE.3	Бит разрешения прерывания от таймера 1. Установка/сброс программой для разрешения/запрета прерываний от таймера 1
EX1	IE.2	Бит разрешения внешнего прерывания 1. Установка/сброс программой для разрешения/запрета прерываний
ET0	IE.1	Разрешение прерывания от таймера 0. Работает аналогично IE.3
EX0	IE.0	Разрешения внешнего прерывания 0. Работает аналогично IE.2

Таблица 4.5 – Регистр приоритетов прерывания IP

Символ	Разряд	Имя и назначение
–	IP.7-5	Не используются
PS	IP.4	Бит приоритета UART. Установка/сброс программой для назначения прерыванию от UART высшего/низшего приоритета
PT1	IP.3	Бит приоритета таймера 1. Установка/сброс программой для назначения прерыванию от таймера 1 высшего/низшего приоритета
PX1	IP.2	Бит приоритета внешнего прерывания 1. Установка/сброс программой для назначения прерыванию INT1 высшего/низшего приоритета
PT0	IP.1	Бит приоритета таймера 0. Работает аналогично IP.3
PX0	IP.0	Приоритет внешнего прерывания 0. Работает аналогично IP.2

Если флаг прерывания был установлен, но по одному из перечисленных условий не получил обслуживания и к моменту окончания блокировки уже был сброшен, то запрос прерывания теряется.

По аппаратно сформированному коду команды LCALL система прерывания помещает в стек содержимое программного счётчика PC и загружает в

PC-адрес вектора прерывания соответствующей подпрограммы обслуживания. По этому адресу должна быть расположена команда безусловного перехода JMP к начальному адресу подпрограммы обслуживания прерывания. Эта подпрограмма в случае необходимости должна начинаться командами записи в стек PUSH слова состояния программы PSW, аккумулятора A, расширителя аккумулятора B, указателя данных DPTR и т. д. и заканчиваться командами восстановления из стека POP. Подпрограммы обслуживания прерывания обязательно завершаются командой RETI, по которой в программный счётчик перезагружается из стека сохранённый адрес возврата в основную программу. Команда RET также возвращает управление, но при этом не снимает блокировку прерывания.

Недостатком программного способа реализации временной задержки является нерациональное использование ресурсов микроконтроллера: во время формирования задержки он практически простаивает, т. к. не может решать никаких задач управления объектом. В то же время аппаратурные средства позволяют реализовать временные задержки на фоне основной программы работы.

На вход таймера/счетчика (T/C) могут поступать сигналы синхронизации с частотой 1/12 частоты тактового генератора микроконтроллера (T/C в режиме таймера) или сигналы от внешнего источника (T/C в режиме счетчика). Оба эти режима могут быть использованы для формирования задержек. Если использовать T/C в режиме таймера полного формата (16 бит), то можно получить задержки в диапазоне 1…65 536 мкс на частоте кварца 12 МГц.

В качестве примера рассмотрим организацию временной задержки длительностью 50 мс:

```
#include <reg51.h>
#define PERIOD_hi = 19457 >> 8;
#define PERIOD_lo = 19457 - PERIOD_hi<<8;

void timer0() interrupt 1 using 2  //Функция обработки прерывания таймера
{
  TH0 = PERIOD_hi;
  TL0 = PERIOD_lo;

  //Периодические действия

}

void main(void)
{

//Настройка таймера
  TH0 = PERIOD_hi;  //Установка периода работы таймера
  TL0 = PERIOD_lo;
```

```
  TMOD = TMOD | 0x01;      //Выбор режима 1
  TR0 = 1;                 // Запуск таймера
  ET0 = 1;                 // Разрешение прерываний
  EA = 1;                  // Глобальное разрешение прерываний

…
}
```

В задачах управления часто возникает необходимость измерения промежутка времени между двумя событиями. Обычно события в объекте управления представляются сигналами от двоичных датчиков. Считая событиями фронт и спад импульса, можно определять временные характеристики импульсных сигналов: частота, длительность, период и скважность.

Для определения частоты сигналов осуществляется подсчет количества импульсов за определенное время (рисунок 4.1).

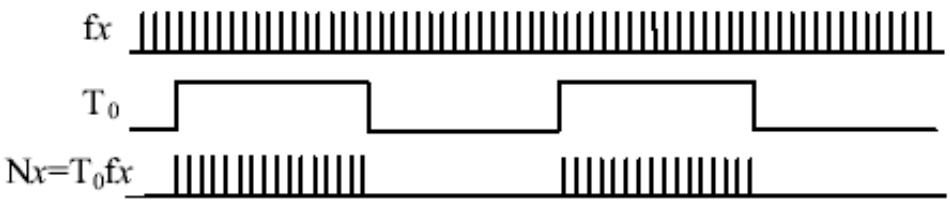


Рисунок 4.1 – Определение частоты сигнала

Для определения частоты сигналов при помощи микроконтроллеров MCS-51 используют входы внешних прерываний INT0 и INT1. Прерывания могут срабатывать не только по уровню, но и по фронту сигнала. При этом временные интервалы можно формировать таймерами.

Для определения длительности импульса используют обратный подход – подсчет количества импульсов известной частоты за время импульса (рисунок 4.2).

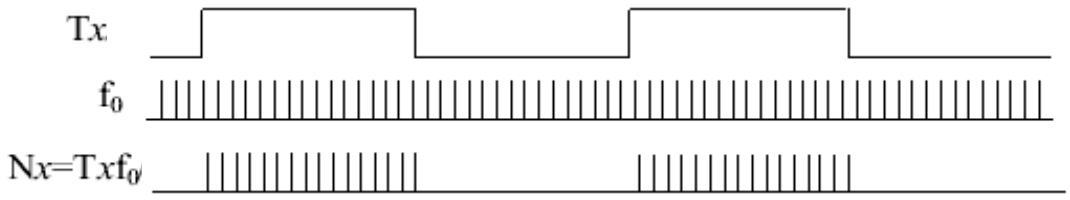


Рисунок 4.2 – Измерение длительности импульса

Простейшим способом измерения длительности импульса является программный. Для обнаружения событий (фронт и спад импульсного сигнала) в этом случае используются типовые процедуры ожидания импульсного сигнала, а отсчёт времени ведется программным способом. Для

«положительного» импульсного сигнала, поступающего на вход T0, программа измерения его длительности будет иметь вид:

```
unsined short count;
…

while (!T0);
while (T0)
{
 count++;
}
…
```

После выхода из процедуры содержимое счетчика count пропорционально длительности импульса.

Для нормальной работы этой программы необходимо, чтобы обращение к ней производилось в моменты, когда на входе T0 присутствует сигнал нулевого уровня.

Для измерения длительности сигнала может быть использован таймер [3]. Особенно эффективно использование для этой цели таймера в MCS-51, имеющего вход разрешения счёта (альтернативная функция входа INT). Измеряемый сигнал можно, например, подавать на вход INT0, а измерение длительности при этом будет выполняться в T/C0.

При необходимости измерения временных интервалов большей длительности можно программным способом подсчитывать число переполнений от таймера, т. е. расширять его разрядность за счет рабочего регистра или ячейки резидентной памяти данных.

Измерение частоты импульсов также можно производить с использованием таймера-счетчика. При этом на один внешний вход, например T1, необходимо подавать интересующую нас последовательность импульсов, а второй счетчик использовать для формирования интервала времени подсчета количества импульсов. В этом случае максимальная частота следования импульсов должна быть меньше 1/24 частоты резонатора. Подсчет количества импульсов за определенный период времени можно осуществить и программным путем. Но максимальная частота следования импульсов значительно снижается.

4.2 Варианты индивидуальных заданий к лабораторной работе

Разработать программу для учебного стенда СУ-МК, позволяющую выполнять следующие действия.

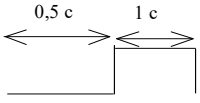
1 Определить длительность импульса с дискретностью 1 с от дискретного датчика ДД1 и отобразить ее на ССИ; использовать таймер/счетчик.

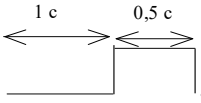
2 Определить количество импульсов, приходящих на ДД1 за 100 мс, и отобразить их на ССИ; использовать таймер/счетчик.

3 Определить частоту импульсов, приходящих на ДД1 в герцах, и отобразить ее на ССИ; использовать таймер/счетчик.

4 Программно определить длительность импульсов на ДД2 с дискретностью 100 мс и отобразить на ССИ.

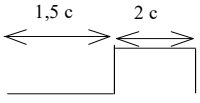
5 Если ДД1 = 0, то ДСИ1 = 0; если ДД1 = 1, то если ДД2 = 0, то на ДСИ1 вывести сигнал

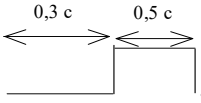


, иначе сигнал


.

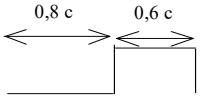
6 Если ДД1 = 0, то ДСИ1 = 1; если ДД1 = 1, то если ДД2 = 0, то на ДСИ1 вывести сигнал

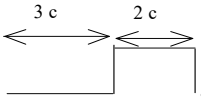


, иначе сигнал


.

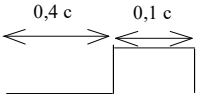
7 Если ДД1 = 0, то ДСИ2 = 0; если ДД1 = 1, то если ДД2 = 0, то на ДСИ2 вывести сигнал

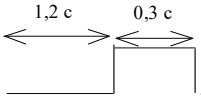


, иначе сигнал


.

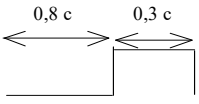
8 Если ДД1 = 0, то ДСИ2 = 1; если ДД1 = 1, то если ДД2 = 0, то на ДСИ2 вывести сигнал

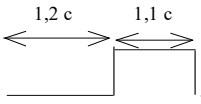


, иначе сигнал


.

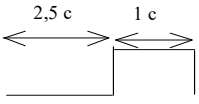
9 Если ДД1 = 0, то ДСИ2 = 0; если ДД1 = 1, то если ДД2 = 0, то на ДСИ2 вывести сигнал

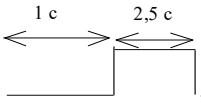


, иначе сигнал


.

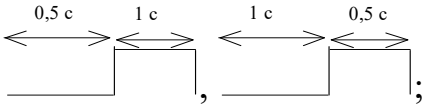
10 Если ДД1 = 0, то ДСИ1 = 1; если ДД1 = 1, то если ДД2 = 0, то на ДСИ1 вывести сигнал



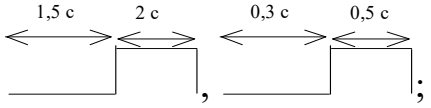
, иначе сигнал


.

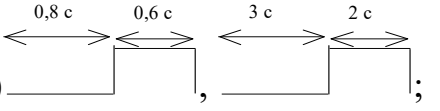
11 Если ДД1 = 1 и ДД2 = 1, то на ДСИ1 и на ДСИ2 вывести следующие последовательности, сформированные при помощи таймера:

а) 

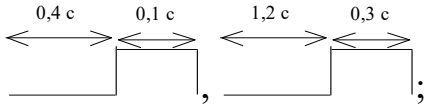
;

б) 

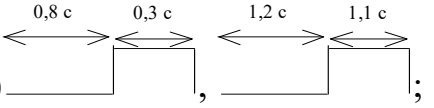
;

в) 

;

г) 

;

д) 

;

4.3 Пример выполнения работы

Разработать программу для учебного стенда СУ-МК, позволяющую определять длительность импульса с дискретностью 100 мс от дискретного датчика ДД2 и отображать ее на ССИ.

Если ДД1 = 1, то на ДСИ1 вывести меандр с периодом 2 с; если ДД1 = 0, то на ДСИ1 вывести меандр с периодом 1 с. Задержки сформировать программно.

Для подсчета длительности импульса выгодно воспользоваться режимом работы таймера/счетчика с разрешением счета по высокому уровню сигнала на входах INTx. Так как таймер срабатывает каждые 1,08 мкс при частоте 11,059 МГц, то для подсчета больших периодов, порядка 1 с, необходимо воспользоваться программным подсчетом количества переполнений счетчика.

Текст программы:

```
#include<reg51.h>
#include<lab8.h>
#include<ssi.h>


#define PERIOD_hi  19457 >> 8                  //Период работы таймера для обеспечения
#define PERIOD_lo  19457 - PERIOD_hi<<8//50 мс интервалов

sbit DD1 = P3^3;
sbit DSI1 = P3^5;
sbit DSI2 = P3^4;

unsigned int count50ms; //Количество 50 мс промежутков времени, вырабатываемых таймером

void timer1 (void) interrupt 3 using 2    //Обработка прерываний от таймера 1
{
 TH1 = PERIOD_hi;
 TL1 = PERIOD_lo;
 count50ms++; //Увеличиваем каждые 50 мс
}
void int1 (void) interrupt 2 using 2                  //Обработка прерываний от входа INT1 (ДД2)
{                                  //обрабатывается задний фронт сигнала
 if (count50ms > 2)
 {
  ssi_write_word(count50ms / 2);
  count50ms = 0;
 }
```

```
}
void delay100ms (void)  //Задержка 100 мс
{
 unsigned short register x;
 x = 11514;
 while (--x!=0);
}
void delay1s (void) //Задержка 1 с
{
  unsigned char i;
  for (i = 10; i>0; i--)
  {
    delay100ms();
  }
}
void delay05s (void) //Задержка 0.5 с
{
  unsigned char i;
  for (i = 5; i>0; i--)
  {
    delay100ms();
  }
}
void main (void)
{
  init_stend();          //перед начало выполнения своих действий необходимо проинициализировать
                 //учебный стенд СУ-МК
  init_SSI ();   //Инициализация ССИ

  IT1 = 1; //Прерывание от внешнего источника INT1 (ДД2) по срезу
  TH1 = PERIOD_hi;     //Загрузка первоначальных значений регистра счетчика
  TL1 = PERIOD_lo;  //для реализации интервала 50 мс
  TMOD &= 0x0F; //Настройка таймера 1
  TMOD |= 0x90; //GATE = 1, режим 1
  TR1 = 1;  //Запуск таймера 1
  ET1 = 1;  //Разрешение прерываний от таймера 1
  EX1 = 1;  //Разрешение прерывания от внешнего источника
  EA = 1;   //Глобальное разрешение прерываний
  while (1) //Обработка состояния датчика ДД1
  {
   if (DD1)
   {
    DSI2 = 1;
```

```
    DSI1 = ~DSI1;
    delay1s();
   }
   else
   {
    DSI2 = 0;
    DSI1 = ~DSI1;
    delay05s();
   }
  }
}
```

Контрольные вопросы

1 Как программно формируются задержки малой длительности?
2 Как программно формируются задержки большой длительности?
3 Использование таймеров для получения задержки.
4 Как определяется частота импульсного сигнала?
5 Как определяется длительность импульса?
6 Использование таймеров/счетчиков для определения длительности импульсов и частоты сигналов.

Список литературы

1 **Беккер, В. Ф.** Технические средства автоматизации. Интерфейсные устройства и микропроцессорные средства : учебное пособие / В. Ф. Беккер. – 2-е изд. – Москва : РИОР ; ИНФРА-М, 2020. – 152 с.

2 **Гуров, В. В.** Микропроцессорные системы : учебное пособие / В. В. Гуров. – Москва : ИНФРА-М, 2021. – 336 с.

3 **Новожилов, О. П.** Основы микропроцессорной техники: учебное пособие / О. П. Новожилов. – 2-е изд. – Москва: РадиоСофт, 2011. – 336 с.

4 **Хоровиц, П.** Искусство схемотехники: пер. с англ. / П. Хоровиц, У. Хилл. – 7-е изд. – Москва: Бином, 2014. – 704 с.