Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Промышленная электроника»

Э. М. Виноградов

ПРОЕКТИРОВАНИЕ МИКРОКОНТРОЛЛЕРНОЙ СИСТЕМЫ УПРАВЛЕНИЯ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ
по курсовому проектированию
по дисциплине «Микропроцессорная техника»
для студентов специальности 1-36 04 02
«Промышленная электроника»
дневной и заочной форм обучения

Электронный аналог печатного издания

Гомель 2015

УДК 621.38(075.8)
ББК 32.973.26-04я73
В49

Рекомендовано к изданию научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 4 от 24.11.2014 г.*)

Рецензент: доц. каф. «Автоматизированный электропривод» ГГТУ им. П. О. Сухого,
канд. техн. наук *В. В. Тодарев*

Виноградов, Э. М.

В49 Проектирование микроконтроллерной системы управления : учеб.-метод. пособие по курсовому проектированию по дисциплине «Микропроцессорная техника» для студентов специальности 1-36 04 02 «Промышленная электроника» днев. и заоч. форм обучения / Э. М. Виноградов. – Гомель : ГГТУ им. П. О. Сухого, 2015. – 54 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

ISBN 978-985-535-262-5.

Содержит рекомендации по выполнению курсового проекта, примеры разработки схем и программ, требования к оформлению проекта.

Для студентов специальности 1-36 04 02 «Промышленная электроника» дневной и заочной форм обучения.

УДК 621.38(075.8)
ББК 32.973.26-04я73

ISBN 978-985-535-262-5

ПРЕДИСЛОВИЕ

Широкое распространение микроконтроллеров требует квалифицированных специалистов. Они должны, с одной стороны, хорошо знать элементную базу цифровых устройств (микроконтроллеров, микросхем различной степени интеграции), с другой стороны, владеть методами алгоритмизации и программирования различных технических задач.

Для закрепления и углубления знаний, приобретенных студентами в курсе лекций по дисциплине «Микропроцессорная техника», программой специальности 1-36 04 02 «Промышленная электроника» предусмотрен курсовой проект.

Целью курсового проекта является приобретение студентами навыков по проектированию и расчету микроконтроллерных систем, осуществляющих сбор данных, их обработку, отображение и преобразование в унифицированные информационные сигналы.

Задача курсового проекта – разработка микроконтроллерной системы управления технологическим объектом, выбор элементов системы, составление структурных и принципиальных схем, разработка программного обеспечения.

1 ЗАДАНИЕ НА ПРОЕКТИРОВАНИЕ

В курсовом проекте требуется разработать микроконтроллерную систему (МКС) управления некоторым технологическим объектом. Структура МКС приведена на рисунке 1.1. МКС состоит из объекта управления, микроконтроллера, пульта управления и аппаратуры их взаимной связи. Микроконтроллер (МК) путем периодического опроса принимает информацию об объекте от аналоговых и цифровых датчиков. Выходные сигналы датчиков вследствие их различной физической природы могут потребовать промежуточного преобразования на аналого-цифровых преобразователях (АЦП) или на схемах формирователей сигналов (ФС), которые чаще всего выполняют функции гальванической развязки и формирования уровней двоичных сигналов стандарта ТТЛ [3], [4].

МК с требуемой периодичностью вырабатывает управляющие сигналы на своих выходных портах в соответствии с законом управления. Некоторая часть этих сигналов интерпретируется как совокупность простых двоичных сигналов, которые через схемы формирователей (усилители мощности, реле, оптроны и т. п.) поступают на

исполнительные устройства. Другая часть управляющих сигналов представляет собой многоразрядные двоичные коды, которые через цифроаналоговые преобразователи (ЦАП) воздействуют на исполнительные устройства аналогового типа. АЦП и ЦАП могут быть отдельными блоками МКС, но могут являться внутренними модулями микроконтроллера. Это определяется типом применяемого МК.

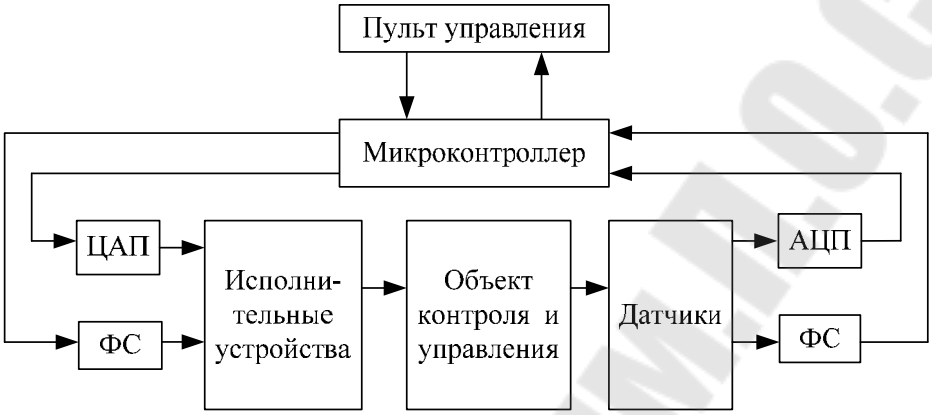

Рисунок 1.1 – Структура микроконтроллерной системы

С помощью пульта управления (ПУ) оператор имеет возможность получать значения некоторых сигналов, выводить на индикаторы информацию о состоянии объекта и т. п.

В курсовом проекте необходимо разработать структурную схему МКС, принципиальную схему системы, разработать программы для МК, обеспечивающие выполнение алгоритмов контроля и управления в соответствии с заданием.

1.1 Алгоритм работы МКС

Блок-схема алгоритма (БСА) работы МКС представлена на рисунке 1.2.

После включения электропитания или нажатия кнопки «Сброс» на пульте управления выполняется начальная установка (инициализация) системы (блок *1* БСА): вывод начальных значений управляющих воздействий, настройка внутренних модулей МК, таймер опроса начинает отсчет времени $T_{ОПР}$. Блок *2* выполняет задачу логического управления. В нем производится ввод и обработка цифровой информации, поступающей от двоичных датчиков. В блоке *3* производится ввод и обработка аналоговой информации. Далее в блоке *4* происходит опрос клавиатуры, находящейся на пульте управления. Если ни одна из клавиш клавиатуры не была нажата, то программа переходит

к блоку *8*, в котором проверяется, не прошло ли время опроса $T_{ОПР}$. Если время истекло, то программа вновь переходит на опрос клавиатуры (блок *4*). Если же оператор нажал какую-либо клавишу клавиатуры с целью вывода на дисплей значения входного напряжения U1–U5, то программа переходит к блоку *6*, в котором в зависимости от номера нажатой клавиши подготавливаются данные, а в блоке *7* происходит их вывод на дисплей. После этого в блоке *8* проверяется, не прошло ли время опроса $T_{ОПР}$. Если время опроса истекло, то программа переходит к блоку *9*, в котором таймер подготавливается для нового отсчета времени $T_{ОПР}$. Затем программа переходит на выполнение блока *2* алгоритма, т. е. зацикливается.

Рисунок 1.2 – Блок-схема алгоритма работы МКС

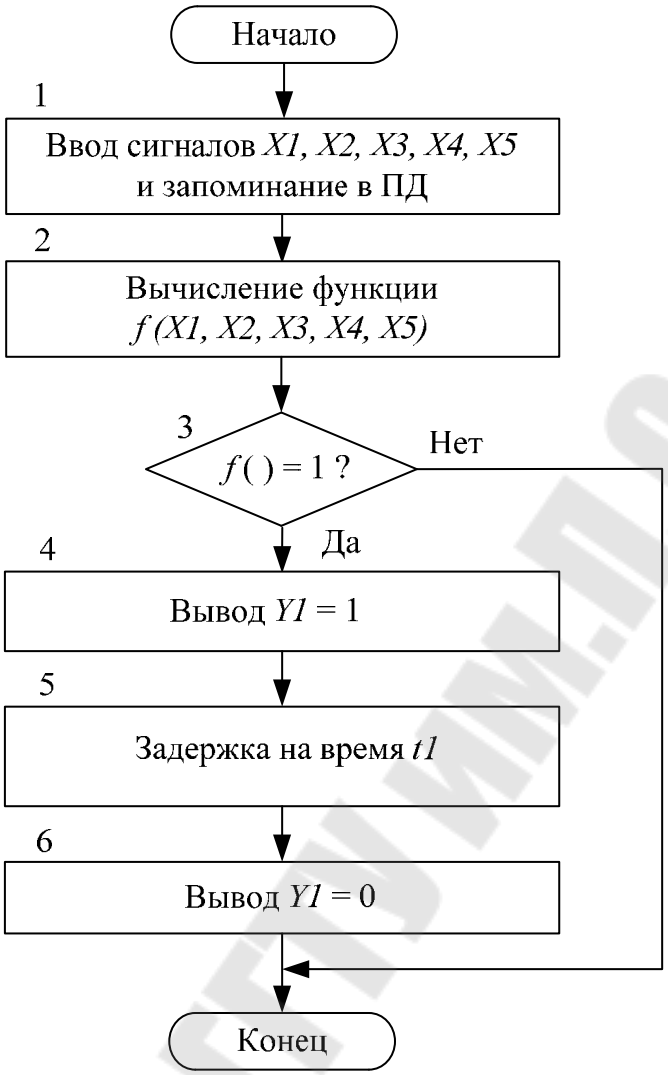

Рисунок 1.3 – БСА обработки цифровой информации

Блок-схема алгоритма обработки цифровой информации приведена на рисунке 1.3. Сигналы от двоичных датчиков X1, X2, X3, X4, X5 вводятся в МК и запоминаются в его памяти данных (ПД). Затем вычисляется значение логической (булевой) функции f(X1,X2,X3,X4,X5) в соответствии с выражением, определенном в задании на курсовой проект. Это значение выдается в качестве управляющего сигнала Y1 по соответствующему выходному каналу на исполнительное устройство. При единичном значении логической функции f(X1,X2,X3,X4,X5) микроконтроллер вырабатывает выходной сигнал ТТЛ-уровня Y=1 длительностью t1. Форма сигнала Y1 приведена на рисунке 1.4.

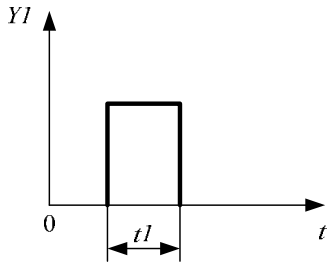

Рисунок 1.4 – Форма сигнала Y1

На рисунке 1.5 приведена БСА обработки аналоговой информации. Сигналы U1, U2, U3, U4, U5 от аналоговых датчиков (однополярное напряжение от 0 до +5 В) преобразуются в цифровую форму с помощью АЦП, входящего в состав МК. С выхода АЦП 10-разрядные коды W1, W2, W3, W4 и W5, представляющие собой целые беззнаковые двоичные числа, поступают на обработку.

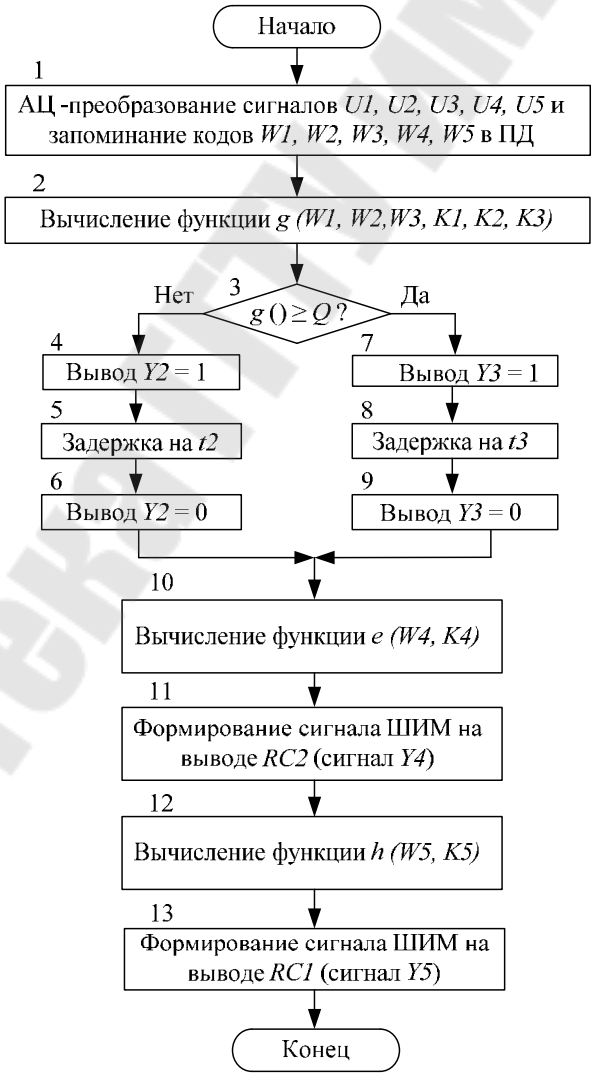

Рисунок 1.5 – БСА обработки аналоговой информации

Обработка кодов начинается с вычисления функции g(W1,W2,W3,K1,K2,K3), где коэффициенты K1, K2, K3 – 16-разрядные целые двоичные числа, хранящиеся в программной памяти МК. Вид функции g( ) определяется из задания на курсовой проект. Полученное значение функции g( ) сравнивается с константой Q (уставкой), хранящейся в программной памяти. В зависимости от результата сравнения МК вырабатывает двоичные управляющие сигналы ТТЛ-уровня Y2 или Y3 длительностью t2 или t3, соответственно. Форма сигналов Y2 и Y3 аналогична сигналу Y1, изображенному на рисунке 1.4.

Цифровой код W4, образованный аналоговым сигналом U4, используется для вычисления функции e(W4,K4), где K4 – коэффициент, хранящийся в программной памяти. Конкретный вид функции e( ) определяется заданием на курсовой проект. Полученное значение функции e( ) используется для формирования управляющего сигнала Y4, представляющего собой последовательность импульсов с широтной модуляцией. При широтно-импульсной модуляции (ШИМ) период следования T импульсов и их частота f являются постоянными, а изменяется длительность импульсов $t_и$ в диапазоне от $t_{и.мин}$ до $t_{и.макс}$ пропорционально значению функции e( ), как показано на рисунке 1.6. Частота импульсов ШИМ определяется исходя из заданной тактовой частоты f_{OSC} работы МК.

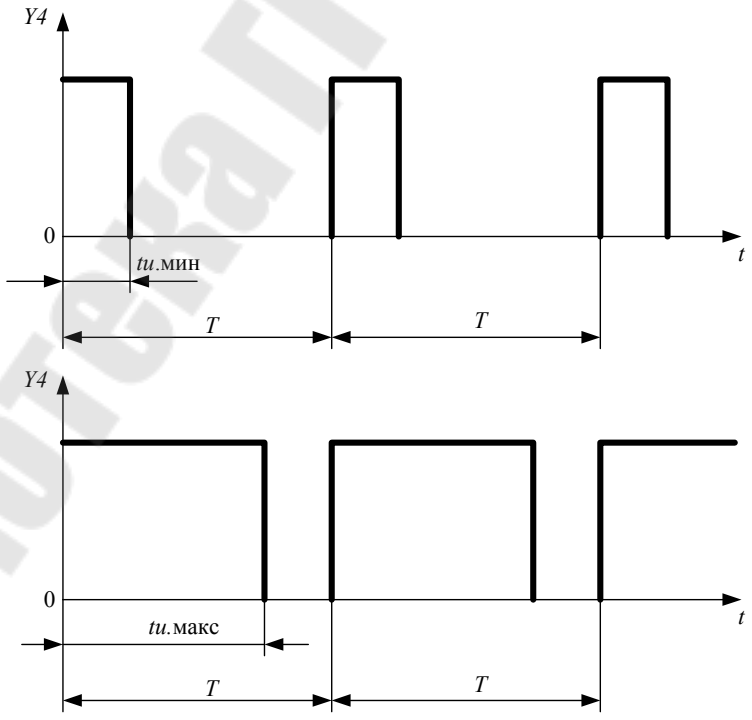

Рисунок 1.6 – Временные диаграммы сигналов при широтно-импульсной модуляции

Цифровой код W5, образованный аналоговым сигналом U5, используется для вычисления функции h(W5,K5). Полученное значение функции h( ) служит для формирования импульсной последовательности ШИМ, из которой путем фильтрации образуется аналоговый сигнал Y5, поступающий на исполнительное устройство.

В МКС имеется также двоичный датчик аварийной ситуации, единичный сигнал с которого X0 должен вызывать аварийный останов системы в любой момент выполнения рабочего цикла. Выход из аварийного режима возможен только после выключения питания МКС.

1.2 Пульт управления

Пульт управления должен содержать следующие элементы.

1 Жидкокристаллический дисплей (ЖКД) для отображения значений аналоговых входных напряжений U1–U5.

2 Клавишный переключатель для выбора вида отображаемой информации на ЖКД.

3 Кнопка «Сброс», при нажатии на которую происходит сброс микроконтроллера.

4 Светодиод индикации и динамик, на которые подаются сигналы от МК в аварийном режиме.

Общий вид передней панели пульта управления приведен на рисунке 1.7.

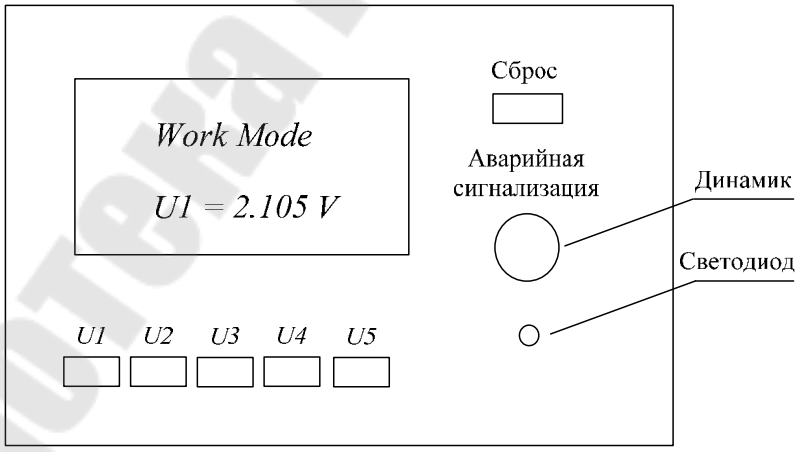

Рисунок 1.7 – Вид передней панели пульта управления

После включения электропитания МКС находится в рабочем режиме, в котором циклически выполняется ввод и обработка информации от датчиков с периодом $T_{ОПР}$ (величина $T_{ОПР}$ определяется в за-

дании на проектирование). В этом режиме на ЖКД выводится текст сообщения “Work Mode” (рабочий режим). Оператор при нажатии на одну из клавиш с надписями “U1”, “U2”, “U3”, “U4”, “U5” может вывести на ЖКД значения входных напряжений U1, U2, U3, U4 или U5, которые подаются от датчиков. Текст сообщения на ЖКД при этом может быть следующим:

Work Mode
U1 = 2.345 V

При поступлении сигнала от аварийного датчика МКС переходит в аварийный режим. В этом режиме прекращается ввод информации от датчиков и формирование управляющих сигналов, а на пульте управления включается аварийная сигнализация (световая – с частотой 1 Гц и звуковая – с частотой 500 Гц, на дисплей выводится текст сообщения “Error” – ошибка). Выход из аварийного режима возможен только после выключения питания МКС.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КУРСОВОГО ПРОЕКТА

Проектирование микроконтроллерной системы рекомендуется начинать с аппаратурной части [2], [4]. Сначала следует составить структурную схему МКС исходя из описания работы системы и заданных параметров для проектирования. Затем нужно выбрать схему включения микроконтроллера, разработать схемы ввода сигналов от датчиков, схемы вывода управляющих сигналов, разработать схемы подключения элементов индикации и управления.

2.1 Структурная схема МКС

Структурная схема разрабатываемой МКС приведена на рисунке 2.1. Ее центральной частью является микроконтроллер PIC16F877 фирмы Microchip [6].

Аналоговые сигналы U1, U2, U3, U4, U5 поступают на выводы МК, которые являются входными каналами внутреннего АЦП. Цифровые (двоичные) сигналы X1, X2, X3, X4, X5 поступают на входы МК, которые являются выводами порта, настроенными на ввод. Управляющие сигналы Y1, Y2 и Y3, представляющие собой одиночные импульсы определенной длительности, вырабатываются микроконтроллером. Управляющий сигнал Y4 представляет собой периодическую последовательность импульсов с ШИМ, которая вырабатывается внутренним мо-

дулем МК. Сигналы Y1–Y4 усиливаются по току с помощью формирователей ФС1–ФС4. Аналоговый управляющий сигнал Y5 формируется с помощью схемы ФС5, на которую подаются импульсы с ШИМ от микроконтроллера.

Пульт оператора включает в себя: жидкокристаллический дисплей (ЖКД) для отображения аналоговых сигналов (напряжений) U1–U5; клавиатуру с 5-ю клавишами для выбора сигналов; кнопку «Сброс» для сброса МК; устройство аварийной сигнализации, выдающее звуковые и световые сигналы определенной частоты.

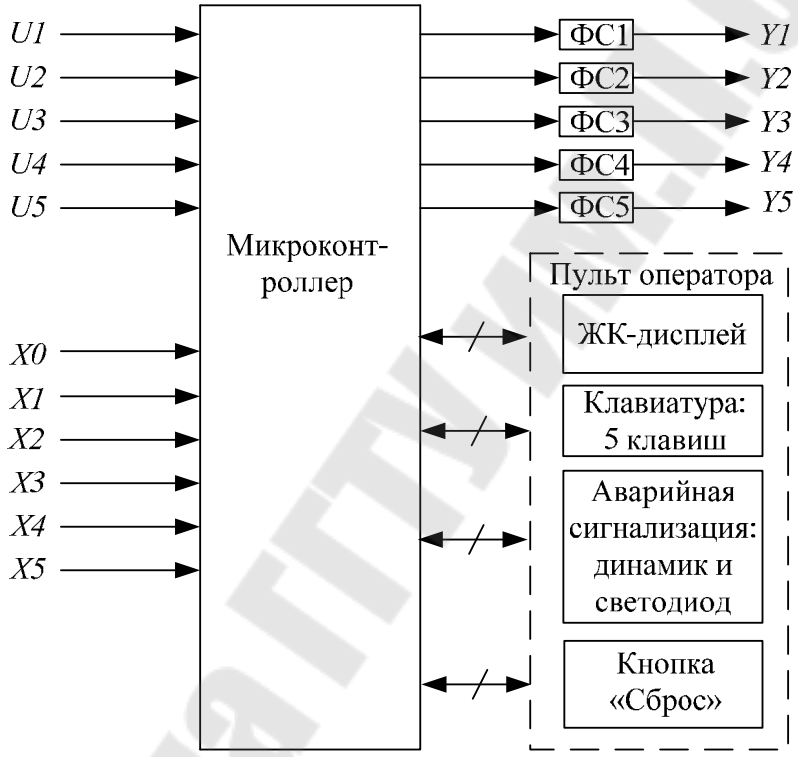

Рисунок 2.1 – Структурная схема МКС

2.2 Схема подключения микроконтроллера

Для работы PIC-микроконтроллера необходимо [1], [6]:

1) присоединить времязадающую цепь для работы внутреннего тактового генератора;

2) обеспечить сброс МК при включении электропитания.

На рисунке 2.2 приведена типовая схема подключения микроконтроллера PIC16F877, который используется для выполнения курсового проекта.

Для обеспечения генерации тактовой частоты f_{OSC} к выводам OSC1 и OSC2 подключен кварцевый резонатор ZQ1. Частота кварцевого резонатора f_{ZQ} выбирается из условия $f_{OSC} = f_{ZQ}$ и может быть в пределах 0…20 МГц (определяется по заданию на курсовой проект).

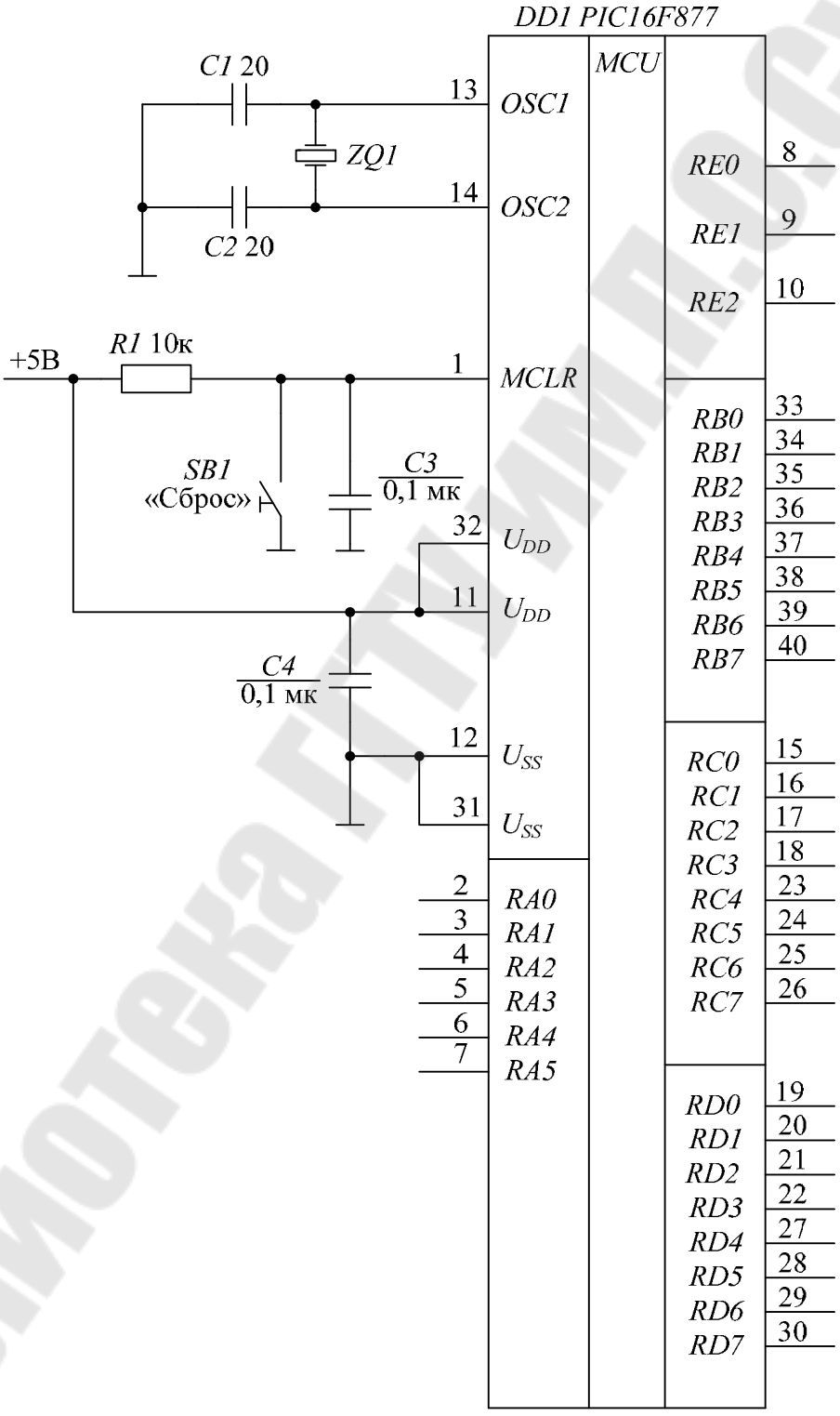

Рисунок 2.2 – Схема подключения микроконтроллера PIC16F877

Конденсаторы C1, C2 служат для облегчения запуска внутреннего тактового генератора. Цепочка C3, R1 обеспечивает сброс МК при подачи электропитания. С помощью кнопки SB1, расположенной на пульте управления, сброс МК может выполнить оператор в любой момент времени.

Конденсатор C4 служит для фильтрации высокочастотных помех, возникающих на выводах источника питания при работе микросхемы.

Тридцать три двунаправленные линии портов A, B, C, D, E микроконтроллера могут быть использованы в МКС для ввода и вывода данных.

2.3 Схема ввода цифровых сигналов

Так как по заданию цифровые сигналы X1–X5 являются двоичными сигналами ТТЛ-уровней, то их можно непосредственно подавать на линии портов МК. Необходимо эти линии настроить на ввод.

На рисунке 2.3 приведена схема подключения сигналов X1–X5 к линиям RC3–RC7 порта C микроконтроллера. Разъем XS1 (розетка) служит для подключения соединительного кабеля (вилки XP1) от датчиков цифровых сигналов.

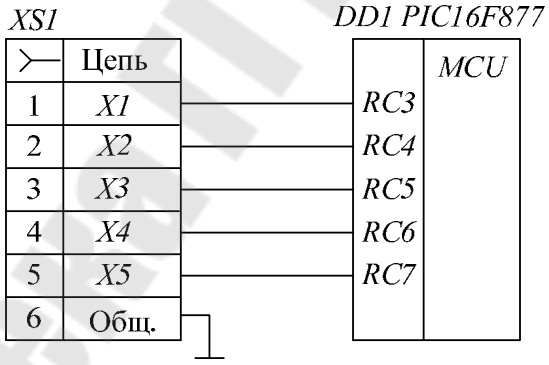

Рисунок 2.3 – Схема подключения цифровых сигналов X1–X5 к МК

2.4 Схема ввода аналоговых сигналов

Так как аналоговые сигналы U1–U5 по заданию являются напряжениями постоянного тока, изменяющимися в пределах от 0 до +5 В, то их можно непосредственно подавать на линии портов, которые служат аналоговыми каналами внутреннего модуля АЦП микроконтроллера.

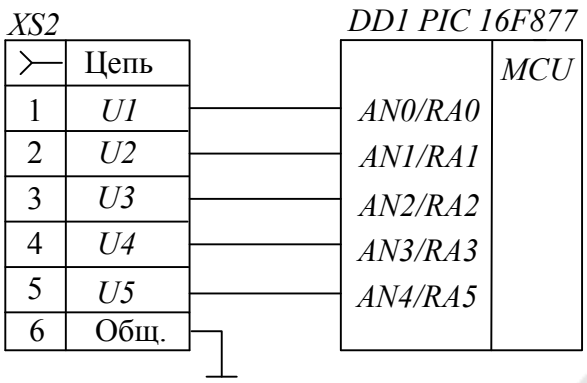

Рисунок 2.4 – Схема подключения аналоговых сигналов U1–U5 к МК

На рисунке 2.4 приведена схема подключения сигналов U1–U5 к линиям RA0–RA5 порта А микроконтроллера. Эти линии могут быть настроены как аналоговые каналы AN0–AN4 модуля АЦП. Разъем XS2 (розетка) служит для подключения соединительного кабеля (вилки XP2) от датчиков аналоговых сигналов.

2.5 Схемы вывода цифровых управляющих сигналов

В проектируемой МКС имеются 4 цифровых управляющих сигнала. Сигналы Y1, Y2, Y3 представляют собой одиночные импульсы ТТЛ-уровней, поступающие на исполнительные устройства МКС. Сигнал Y4 – это периодическая последовательность импульсов с широтно-импульсной модуляцией, вырабатываемая внутренним модулем ШИМ микроконтроллера.

Управляющие сигналы снимаются с линий портов МК. Нагрузочная способность линий портов PIC-микроконтроллеров довольно высока: максимальный выходной ток равен 25 мА [6]. Такая нагрузочная способность позволяет подключать многие исполнительные устройства непосредственно к линиям портов МК. Однако с целью повышения надежности рекомендуется подключать исполнительные устройства к портам МК через буферные элементы, которые в простейшем случае могут представлять собой микросхемы повторителей с открытым коллектором, например, К155ЛП9. Такие микросхемы могут обеспечить ток нагрузки до 40 мА, а главное, при аварийной ситуации в нагрузке, например, коротком замыкании, выходит из строя микросхема повторителя, а не дорогостоящий микроконтроллер.

На рисунке 2.5 приведена схема формирования управляющих сигналов МКС с использованием микросхемы К155ЛП9, в состав ко-

торой входят 6 повторителей с открытым коллектором [10]. Разъем XS3 (розетка) служит для подключения соединительного кабеля (вилки XP3) к исполнительным устройствам.

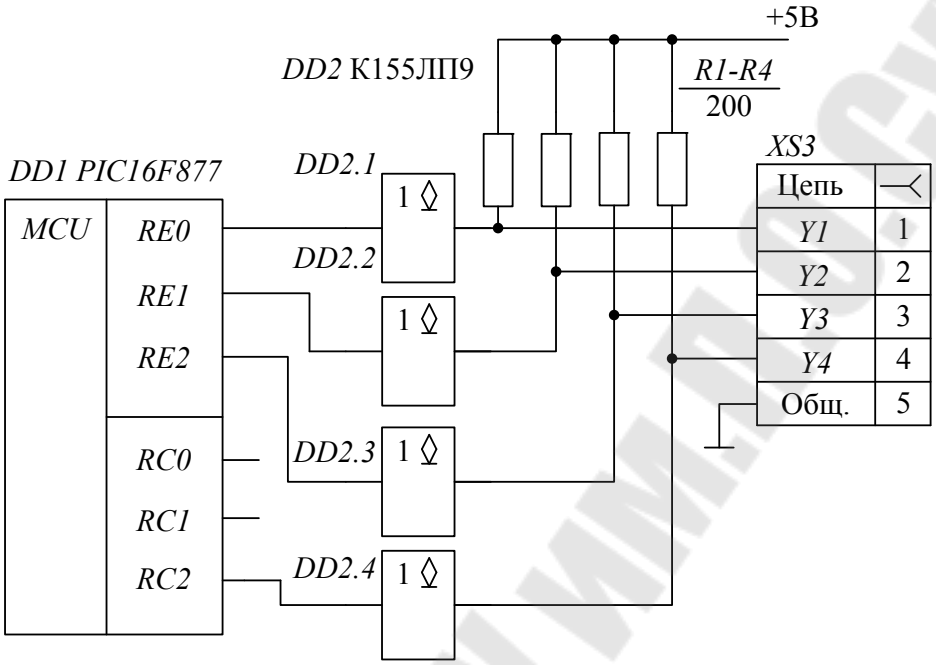

Рисунок 2.5 – Схема вывода управляющих сигналов Y1–Y4

В некоторых случаях, например, при значительном расстоянии исполнительного устройства от МКС, а также для исключения влияния помех от наводок на соединительные провода, необходимо гальваническое разделение, т. е. по постоянному току, цепей МК и управляющих сигналов. На рисунке 2.6 приведена схема формирования управляющего сигнала Y1 с гальванической развязкой, реализованной с помощью транзисторной оптопары АОТ128А. Следует иметь в виду, что источник питания с напряжениями $+U_{п1}$, $–U_{п2}$ должен быть гальванически не связан с основным источником +5 В (получать питание от отдельной обмотки сетевого трансформатора).

2.6 Схема вывода аналогового управляющего сигнала

Для получения аналогового выходного сигнала в микроконтроллерных системах используются цифроаналоговые преобразователи (ЦАП). Модули ЦАП в составе микроконтроллеров являются большой редкостью, поэтому для этих целей приходится применять специальные микросхемы ЦАП, что значительно усложняет и удорожает МКС.

Однако для PIC-микроконтроллеров более простой альтернативой является выполнение цифроаналогового преобразования с помощью ШИМ-сигналов, получаемых с помощью внутренних модулей МК, которые могут работать в режиме широтно-импульсного модулятора.

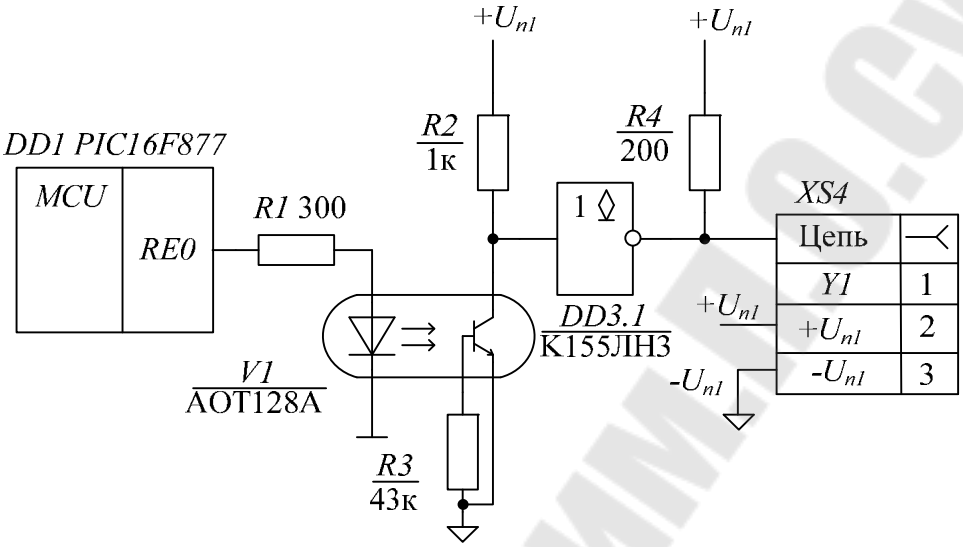

Рисунок 2.6 – Схема вывода управляющего сигнала Y1
с гальванической развязкой

Способ преобразования цифровых значений в аналоговые напряжения отличается сравнительной простотой. Он заключается в изменении коэффициента заполнения импульсов прямоугольного напряжения пропорционально исходному цифровому значению. Временные диаграммы импульсов ШИМ были приведены на рисунке 1.6. Если обозначить T – период ШИМ-сигнала, $f_{ШИМ} = 1/T$ – его частота, $t_и$ – длительность импульса, то коэффициент заполнения $\gamma = t_и/T$.

В англоязычной литературе вместо коэффициента заполнения используется термин Duty Cycle (величина рабочего цикла).

Спектральный анализ ШИМ-сигнала на основе преобразования Фурье позволяет представить этот сигнал в виде суммы постоянной составляющей и отдельных гармоник с различными амплитудами и частотами:

$$u(t) = U_0 + U_{1m}\sin\omega t + U_{2m}\sin 2\omega t + U_{3m}\sin 3\omega t + \dots ,$$

где U_0 – постоянная составляющая; U_{1m}, U_{2m}, U_{3m}, … – амплитуды гармоник.

Если ШИМ-сигнал пропустить через фильтр низкой частоты (ФНЧ) и отфильтровать составляющие переменного напряжения, то в

идеальном случае остается составляющая постоянного напряжения. При этом выходной сигнал ФНЧ определяется как

$$u(t) = U_0 = U_m \cdot \gamma,$$

где U_m – амплитуда ШИМ-сигнала.

В PIC-микроконтроллерах ШИМ-сигнал снимается с выводов порта С, поэтому можно считать, что U_m равно напряжению питания микроконтроллера U_{DD}, следовательно:

$$u(t) = U_{DD} \cdot \gamma.$$

Таким образом, исходное цифровое значение, представленное коэффициентом заполнения γ, преобразуется в постоянный аналоговый сигнал U_0. Причем преобразование выполняется по линейному закону с известным коэффициентом пропорциональности.

На рисунке 2.7 приведена схема формирования аналогового напряжения посредством фильтрации ШИМ-сигнала, снимаемого с вывода RC1 порта С. Фильтр низкой частоты представляет собой простейший RC-фильтр 1-го порядка. Операционный усилитель включен по схеме повторителя. Он необходим для устранения влияния сопротивления нагрузки на работу RC-фильтра.

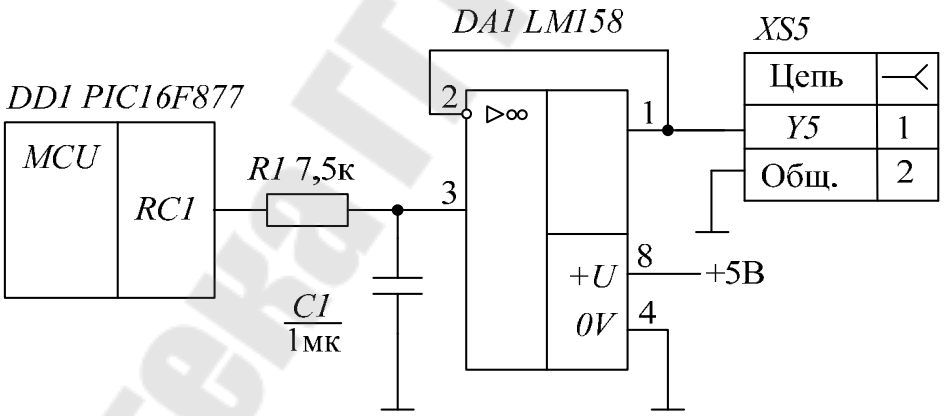

Рисунок 2.7 – Схема подключения фильтра к выходу канала PWM2

Частота среза RC-фильтра равна:

$$f_c = 1/(2\pi RC). \qquad (2.1)$$

Произведение RC называют постоянной времени фильтра.

Выбор параметров RC-фильтра определяется многими факторами, но можно воспользоваться некоторыми рекомендациями для первоначального выбора [8].

Для расчета постоянной времени фильтра можно применить следующую эмпирическую формулу:

$$RC = (U_{DD} \cdot T_{ШИМ} \cdot 0{,}37)/\Delta U, \qquad (2.2)$$

где U_{DD} – напряжение питания МК; $T_{ШИМ}$ – период ШИМ-сигнала; ΔU – допустимая величина пульсаций на выходе фильтра.

Формула (2.2) предполагает, что постоянная времени RC по крайней мере в 10 раз больше, чем период ШИМ-сигнала.

Например, если U_{DD} = 5 В; $T_{ШИМ}$ = 0,2 мс (частота $f_{ШИМ}$ = 5 кГц), допустимая пульсация на выходе фильтра ΔU = 0,05 В (1% от U_{DD}), то из формулы (2.2) получим RC = 0,0074 с.

Если взять конденсатор C = 1 мкФ, то сопротивление R = 7400 Ом. Выбираем ближайший номинал из стандартного ряда R = 7,5 кОм [9]. Частота среза такого фильтра будет f_c = 21,2 Гц.

2.7 Схема подключения ЖК-дисплея

Алфавитно-цифровые ЖК-дисплеи (ЖКД), или по-английски LCD (Liquid Crystal Display) представляют собой недорогие и удобные модули, позволяющие сэкономить время и ресурсы при разработке новых изделий. Они обеспечивают отображение большого объема информации при хорошей различимости и низком энергопотреблении.

Японская фирма Hitachi разработала специальную микросхему – контроллер HD44780 для управления подобными ЖК-дисплеями. Этот контроллер определил интерфейс, который стал стандартом «де-факто» для ЖК-дисплеев. Аналоги этого контроллера или совместимые с ним по интерфейсу микросхемы выпускают множество фирм, среди которых Epson, Toshiba, Sanyo, Samsung, Philips. ЖК-дисплеи на базе данного контроллера можно встретить в самых разнообразных устройствах: измерительных приборах, медицинском оборудовании, промышленном и технологическом оборудовании, офисной технике – принтерах, телефонах, факсимильных и копировальных аппаратах.

Контроллер HD44780 может управлять 2-мя строками по 40 символов в каждой (для дисплеев с 4-мя строками по 40 символов используются два однотипных контроллера) при матрице символов 5 × 7 точек.

Существует несколько различных стандартных форматов ЖК-дисплеев (количество символов × число строк): 8 × 2; 16 × 1; 16 × 2; 16 × 4; 20 × 1; 20 × 2; 20 × 4; 40 × 2; 40 × 4. В курсовом проекте будет применяться ЖК-дисплей типа WM-C1602N-2YLY с 32-позиционным

индикатором (16 × 2 символов). Конструктивно дисплей представляет собой печатную плату с установленными на ней контроллером HD44780 и ЖК-индикатором. Плата содержит 14-контактное поле, расположенное в нижней части, а также 2 контакта (выводы питания подсветки) в правой части. Назначение выводов поясняет таблица 2.1.

Таблица 2.1 – Назначение выводов ЖК-дисплея

Номер вывода	Обозначение	Функция
1	V_{DD}	Напряжение питания (+5 В)
2	V_{SS}	Общий (земля)
3	V_{EE}	Управление контрастом
4	RS	Сигнал выбора регистра
5	R/W	Сигнал чтение/запись
6	E	Сигнал разрешения
7–14	D0–D7	Биты данных

Основными чертами интерфейса контроллера HD44780 являются следующие характеристики.

Данные передаются по 4- или 8-разрядной шине, что определяется пользователем. Эти данные могут быть либо командами, либо символьной информацией. Использование 4-разрядного режима позволяет ограничить весь интерфейс 7-ю линиями, однако процесс передачи данных буде немного более медленным, чем при 8-разрядном режиме.

Управление выполняется с использованием трех линий:

– линия RS (выбор регистра), которая определяет, будет ли передаваться команда или символьные данные;

– линия R/W (чтение/запись), которая определяет направление перемещения данных (R/W = 1 – чтение; R/W = 0 – запись);

– линия E (разрешение), которая выполняет функцию тактирования с целью синхронизации процесса передачи данных.

Контроллер имеет простой набор команд, который позволяет управлять работой дисплея. В его состав входят команды инициализации и сброса дисплея, управления положением и характеристиками курсора и т. д.

На рисунке 2.8 приведена схема подключения модуля ЖКД к микроконтроллеру с использованием 4-разрядной шины данных. Вывод R/W дисплея присоединен к общему проводу питания, так как библиотечные функции управления ЖК-дисплеем компилятора mikroC PRO,

которые будут использоваться в данном проекте, выполняют только запись данных из микроконтроллера, т. е. при этом должно быть R/W = 0. Потенциометр R1 служит для регулирования контраста отображаемых символов на дисплее.

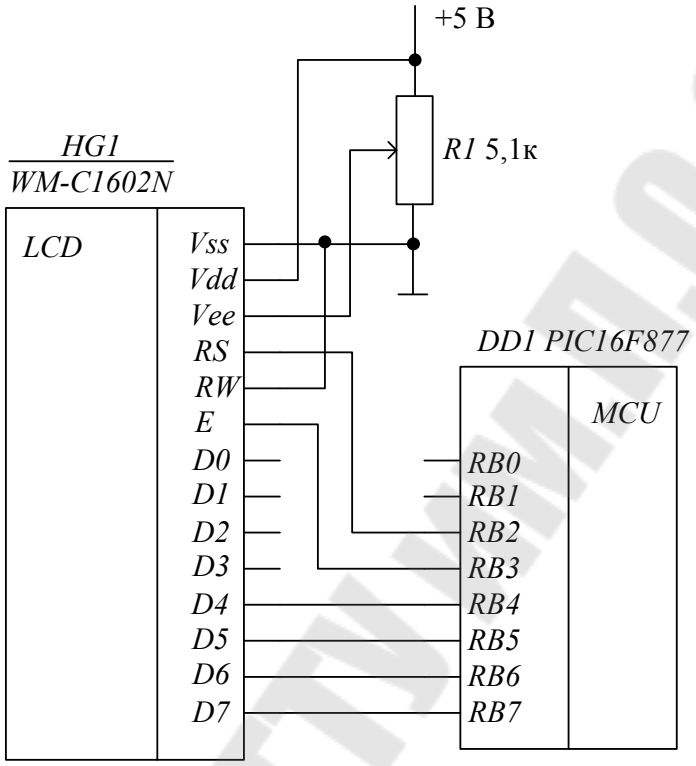

Рисунок 2.8 – Схема подключения ЖКД к микроконтроллеру

2.8 Схема подключения клавиатуры

В разрабатываемой МКС на пульте управления имеется простейшая клавиатура, состоящая из клавиш для выбора отображаемой на дисплее информации. Клавиатуры по методу аппаратурной реализации бывают двух видов: кодирующие и некодирующие [4].

В кодирующих клавиатурах каждый контакт подключается к отдельной линии порта ввода МК. При этом схемным путем формируется код, соответствующий нажатой клавише. Благодаря простоте реализации, эти клавиатуры широко используются при небольшом количестве клавиш.

При большом числе клавиш удобнее применять некодирующие (матричные) клавиатуры, которые представляют собой простую матрицу переключателей (требуемой размерности), включенных на пере-

сечении строк и колонок матрицы. Идентификация (кодирование) нажатой клавиши в таких клавиатурах выполняется программой.

Контакты клавиатур (переключателей и кнопок) бывают с фиксацией замкнутого состояния и без фиксации. Первые остаются в нажатом состоянии (контакты замкнуты), вторые после отжатия (освобождения клавиши или кнопки) размыкают свои контакты.

На рисунке 2.9 приведена схема подключения клавиатуры из пяти клавиш SB1–SB5 к порту D микроконтроллера.

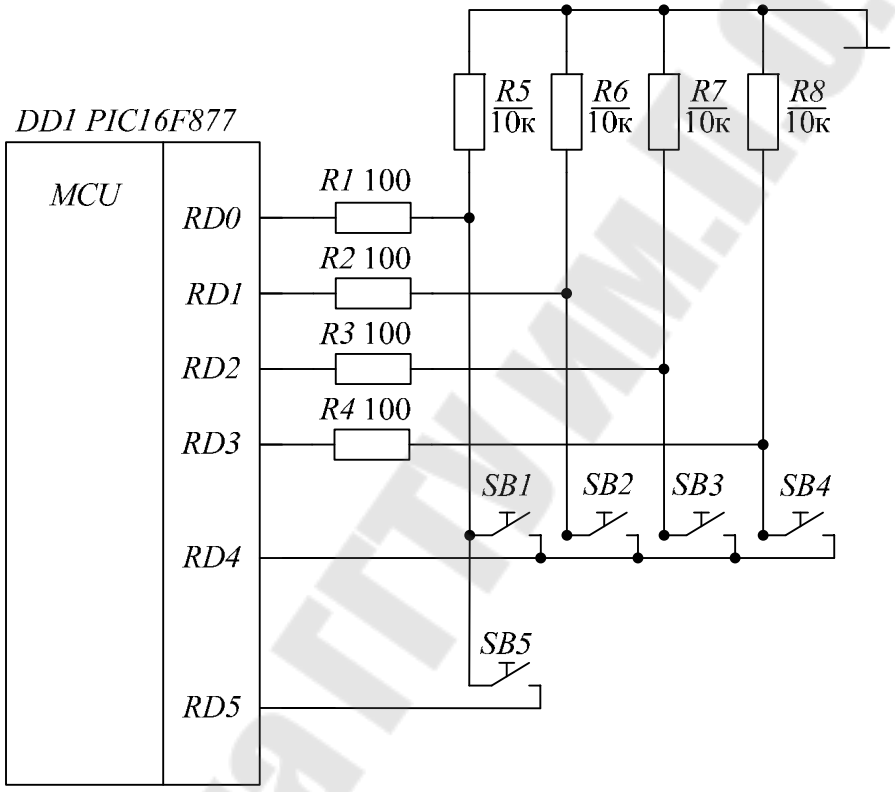

Рисунок 2.9 – Подключение клавиатуры к микроконтроллеру

Эта клавиатура может быть отнесена к типу 4 × 2, где 4 – число вертикальных линий (колонок); 2 – число горизонтальных линий (строк) матрицы. Контакты клавиатуры – без фиксации замкнутого состояния. Клавиши служат для выбора аналоговых сигналов U1–U5. Резисторы R1–R4 включены для защиты выходов линий RD0–RD3 порта D в случае одновременного нажатия нескольких клавиш. Резисторы R5–R8 обеспечивают уровень логического нуля на вертикальных линиях клавиатуры.

2.9 Подключение аварийного датчика

Так как по заданию сигнал X0 от аварийного датчика является двоичным сигналом ТТЛ-уровня, то его можно непосредственно подавать на линию порта МК. Необходимо эту линию настроить на ввод.

На рисунке 2.10 приведена схема подключения сигнала X0 к линии RB0 порта B микроконтроллера. Разъем XS4 (розетка) служит для подключения соединительного кабеля (вилки XP4) от аварийного датчика.

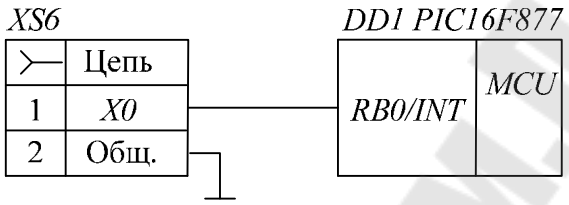

Рисунок 2.10 – Схема подключения аварийного датчика

2.10 Схемы подключения аварийной сигнализации

На рисунке 2.11 приведена схема подключения динамика BA1 для звуковой сигнализации. Транзистор VT1 усиливает по току выходной сигнал с линии RC0 порта.

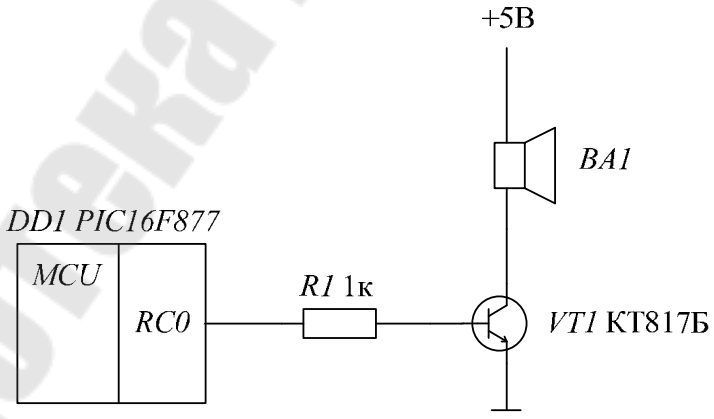

Рисунок 2.11 – Схема подключения динамика для звуковой сигнализации

На рисунке 2.12 приведена схема подключения светодиода VD1 для аварийной световой сигнализации.

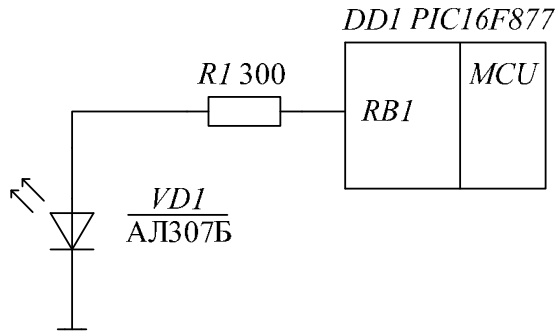

Рисунок 2.12 – Подключение светодиода для аварийной сигнализации

2.11 Библиотечные функции компилятора mikroC PRO for PIC

При разработке программ в курсовом проекте будет использоваться язык Си для компилятора mikroC PRO for PIC компании MikroElektronika [1], [7]. Этот компилятор является мощным, обладающим богатыми функциональными инструментами для разработки устройств на базе PIC-микроконтроллеров.

Компилятор mikroC PRO имеет огромное количество библиотечных функций для работы с внутренними модулями микроконтроллера и стандартными внешними устройствами. Их полное описание можно найти в [7]. Ниже кратко рассмотрены библиотечные функции, которые используются при разработке программ в курсовом проекте.

2.11.1 Функции для работы с АЦП (ADC Library functions)

ADC_Init – эта функция производит инициализацию внутреннего модуля АЦП микроконтроллера. При этом предполагается, что АЦП будет тактироваться от внутреннего RC-генератора модуля, а выводы портов МК, используемые как аналоговые входы, сконфигурированы на ввод.

Прототип:

```
void  ADC_Init( );
```

Пример:

```
// инициализировать модуль АЦП с использованием в качестве
// аналоговых входов линий порта A – RA0, RA1, RA2
// (аналоговые каналы AN0, AN1, AN2)
TRISA = 0b00000111;  // настроить линии порта A – RA0, RA1, RA2
                     // на ввод, остальные – на вывод
```

……..
ADC_Init( );

ADC_Read – эта функция вводит аналоговое напряжение из заданного канала и запускает АЦП. Функция возвращает 10-разрядное двоичное число без знака (выходной код АЦП). Перед использованием этой функции модуль АЦП должен быть инициализирован.

Прототип:

unsigned int ADC_Read(unsigned char channel);

Здесь параметр channel (канал) представляет собой номер аналогового канала, из которого извлекается напряжение.

Для микроконтроллера PIC16F877 номера аналоговых каналов соответствуют следующим выводам портов A и E:

Номер канала	Вывод порта
0	RA0 (AN0)
1	RA1 (AN1)
2	RA2 (AN2)
3	RA3 (AN3)
4	RA5 (AN4)
5	RE0 (AN5)
6	RE1 (AN6)
7	RE2 (AN7)

Пример:

```
// прочитать код из канала 4 модуля АЦП
unsigned  int  adc_result;       // объявление двухбайтной переменной
……
adc_result = ADC_Read(4);
```

2.11.2 Библиотечные функции управления ЖК-дисплеем (LCD Library functions)

Рассмотрим библиотечные функции для управления алфавитно-цифровым ЖК-дисплеем на основе контроллера HD44780 с 4-разрядной шиной данных.

Lcd_Init – эта функция инициализирует модуль ЖКД.

Прототип:

void Lcd_Init( );

Перед тем, как использовать эту функцию, необходимо указать соединение ЖК-модуля с выводами портов МК.

Пример:

```
// инициализация ЖКД, присоединенного к порту B
// микроконтроллера по схеме, приведенной на рис. 2.6.
// присоединение выводов ЖКД
sbit LCD_RS at RB2_bit;
sbit LCD_EN at RB3_bit;
sbit LCD_D4 at RB4_bit;
sbit LCD_D5 at RB5_bit;
sbit LCD_D6 at RB6_bit;
sbit LCD_D7 at RB7_bit;
// направление выводов
sbit LCD_RS_Direction at TRISB2_bit;
sbit LCD_EN_Direction at TRISB3_bit;
sbit LCD_D4_Direction at TRISB4_bit;
sbit LCD_D5_Direction at TRISB5_bit;
sbit LCD_D6_Direction at TRISB6_bit;
sbit LCD_D7_Direction at TRISB7_bit;
// инициализация модуля ЖКД
Lcd_Init( );
```

Lcd_Out – эта функция выводит текст на ЖКД, начиная с позиции, которая указана как параметр функции.

Прототип:

```
void Lcd_Out(char row, char column, char *text);
```

Параметры:

row – начальная позиция номера строки;

column – начальная позиция номера колонки;

text – текст для вывода. Может быть строковая переменная или текст, заключенный в двойные кавычки.

Пример:

```
// вывести текст Hello! на ЖКД, начиная со строки 2,
// колонка 1
Lcd_Out(2, 1, “Hello!”);
```

Lcd_Out_Cp – эта функция выводит текст на ЖКД, начиная с текущей позиции курсора.

Прототип:

```
void Lcd_Out_Cp(char *text);
```

Параметр:

text – текст для вывода. Может быть строковая переменная или текст, заключенный в двойные кавычки.

Пример:

// вывести на ЖКД текст Student с текущей позиции курсора

Lcd_Out_Cp(“Student”);

Lcd_Chr – эта функция выводит символ в позицию, которая указана как параметр.

Прототип:

void Lcd_Chr(char row, char column, char out_char);

Параметры:

row – начальная позиция номера строки;

column – начальная позиция номера колонки;

out_char – символ для вывода. Может быть строковая переменная или литерал, заключенный в одиночные кавычки.

Пример:

// вывести символ d в строку 2, колонка 3

Lcd_Chr(2, 3, ‘d’);

Lcd_Chr_Cp – эта функция выводит символ на ЖКД в текущую позицию курсора.

Прототип:

void Lcd_Chr_Cp(char out_char);

Параметр:

out_char – символ для вывода. Может быть строковая переменная или литерал, заключенный в одиночные кавычки.

Пример:

// вывести символ m в текущую позицию курсора.

Lcd_Chr_Cp(‘m’);

Lcd_Cmd – эта функция посылает команду в ЖКД.

Прототип:

void Lcd_Cmd(char out_char);

Параметр:

out_char – команда для вывода.

При выполнении данного курсового проекта будут использованы только 2 команды:

_LCD_CLEAR – очистка дисплея;

_LCD_CURSOR_OFF – отключение отображения курсора.

Пример:

```
    // очистить ЖКД
Lcd_Cmd(_LCD_CLEAR);
    // отключить отображение курсора
Lcd_Cmd(_LCD_CURSOR_OFF);
```

2.11.3 Библиотечные функции для работы с клавиатурой (Keypad Library functions)

Компилятор mikroC PRO поддерживает библиотеку для работы с клавиатурой формата 4 × 4. Библиотечные функции также могут быть использованы для клавиатур формата 4 × 3, 4 × 2 или 4 × 1.

Keypad_Init – эта функция инициализирует порт МК для работы с клавиатурой.

Прототип:

```
  void  Keypad_Init( );
```

Требования: порт для подключения клавиатуры – глобальная переменная keypadPort; должен быть определен до использования этой функции.

Пример:

```
// присоединение модуля клавиатуры к порту D микроконтроллера
char  keypadPort  at  PORTD;   // определение глобальной переменной
      ……
Keypad_Init( );                  // инициализация порта для работы
```

Keypad_Key_Press – функция читает номер нажатой клавиши в момент ее нажатия.

Прототип:

```
  char  Keypad_Key_Press( );
```

Возвращает: номер нажатой клавиши (1…16). Если нет нажатой клавиши, возвращает 0.

Пример:

```
    // прочитать номер нажатой клавиши
char  kp;              // объявление переменной для хранения номера
……..
kp = Keypad_Key_Press( );
```

Keypad_Key_Click – функция читает номер нажатой клавиши.

Прототип:

```
char  Keypad_Key_Click( );
```

Возвращает: номер нажатой клавиши (1…16). Если ни одна клавиша не нажата, возвращает 0.

Описание. Вызов функции Keypad_Key_Click( ) – это блокирующий вызов: функция ждет, пока какая-то клавиша не будет нажата и затем отпущена. Когда клавиша отпущена, функция возвращает код от 1 до 16, в зависимости от клавиши. Если одновременно нажато более одной клавиши, функция ждет освобождения всех нажатых клавиш. После этого функция возвращает номер первой нажатой клавиши.

Пример:

```
   // прочитать номер нажатой клавиши
char  kp;                  // объявление переменной для хранения номера
…….
kp = Keypad_Key_Click( );
```

2.11.4 Библиотечные функции управления модулем ШИМ (PWM Library functions)

Микроконтроллер PIC16F877 имеет два модуля ШИМ, которые обозначаются как PWM1 и PWM2 (PWM – Pulse Width Modulation). Выходной сигнал модуля PWM1 выводится на линию RC2 порта C, а модуля PWM2 – на линию RC1.

Рассмотрим библиотечные функции управления на примере модуля PWM1. Функции модуля PWM2 будут аналогичны, нужно только заменить цифру 1 на цифру 2.

PWM1_Init – эта функция инициализирует модуль ШИМ с величиной рабочего цикла (коэффициента заполнения), равной 0.

Прототип:

```
void  PWM1_Init(long  frequency);
```

Параметр:

frequency – это желаемая частота импульсов ШИМ в герцах.

Максимальная частота ШИМ определяется тактовой частотой f_{OSC} работы микроконтроллера. Для выбора можно ориентировочно воспользоваться данными таблицы 2.2.

Таблица 2.2 – Максимальная частота импульсов ШИМ

Тактовая частота, МГц	Максимальная частота ШИМ, Гц
18–20	60000
16–18	50000
14–16	40000
12–14	30000
10–12	25000
8–10	20000
6–8	15000
4–6	10000
2–4	5000

Пример:

```
   // инициализировать модуль ШИМ с частотой 5000 Гц
PWM1_Init(5000);
```

PWM1_Set_Duty – функция устанавливает значение рабочего цикла (коэффициента заполнения) ШИМ-сигнала.

Прототип:

```
void PWM1_Set_Duty(unsigned char duty_ratio);
```

Параметр:

duty_ratio – значение рабочего цикла. Может изменяться от 0 до 255 (0…0xFF).

Если задается величина рабочего цикла DC в процентах, то значение duty_ratio может быть вычислено как:

$$duty\_ratio = (DC \cdot 255) / 100.$$

Пример:

```
   // установить рабочий цикл DC, равный 75 %
   // имеем  duty_ratio = (75 · 255) / 100 = 192
PWM1_Set_Duty(192);
```

PWM1_Start – функция активизирует модуль ШИМ.

Прототип:

```
void PWM1_Start( );
```

Пример:

```
   // активизировать модуль PWM1
PWM1_Start( );
```

PWM1_Stop – эта функция останавливает работу модуля ШИМ.

Прототип:

void PWM1_Stop( );

Пример:

```
// прекратить работу модуля ШИМ
PWM1_Stop( );
```

Примечание. Следует иметь в виду, что в PIC-микроконтроллерах для работы модулей ШИМ используется таймер TMR2, поэтому невозможно задать различные частоты импульсов ШИМ для различных модулей.

2.11.5 Библиотечные функции генерации звуков (Sound Library functions)

Компилятор mikroC PRO имеет библиотеку, обеспечивающую пользователей функциями для генерации звука.

Sound_Init – функция конфигурирует выбранный вывод микроконтроллера для генерации звука.

Прототип:

void Sound_Init(char *snd_port, char snd_pin);

Параметры:

snd_port – адрес выходного порта;

snd_pin – вывод порта для генерации звука.

Пример:

```
// инициализировать вывод RC3 порта C для генерации звука
Sound_Init(&PORTC, 3);
```

Sound_Play – функция генерирует прямоугольный периодический сигнал (меандр) на соответствующем выводе порта.

Прототип:

void Sound_Play(unsigned freq_in_hz, unsigned duration_ms);

Параметры:

freq_in_hz – частота звука в герцах (Гц);

duration_ms – длительность звучания в миллисекундах (мс), диапазон от 1 до 65535 мс.

Пример:

```
// генерировать звук частоты 1 кГц в течение 100 мс
Sound_Play(1000, 100);
```

2.11.6 Встроенные функции mikroC PRO

Компилятор mikroC PRO имеет две очень полезные функции, которые генерируются непосредственно в месте их вызова в программе.

Delay_us – эта функция реализует программную задержку на определенное число микросекунд, задаваемое константой.

Прототип:

void Delay_us(const unsigned long time_in_us);

Параметр:

time_in_us – длительность задержки в микросекундах.

Пример:

```
Delay_us(500);       // задержка на 500 мкс.
```

Delay_ms – функция реализует программную задержку на определенное число миллисекунд, задаваемое константой.

Прототип:

void Delay_ms(const unsigned long time_in_ms);

Параметр:

time_in_ms – длительность задержки в миллисекундах.

Пример:

```
Delay_ms(1000);     // задержка на 1000 мс (1 с).
```

2.12 Примеры разработки программного обеспечения МКС

2.12.1 Разработка блок-схемы алгоритма главной программы работы МКС

Программы на языке Си оформляются как функции. Каждая программа обязательно содержит одну функцию, называемую главной – main( ). Выполнение программы всегда начинается с главной функции, и программа находится внутри нее.

За исключением главной функции, в некотором смысле функции языка Си похожи на подпрограммы языка Ассемблер. Они используются аналогичным образом, в основном, для хранения программных действий. Для того чтобы иметь хорошую структуру программы, нужно стремиться поместить как можно больше программного кода внутри функций. Главная же функция будет вызывать вложенные функции.

Алгоритм работы проектируемой МКС был приведен на рисунке 1.2. По нему составим блок-схемы алгоритма (БСА) главной программы работы МКС, которая на языке Си будет иметь имя main(). БСА этой программы приведена на рисунке 2.13.

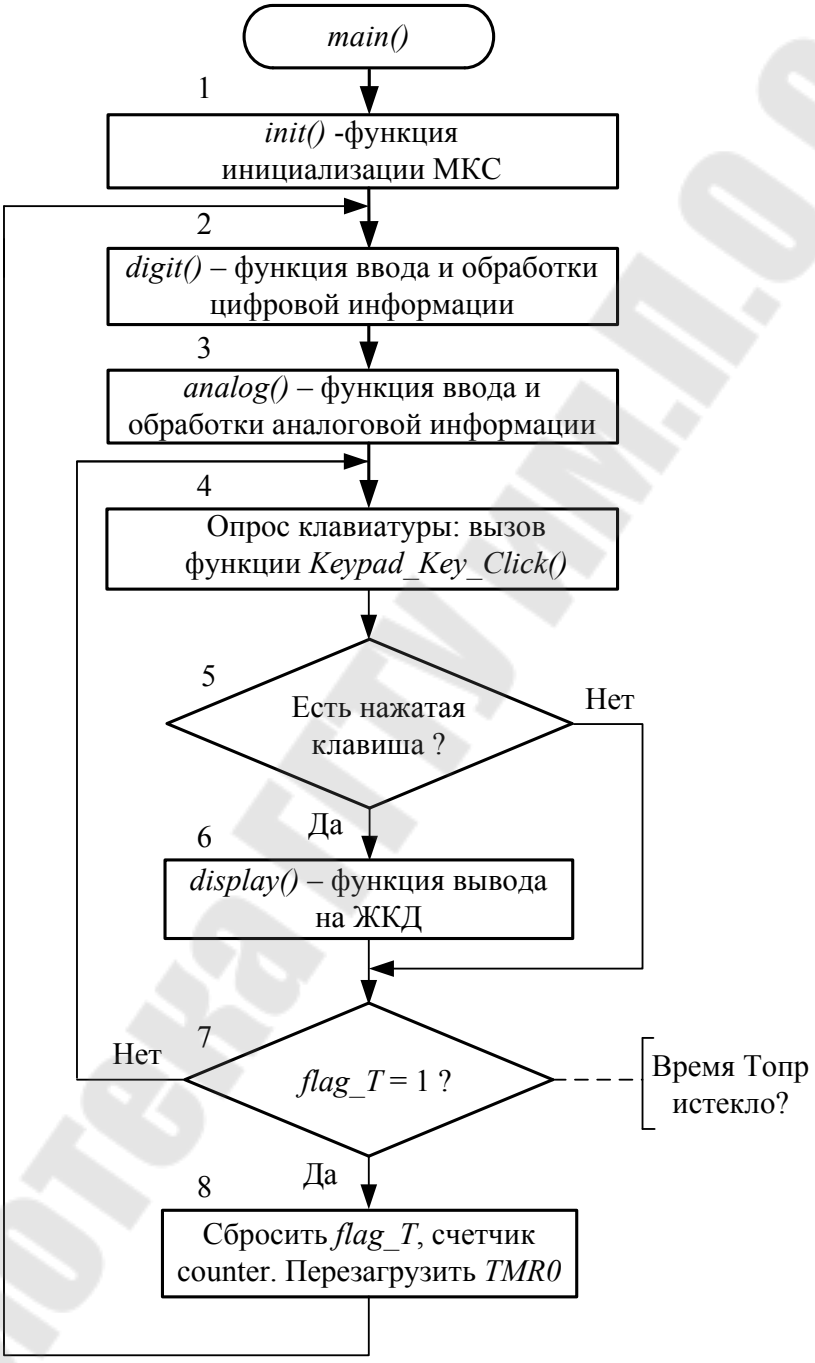

Рисунок 2.13 – БСА главной программы main( )

Главная программа main( ) состоит из нескольких функций, которые вызываются по мере надобности согласно алгоритму работы МКС. Такое построение упрощает структуру программы, делает ее более по-

нятной. Кроме того, использование функций облегчает разработку и отладку всей программы.

Согласно БСА, на рисунке 2.13 в главную программу входят следующие функции:

init( ) – функция инициализации МКС;

digit( ) – функция ввода и обработки цифровой информации;

analog( ) – функция ввода и обработки аналоговой информации;

display( ) – функция вывода данных на дисплей.

Кроме этих функций в программу работы МКС будет входить функция обработки прерывания interrupt( ), которая обрабатывает запросы прерывания от таймера и аварийного датчика.

В блоке *7* алгоритма главной программы проверяется значение переменной-флага с именем flag_T. Этот флаг устанавливает функция обработки прерывания interrupt( ). Если значение flag_T = 0, то время отсчета $T_{ОПР}$ еще не окончено. По окончании отсчета времени $T_{ОПР}$ обработчик прерывания устанавливает флаг flag_T = 1, что является для главной программы сигналом к завершению цикла ожидания.

Рассмотрим возможную реализацию функций, составляющую программу работы МКС. Будем считать, что заданы следующие параметры: тактовая частота работы микроконтроллера f_{OSC} = 4 МГц; время опроса $T_{ОПР}$ = 2 с.

2.12.2 Функция инициализации init( )

Функция init( ) используется для инициализации МКС после включения электропитания или сброса МК при помощи кнопки «Сброс», расположенной на пульте управления. БСА этой функции приведена на рисунке 2.14.

Сначала необходимо настроить линии портов МК на ввод или вывод согласно требованиям подключенных к ним внешних устройств. Далее нужно вывести нулевые начальные значения управляющих сигналов Y1, Y2, Y3, Y4, Y5. Затем необходимо произвести инициализацию ЖКД, клавиатуры, модулей АЦП и ШИМ, а также канала звука по методике, изложенной в описании библиотечных функций mikroC PRO для этих устройств. После инициализации ЖКД надо вывести на дисплей текст “Work Mode”, сообщающий о начале работы МКС после запуска микроконтроллера. Заключительным этапом инициализации является настройка таймера TMR0 на заданный режим работы и настройка системы прерывания.

Таймер TMR0 будет использоваться в программе для отсчета времени опроса $T_{ОПР}$, при этом он считает импульсы внутреннего тактового генератора. Счетные импульсы поступают с частотой, определяемой частотой системной синхронизации и коэффициентом предварительного деления, задаваемым регистром специальных функций OPTION_REG [1], [6].

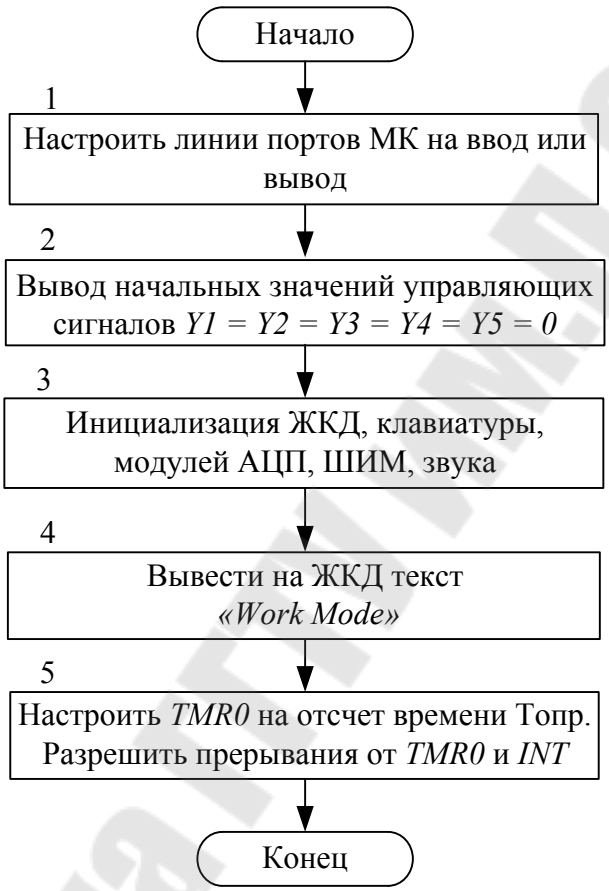

Рисунок 2.14 – БСА функции инициализации init( )

Формат регистра OPTION_REG следующий:

7	6	5	4	3	2	1	0
RBPU	INTEDG	T0CS	T0CE	PSA	PS2	PS1	PS0

Коэффициент деления выбирают с помощью битов PS2, PS1, PS0 в диапазоне от 2 до 256, как показано в таблице 2.3. Например, при использовании тактовой частоты $f_{OSC} = 4$ МГц период тактовых импульсов будет $T_{OSC} = 1/f_{OSC} = 0{,}25$ мкс; командный (машинный) цикл будет $4 \cdot T_{OSC} = 1$ мкс (частота тактирования внутренне делится на 4). Таким образом, если, например, выбрать коэффициент деления 16, то содер-

жимое счетного регистра таймера будет увеличиваться через каждые 16 мкс.

При переполнении счетного регистра (т. е. переходе от 255 к 0) формируется запрос прерывания от таймера (если оно разрешено). Загружая некоторое значение в регистр TMR0, можно управлять количеством отсчетов до получения переполнения.

Таблица 2.3 – Выбор коэффициента деления с помощью регистра OPTION_REG

PS2	PS1	PS0	Коэффициент деления
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	256

Для вычисления времени до момента переполнения таймера можно воспользоваться следующей формулой:

$$T_{OV} = 4 \cdot T_{OSC} \cdot K \cdot (256 - TMR0), \qquad (2.3)$$

где T_{OV} – время до момента переполнения (мкс); T_{OSC} – период следования импульсов тактового генератора, мкс; K – коэффициент деления частоты, выбранный с помощью регистра OPTION_REG; TMR0 – значение, загруженное в регистр TMR0 (десятичное число).

Из формулы (2.3) можно определить, что максимальное время переполнения, а следовательно, и время задержки при тактовой частоте 4 МГц, т. е. T_{OSC} = 0,25 мкс, коэффициенте деления K = 256, начальном значении TMR0 = 0 будет:

$$T_{OV} = 4 \cdot 0{,}25 \cdot 256 \cdot (256 - 0) = 65536 \text{ мкс}.$$

Это значение значительно меньше требуемого времени $T_{ОПР}$ = 2 с. Поэтому на таймере реализуем задержку на 50 мс, а в переменной, например, с именем counter (счетчик) будем накапливать количество переполнений с целью получения требуемой задержки в 2 с. Необходимое количество переполнений для отсчета времени 2 с будет:

$$2000 \text{ мс}/50 \text{ мс} = 40.$$

Для определения значения, которое должно быть загружено в регистр TMR0 с целью получения заданного времени переполнения, формулу (2.3) удобно записать в виде:

$$TMR0 = 256 - T_{OV}/(4 \cdot 0{,}25 \cdot K). \qquad (2.4)$$

Для получения времени 50 мс при тактовой частоте 4 МГц и коэффициенте деления 256 из формулы (2.4) имеем:

$$TMR0 = 256 - 50000/(4 \cdot 0{,}25 \cdot 256) = 60{,}69 \approx 61.$$

Таким образом, в регистр TMR0 необходимо загрузить десятичное число 61. Для получения коэффициента деления К = 256 в младшие 3 разряда регистра OPTION_REG надо записать двоичное число 111. Кроме того, необходимо установить бит INTEDG = 1 для того, чтобы прерывание по входу INT микроконтроллера от аварийного датчика возникало при изменении сигнала X0 от “0” к “1”. Таким образом, в регистр OPTION_REG необходимо загрузить число 0b01000111 = 0x47.

В функции init( ) необходимо также разрешить прерывания при переполнении таймера TMR0 и по сигналу на входе INT от аварийного датчика.

В микроконтроллерах семейства PIC16 управление прерываниями реализовано с помощью регистров специальных функций INTCON, PIE и PIR, а программа – обработчик прерываний всегда начинает исполняться с адреса 0004h [1], [6].

Формат регистра INTCON следующий:

7	6	5	4	3	2	1	0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

GIE – это бит общего разрешения прерываний. Если он установлен в “1”, то все немаскируемые прерывания разрешены. Если он сброшен в “0”, то все прерывания запрещены.

PEIE используется в качестве бита разрешения всех прерываний от периферийных устройств, определяемых с помощью регистров PIE и PIR. В данном курсовом проекте не предусмотрены прерывания от периферийных устройств, поэтому этот бит равен 0.

T0IE является битом разрешения (логическая 1) или запрещения (логический 0) прерывания при переполнении таймера TMR0.

Бит T0IF является флагом запроса прерывания при переполнении TMR0 (T0IF = 1 – есть запрос прерывания; T0IF = 0 – нет запроса).

INTE – это бит разрешения внешнего прерывания по входу INT, а бит INTF – это флаг запроса прерывания по этому входу.

RBIE – это бит разрешения прерывания при изменении уровней сигналов на входах линий RB4, RB5, RB6, RB7 порта B, а бит RBIF – это флаг соответствующего запроса прерывания.

В данном курсовом проекте имеются два источника прерывания – это таймер TMR0 и сигнал по входу INT, который поступает от аварийного датчика. Поэтому для разрешения прерываний от них необходимо в регистр INTCON загрузить число 0b10110000 = 0xB0.

Рассмотрим пример реализации функции инициализации МКС init( ) на языке mikroC:

```
        /* Объявления глобальных переменных, используемых в  init( )  */
          // присоединение выводов ЖКД
sbit  LCD_RS  at  RB2_bit;
sbit  LCD_EN  at  RB3_bit;
sbit  LCD_D4  at  RB4_bit;
sbit  LCD_D5  at  RB5_bit;
sbit  LCD_D6  at  RB6_bit;
sbit  LCD_D7  at  RB7_bit;
        // направление выводов
sbit  LCD_RS_Direction  at  TRISB2_bit;
sbit  LCD_EN_Direction  at  TRISB3_bit;
sbit  LCD_D4_Direction  at  TRISB4_bit;
sbit  LCD_D5_Direction  at  TRISB5_bit;
sbit  LCD_D6_Direction  at  TRISB6_bit;
sbit  LCD_D7_Direction  at  TRISB7_bit;
        // присоединение клавиатуры к порту D микроконтроллера
char  keypadPort  at  PORTD;

void  init( )                  // функция инициализации
        {
                  // настройка линий портов
    TRISA = 0xFF;
    TRISB = 0b11111101;
    TRISC = 0b11111000;
    TRISE = 0;
    TRISD = 0xFF;
         // вывод начальных значений Y1 = Y2 = Y3 = Y4 = Y5 = 0
    PORTE = 0;
    PORTC.B0 = 0;
    PORTC.B1 = 0;
    PORTC.B2 = 0;
```

```
    PORTB.B1 = 0;
        // инициализация ЖКД
    Lcd_Init( );
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Cmd(_LCD_CURSOR_OFF);
    Lcd_Out(1, 4, “Work  Mode”);
        // инициализация модуля АЦП
    ADC_Init( );
       // инициализация клавиатуры
    Keypad_Init( );
      // инициализация модулей ШИМ
    PWM1_Init(5000);
    PWM2_Init(5000);
    PWM1_Start( );
    PWM2_Start( );
      // инициализация линии RC0 порта C для генерации звука
    Sound_Init(&PORTC, 0);
      // инициализация таймера TMR0
    OPTION_REG = 0x47;
    TMR0 = 61;
      // разрешить прерывания от TMR0 и INT
    INTCON = 0xB0;
}
```

2.12.3 Функция interrupt( )

Функция interrupt( ) является обработчиком прерывания при поступлении запроса прерывания INT (сигнал X0 от аварийного датчика) или переполнении таймера TMR0. Функция также производит отсчет времени опроса $T_{ОПР}$.

Блок-схема алгоритма обработчика прерывания interrupt( ) приведена на рисунке 2.15. Так как прерывание по INT должно иметь более высокий приоритет, то программа вначале проверяет флаг запроса внешнего прерывания INTF. Если флаг INTF установлен (поступил сигнал X0 от аварийного датчика), то программа переходит на выполнение блоков *2*, *3* и *4* аварийной сигнализации. После вывода мигающего светового сигнала с частотой 1 Гц программа в блоке *4* зацикливается. При этом прекращается выполнение каких-либо других действий. Для выхода из аварийного режима нужно отключить питание МК или нажать кнопку «Сброс» на пульте управления.

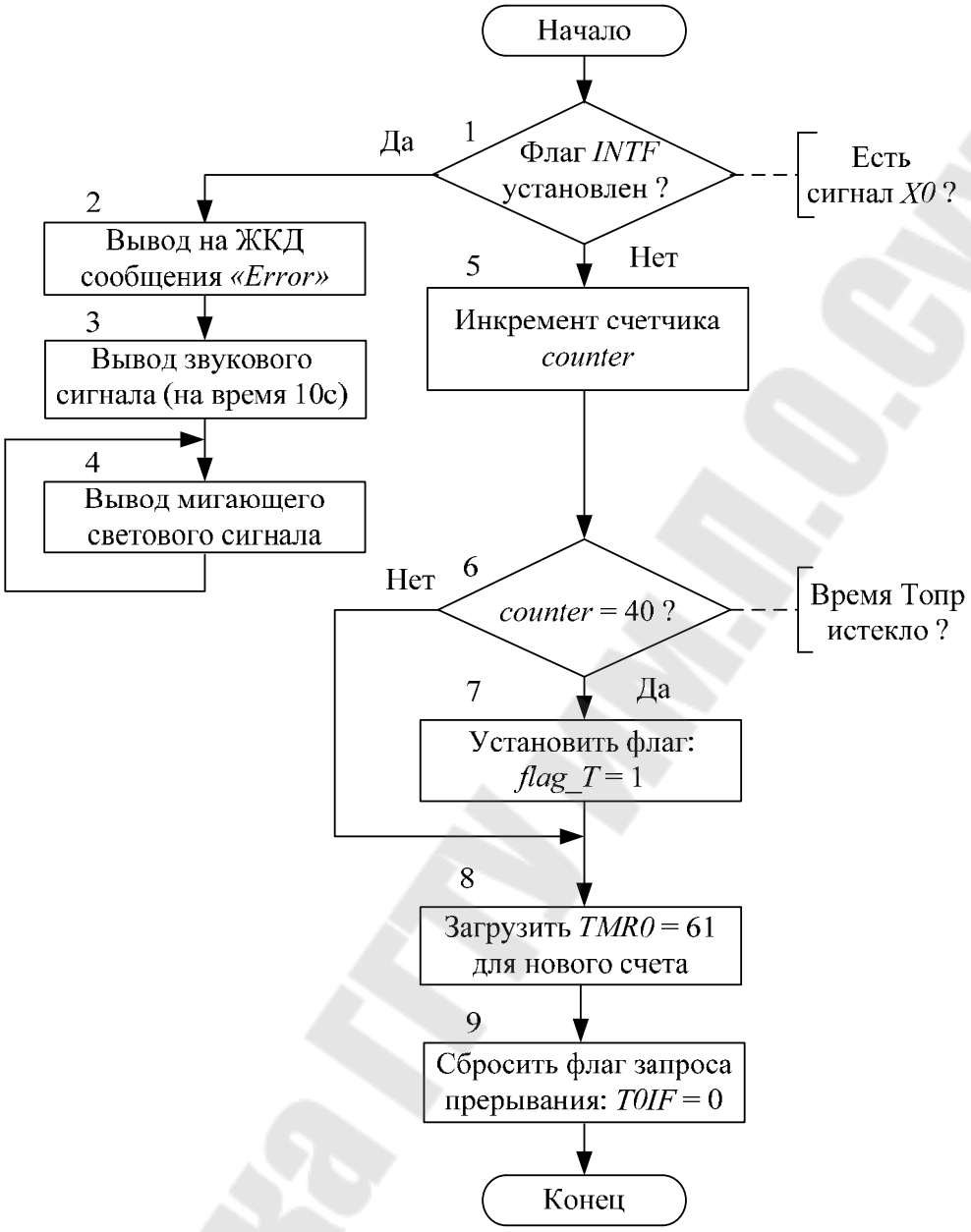

Рисунок 2.15 – БСА функции interrupt( )

Если флаг запроса внешнего прерывания INTF сброшен, то это означает, что запрос прерывания пришел от таймера TMR0 (установлен флаг T0IF). Программа переходит к обслуживанию прерываний от таймера. При каждом прерывании вследствие переполнения таймера (прошло время – 50 мс) инкрементируется переменная-счетчик counter. Если содержимое counter станет равным 40, то это означает, что истекло время опроса $T_{ОПР}$ = 2 с (блок *6*). После окончания отсчета времени 2 с функция interrupt( ) устанавливает в 1 переменную-флаг flag_T, что является сигналом для главной программы main( ) начать новый цикл опроса датчиков и выполнения других функций

управления. Если же время опроса не закончилось, то в блоке *8* в счетчик таймера TMR0 загружается число 61 для нового отсчета 50 мс. Перед выходом из обработчика в блоке *9* выполняется сброс флага запроса прерывания то таймера TMR0, после чего МК будет вновь реагировать на переполнение таймера TMR0.

Рассмотрим пример реализации обработчика прерывания interrupt( ):

```
/* Определение глобальных переменных, используемых в функции */
char  counter = 0;              // счетчик переполнений таймера TMR0
char  flag_T = 0;               // флаг окончания счета времени опроса – 2 с

void  interrupt( )   // функция – обработчик прерываний от TMR0 и INT
{
   if(INTF == 1)            // если запрос прерывания по INT
   {
      Lcd_Cmd(_LCD_CLEAR);          // очистить ЖКД
      Lcd_Out(1, 5, “ERROR”);       // вывод на дисплей
                                    // сообщения об аварии
      Sound_Play(500,  10000);      // вывод звукового сигнала
                            // частоты 500 Гц длительностью 10 с
      while (1)             // вывод мигающего светового сигнала и
      {                             // зацикливание программы
         PORTB.B1 = 1;              // включить светодиод
         Delay_ms(500);             // задержка на 500 мс
         PORTB.B1 = 0;              // выключить светодиод
         Delay_ms(500);
      }
   }
       // запрос прерывания от таймера TMR0
   counter++;
   if(counter == 40)        // если прошло время опроса – 2 с
   {
      flag_T = 1;           // установить флаг окончания отсчета
                            // времени T_ОПР = 2 с
      counter = 0;
   }
   TMR0 = 61;               // перезагрузить таймер
   T0IF_bit = 0;            // сбросить флаг запроса прерывания T0IF
}
```

2.12.4 Функция digit( )

Функция digit( ) выполняет обработку цифровых сигналов X1–X5, поступающих в МКС от двоичных датчиков. БСА обработки цифровой информации была приведена на рисунке 1.3.

Рассмотрим программную реализацию алгоритма на примере вычисления логической функции двух переменных:

$$f(X1, X2) = \overline{X1} \wedge \overline{(X1 \vee X2)},$$

где $\wedge$ – знак операции логического умножения (операции И); $\vee$ – знак операции логического сложения (операции ИЛИ); $\overline{\ }$ – знак операции логического отрицания (операции НЕ).

Входные сигналы X1, X2 поступают на линии RC3, RC4 порта C, настроенные на ввод. Выходной сигнал Y1 вырабатывается на линии RE0 порта E, настроенной на вывод. Для формирования одиночного управляющего импульса Y1 длительностью, к примеру, t1 = 80 мкс, будем использовать библиотечную функцию задержки Delay_us.

Пример:

```
/* Объявление глобальных переменных, используемых в функции */
sbit  X1  at  RC3_bit;   // битовая переменная X1 на линии порта RC3
sbit  X2  at  RC4_bit;   // битовая переменная X2 на линии порта RC4
sbit  Y1  at  RE0_bit;   // битовая переменная Y1 на линии порта RE0

void  digit( )           // функция обработки цифровой информации
{
   bit  f;               // битовая переменная для хранения результата
                         // вычисления логической функции
   f = (!X1) && ( ! (X1 || X2));  // вычисление логической функции
   if(f == 1)          // если результат равен 1
   {
      Y1 = 1;
      Delay_us(80);
      Y1 = 0;
   }
}
```

2.12.5 Функция analog( )

Функция analog( ) выполняет ввод и обработку аналоговой информации, поступающей в МКС от датчиков. Блок-схема алгоритма этой программы была приведена на рисунке 1.5.

С помощью АЦП, входящего в состав микроконтроллера, аналоговые сигналы U1–U5 преобразуются в 10-разрядные двоичные коды W1–W5. Эти коды должны храниться в ячейках памяти данных (ПД) для дальнейшего использования в программе. Далее вычисляется функция g(W1,W2,W3,K1,K2,K3), которая затем сравнивается с константой Q (уставкой), хранящейся в программной памяти МК.

В зависимости от результата сравнения вырабатываются управляющие сигналы Y2 или Y3, представляющие собой одиночные импульсы длительностью t2 или t3. В блоке *10* вычисляется функция e(W4,K4), которая используется для формирования сигнала Y4 на выводе RC2 микроконтроллера, представляющегося собой импульсную последовательность ШИМ. В блоке *12* вычисляется функция h(W5,K5), которая используется для формирования импульсной последовательности ШИМ на выводе RC1. Из нее путем фильтрации получается сигнал Y5, поступающий на исполнительные устройства. По условию коэффициенты K1–K5, Q – это двухбайтные целые числа, хранящиеся в программной памяти микроконтроллера.

Рассмотрим пример реализации функции analog( ). Предположим, что требуется вычислить функции:

$$g( ) = K1 \cdot W1 + 5 \cdot W2 – K2;$$

$$e( ) = W4 / K4;$$

$$h( ) = W5 / K5.$$

Коэффициенты: K1 = 20; K2 = 100; K3 = 45; K4 = 4; K5 = 8; Q = 15.

Длительность импульсов: t2 = 75 мс; t3 = 24 мс.

Пример:

```
/* Глобальные переменные, используемые в функции analog( ) */
// переменные для хранения кодов АЦП
unsigned int W1;
unsigned int W2;
unsigned int W3;
unsigned int W4;
unsigned int W5;
```

```
    // размещение констант в памяти программ
code  const  int  K1 = 10;
code  const  int  K2 = 100;
code  const  int  K3 = 45;
code  const  int  K4 = 4;
code  const  int  K5 = 8;
code  const  int Q = 15;
sbit  Y2  at  RE1_bit;  // битовая переменная Y2 на линии порта RE1
sbit  Y3  at  RE2_bit;  // битовая переменная Y3 на линии порта RE2

void  analog( )                    // функция analog( )
{
   int  g;                 // двухбайтные переменные для хранения
   int  e;                 // результатов вычисления
   int  h;                 // функций  g( ),  e( ), h( )
   char  DC;               // переменная для рабочего цикла ШИМ
    // чтение кодов АЦП
   W1 = ADC_Read(0);     // аналоговый канал 0 (сигнал U1)
   W2 = ADC_Read(1);     // аналоговый канал 1 (сигнал U2)
   W3 = ADC_Read(2);     // аналоговый канал 2 (сигнал U3)
   W4 = ADC_Read(3);     // аналоговый канал 3 (сигнал U4)
   W5 = ADC_Read(4) ;     // аналоговый канал 4 (сигнал U5)

   g = K1 * W1 + 5 * W2 – K2;    // вычисление функции g( )
   if(g >= Q)
   {
      Y3 = 1;                          // вывод сигнала Y3
      Delay_ms(24);
      Y3 = 0;
    }
    else
    {
       Y2 = 1;                         // вывод сигнала Y2
       Delay_ms(75);
       Y2 = 0;
    }
    e = W4 / K4;                       // вычисление функции  e( )
    DC = e;                      // определить величину рабочего цикла
    PWM1_Set_Duty(DC);         // установить текущий рабочий
                                 // цикл для PWM1
```

```
    h = W5 / K5;                    // вычисление функции  h( )
    DC = h;                  // определить величину рабочего цикла
    PWM2_Set_Duty(DC);       // установить текущий рабочий
                               // цикл для PWM2
}
```

Отметим, что в некоторых заданиях для выполнения курсового проекта требуется вычислять функции g( ) вида:

$$g( ) = \min \{5 \cdot W1 + W2;\ W3 - K3\}, \qquad (2.5)$$

или

$$g( ) = \max \{5 \cdot W1 + W2;\ W3 - K3\}. \qquad (2.6)$$

Функцию g( ) в формуле (2.5) можно условно записать в следующем виде:

$$g( ) = \min \{N; M\}.$$

Сначала следует вычислить выражения $N = 5 \cdot W1 + W2$ и $M = W3 - K3$, которые будут двухбайтными целыми числами. Затем нужно их сравнить между собой с целью определения, какое из них является меньшим. Оно и будет являться искомым значением функции g( ). Например, если $N < M$, то $g( ) = N$.

Во втором случае формулу (2.6) можно условно записать в следующем виде:

$$g( ) = \max \{N; M\}.$$

После вычисления выражений $N = 5 \cdot W1 + W2$ и $M = W3 - K3$ их нужно сравнить между собой с целью определения, какое из них является большим. Оно и будет являться искомым значением функции g( ).

2.12.6 Функция display( )

Функция display( ) используется для обслуживания пульта управления. Блок-схема алгоритма этой функции приведена на рисунке 2.16.

В блоках *1*, *4*, *7*, *10*, *13* по значению переменной key, в которой хранится номер (1–5), анализируется, какая клавиша была нажата оператором. В зависимости от нажатой клавиши (SB1–SB5) производится вычисление напряжения U_i в вольтах по его коду W_i.

Модуль АЦП в микроконтроллере PIC16F877 выдает 10-разрядный двоичный код [6]. Диапазон изменения выходного кода будет 0000h…03FFh, что соответствует входному однополярному напряже-

нию в диапазоне 0…+5000 мВ. Чтобы найти напряжение U_i, соответствующее коду W_i, составим пропорцию:

$$U_i, \text{ мВ} – W_i;$$

$$5000, \text{ мВ} – 03FF.$$

Откуда получим:

$$U_i = (W_i \cdot 5000)/03FF, \text{ мВ}. \qquad (2.7)$$

Полученное по формуле (2.7) значение напряжения U_i будет в милливольтах. Для отображения на дисплее значения U_i в вольтах можно находить цифры путем целочисленного деления результата последовательно на 1000, 100 и 10.

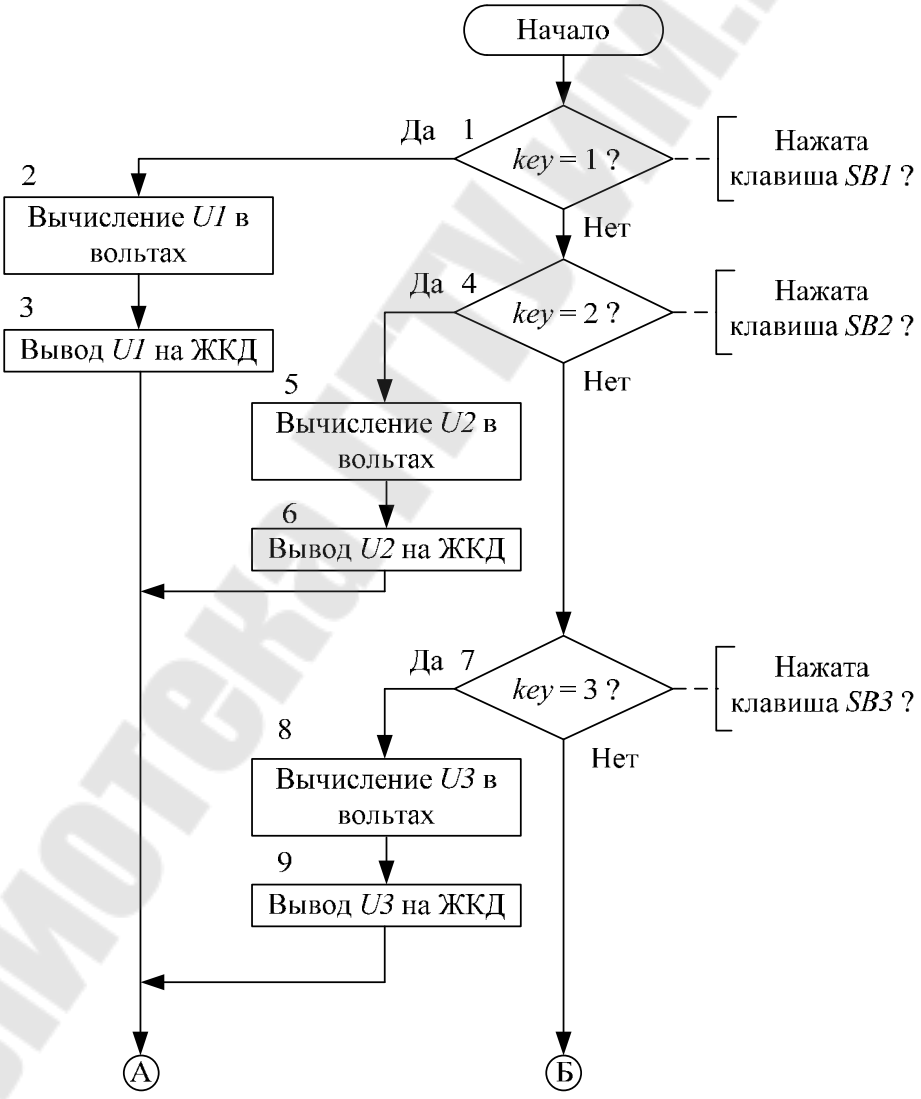

Рисунок 2.16 – БСА функции display( )
(начало, окончание на с. 46)

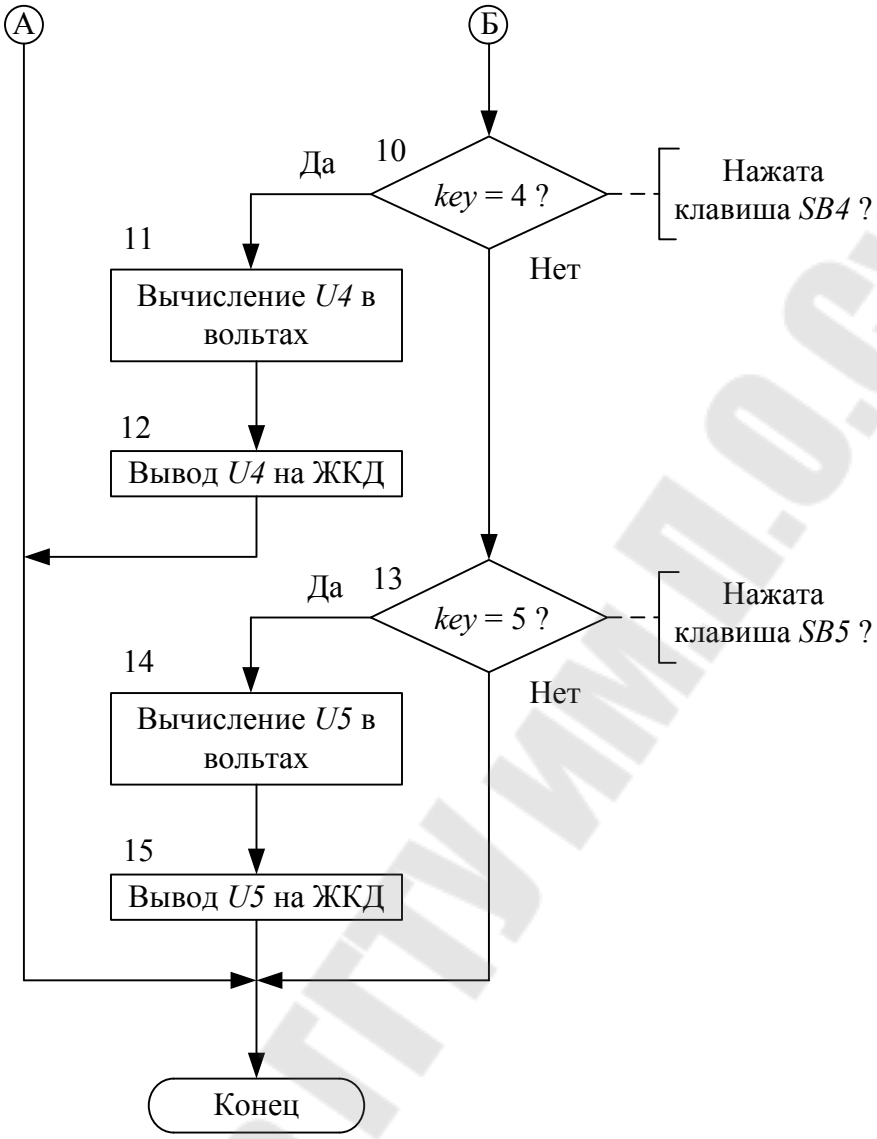

Рисунок 2.16 – БСА функции display( )
(окончание, начало на с. 45)

Рассмотрим пример реализации функции display( ):

```
/* Глобальные переменные, используемые в функции display( ) */
unsigned int W1, W2, W3, W4, W5;      // коды АЦП
char key;                             // номер нажатой клавиши

void display( )                       // функция display( )
{
   int mvolts;          // переменная для хранения величины
                        // напряжения в милливольтах
   char num;            // переменная для хранения цифр
                        // напряжения в вольтах
```

```
switch(key)
{
   case  1:              // нажата клавиша SB1 – “U1”
   {
      mvolts = ((long)W1 * 5000) / 0x03FF;   // преобразование
                        // кода W1 в милливольты
      Lcd_Cmd(_LCD_CLEAR);      // очистить дисплей
      Lcd_Out(1, 4, “Work  Mode”);   // вывод текста сообщения
                                          // в 1-ю строку
      Lcd_Out(2, 4, “U1=”);        // вывод текста во 2-ю строку
      num = mvolts / 1000;   // извлечение единиц вольт
      Lcd_Chr_Cp(48 + num);  // вывод цифры в коде ASCII
      Lcd_Chr_Cp(‘.’);         // вывод на ЖКД десятичной точки
      num = (mvolts / 100) % 10;  // извлечение десятых
                                    // долей вольта
      Lcd_Chr_Cp(48 + num);   // вывод цифры в коде ASCII
      num = (mvolts / 10) % 10;   // извлечение сотых
                                    // долей вольта
      Lcd_Chr_Cp(48 + num);   // вывод цифры в коде ASCII
      num = mvolts % 10;           // извлечение тысячных
                                    // долей вольта
      Lcd_Chr_Cp(48 + num);   // вывод цифры в коде ASCII
      Lcd_Out_Cp(“ V”);          // вывод текста в текущую
                                 // позицию курсора
      break;
   }
   case  2:                      // нажата клавиша  SB2 – “U2”
   {
      ……...                     // как в случае  case 1,
      ……..                      // но для сигнала  U2
      break;
   }
   case  3:                     // нажата клавиша  SB3 – “U3”
   {
      ……..                      // как в случае  case 1,
      …….                       // но для сигнала  U3
      break;
   }
   case  4:                     // нажата клавиша  SB4 – “U4”
```

```
        {
            ……..            // как в случае  case 1,
            ……..            // но для сигнала  U4
            break;
        }
        case  5:                    // нажата клавиша  SB5 – “U5”
        {
            ……..            // как в случае  case 1,
            ……..            // но для сигнала  U5
            break;
        }
    }
}
```

2.12.7 Программа управления МКС

Программу управления для МКС назовем control.c. Ее текст может быть следующий:

```
/*************************************************************
control.c – программа управления микроконтроллерной системой
*************************************************************/
        /* Определение глобальных переменных  */
         // присоединение выводов ЖКД
sbit  LCD_RS  at  RB2_bit;
sbit  LCD_EN  at  RB3_bit;
sbit  LCD_D4  at  RB4_bit;
sbit  LCD_D5  at  RB5_bit;
sbit  LCD_D6  at  RB6_bit;
sbit  LCD_D7  at  RB7_bit;
         // направление выводов
sbit  LCD_RS_Direction  at  TRISB2_bit;
sbit  LCD_EN_Direction  at  TRISB3_bit;
sbit  LCD_D4_Direction  at  TRISB4_bit;
sbit  LCD_D5_Direction  at  TRISB5_bit;
sbit  LCD_D6_Direction  at  TRISB6_bit;
sbit  LCD_D7_Direction  at  TRISB7_bit;
         // присоединение клавиатуры к порту D микроконтроллера
char  keypadPort  at  PORTD;
char  key;                    // переменная – номер нажатой клавиши
```

```
char  counter = 0;          // счетчик переполнений таймера TMR0
char  flag_T = 0;           // флаг окончания счета времени опроса – 2 с
      // объявление входных переменных X1–X5
sbit  X1  at  RC3_bit;
sbit  X2  at  RC4_bit;
sbit  X3  at  RC5_bit;
sbit  X4  at  RC6_bit;
sbit  X5  at  RC7_bit;
      // объявление выходных переменных Y1–Y3
sbit  Y1  at  RE0_bit;
sbit  Y2  at  RE1_bit;
sbit  Y3  at  RE2_bit;
      // переменные для хранения кодов АЦП
unsigned  int  W1;
unsigned  int  W2;
unsigned  int  W3;
unsigned  int  W4;
unsigned  int  W5;
      // размещение констант в памяти программ
code  const  int  K1 = 10;
code  const  int  K2 = 100;
code  const  int  K3 = 45;
code  const  int  K4 = 4;
code  const  int  K5 = 8;
code  const  int  Q = 15;
      // прототипы функций, входящих в  main( )
void  init( );
void  digit( );
void  analog( );
void  display( );

void  interrupt( )          // функция – обработчик прерывания
{
    ……..
    …….
}

void  main( )               // главная функция  main( )
{
```

```
    init( );                    // вызов функции инициализации
    again:
        digit( );              // вызов функции обработки
                               // цифровой информации
        analog( );             // вызов функции обработки
                               // аналоговой информации
    opros:
        key = 0;
        key = Keypad_Key_Click( );    // вызов функции опроса
                                      // клавиатуры
        if(key != 0)                  // если есть нажатая клавиша
            display( );          // вызов функции вывода на дисплей
        if(flag_T == 0)       // если отсчет времени T_ОПР не закончен
            goto  opros;
        else                   // отсчет времени T_ОПР  закончен
        {
            flag_T = 0;        // сбросить флаг  flag_T
            counter = 0;       // очистить счетчик переполнений таймера
            TMR0 = 61;         // перезагрузить таймер
            goto  again;       // идти на метку  again
        }
}

void  init( )                  // функция инициализации МКС
{
    ……..
}

void  digit( )             // функция обработки цифровой информации
{
    …….
}

void  analog( )          // функция обработки аналоговой информации
{
    ……...
}

void  display( )         // функция вывода на дисплей
```

```
{
   ……………
}
```

Программа control.c составлена на основе блок-схемы алгоритма, приведенного на рисунке 2.13. Входящие в нее функции interrupt( ), init( ), digit( ), analog( ) и display( ), а также определение глобальных переменных необходимо доработать в соответствии с заданием на курсовой проект.

3 ОФОРМЛЕНИЕ КУРСОВОГО ПРОЕКТА

Курсовой проект состоит из расчетно-пояснительной записки и графической части.

3.1 Расчетно-пояснительная записка

Расчетно-пояснительная записка должна содержать:

- Титульный лист.
- Бланк рецензии на курсовой проект.
- Задание на курсовой проект.
- Оглавление (содержание) с указанием страниц.
- Введение.
- Описание алгоритма работы МКС.
- Разработка структурной схемы МКС.
- Разработка принципиальной схемы МКС.
- Разработка программного обеспечения МКС.
- Заключение.
- Список литературы.
- Приложение.

Во введении должны быть сформулированы цели и задачи курсового проекта, кратко охарактеризовано содержание работы.

Структурная схема разрабатывается в соответствии с индивидуальным заданием. Следует дать краткое описание состава и назначения основных элементов системы.

В разделе «Разработка принципиальной схемы МКС» необходимо разработать принципиальные схемы отдельных модулей микроконтроллерной системы: схему подключения МК, модулей ввода/вывода цифровой и аналоговой информации, пульта управления. Нужно дать краткое описание микросхем, входящих в модули (задан-

ных или выбранных самостоятельно). Привести их условное изображение, назначение выводов, таблицы функционирования и т. п.

В разделе «Разработка программного обеспечения работы МКС» необходимо привести блок-схемы алгоритмов программ, дать их краткое описание. Нужно привести тексты всех программ на языке mikroC. В программах обязательно должны быть комментарии.

В заключении необходимо привести основные результаты выполнения курсового проекта, параметры разработанной МКС.

В приложении помещается перечень элементов принципиальной схемы, разработанной МКС.

Пояснительная записка оформляется компьютерными средствами на листах формата А4 (210 × 297 мм), размер шрифта 14pt, стиль Times New Roman, интервал – полуторный. Пояснительная записка должна быть переплетена и помещена в стандартную папку. Примерный объем пояснительной записки – 35–40 страниц.

3.2 Графическая часть

Графическая часть курсового проекта состоит из двух чертежей формата А1:

1 Принципиальная электрическая схема МКС.

2 Блок-схемы алгоритмов программ работы МКС.

Все элементы принципиальной схемы должны иметь нумерацию, а микросхемы – нумерацию выводов.

Примечание. Нумерация микросхем и других элементов на общей принципиальной схеме, как правило, отличается от нумераций на схемах отдельных модулей.

Графическая часть должна быть выполнена компьютерными средствами и оформлена в соответствии с требованиями ЕСКД.

Литература

1 Шпак, Ю. А. Программирование на языке С для AVR и PIC микроконтроллеров / Ю. А. Шпак. – К. : МК-Пресс ; СПб. : КОРОНА-ВЕК, 2011.

2 Уилмсхерст, Т. Разработка встроенных систем с помощью микроконтроллеров PIC. Принципы и практические примеры / Т. Уилмсхерст ; пер. с англ. – К. : МК-Пресс ; СПб. : КОРОНА-ВЕК, 2008.

3 Васильев, А. Е. Микроконтроллеры. Разработка встраиваемых приложений : учеб. пособие / А. Е. Васильев. – СПб. : Изд-во СПбГПУ, 2003.

4 Белов, А. В. Конструирование устройств на микроконтроллерах / А. В. Белов. – СПб. : Наука и техника, 2005.

5 Брей, Б. Применение микроконтроллеров PIC18. Архитектура, программирование и построение интерфейсов с применением С и ассемблера : пер. с англ./ Б. Брей. – К. : МК-Пресс ; СПб. : КОРОНА-ВЕК, 2008.

6 PIC16F877X. Однокристальные 8-разрядные микроконтроллеры компании Microchip. – М. : Микро-чип, 2002.

7 MikroC PRO for PIC. User's manual. – 2014. – Режим доступа: http://www.mikroe.com.

8 Бэйкер, Бонни. Что нужно знать цифровому инженеру об аналоговой электронике : пер. с англ. / Бонни Бэйкер. – М. : Додэка-XXI, 2010.

9 Резисторы, конденсаторы, трансформаторы, дроссели, коммутационные устройства РЭА : справочник / Н. Н. Акимов и [др.]. – Минск : Беларусь, 1994.

10 Цифровые интегральные микросхемы : справочник / М. И. Богданович и [др.]. – Минск : Беларусь, 1991.

Содержание

Учебное электронное издание комбинированного распространения

Учебное издание

Виноградов Эдуард Михайлович

ПРОЕКТИРОВАНИЕ МИКРОКОНТРОЛЛЕРНОЙ СИСТЕМЫ УПРАВЛЕНИЯ

Учебно-методическое пособие по курсовому проектированию по дисциплине «Микропроцессорная техника» для студентов специальности 1-36 04 02 «Промышленная электроника» дневной и заочной форм обучения

Электронный аналог печатного издания

Редактор *Т. Н. Мисюрова*
Компьютерная верстка *Н. Б. Козловская*

Подписано в печать 14.05.15.
Формат 60x84/16. Бумага офсетная. Гарнитура «Таймс».
Ризография. Усл. печ. л. 3,25. Уч.-изд. л. 3,38.
Изд. № 138.
http://www.gstu.by

Издатель и полиграфическое исполнение
Гомельский государственный
технический университет имени П. О. Сухого.
Свидетельство о гос. регистрации в качестве издателя
печатных изданий за № 1/273 от 04.04.2014 г.
246746, г. Гомель, пр. Октября, 48