МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ЧЕРНИГОВСКИЙ НАЦИОНАЛЬНЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРНЫХ СИСТЕМ НА БАЗЕ ЯДРА ARM7 НА ЯЗЫКЕ С

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ по дисциплине
“Микроконтроллерные системы автоматики и управления”
для студентов направления подготовки
6.050102 – “Компьютерная инженерия”

Обсуждено и рекомендовано
на заседании кафедры
информационных и компьютерных систем
Протокол № 2
от 27.06.2014 г.

Чернигов ЧНТУ 2014

Програмування мікроконтролерних систем на базі ядра ARM7 на мові С. Методичні вказівки до виконання лабораторних робіт з дисципліни "Мікроконтролерні системи автоматики та управління" для студентів напряму підготовки 6.050102 – "Комп'ютерна інженерія"./ Укл. А.І. Роговенко, О.В. Красножон, А.В. Красножон – Чернігів: ЧДТУ, 2014. – 51 с. Рос. мовою.

Составители: Роговенко Андрей Иванович, старший преподаватель кафедры информационных и компьютерных систем
Красножон Алексей Васильевич, ассистент кафедры информационных и компьютерных систем
Красножон Андрей Васильевич, кандидат технических наук, старший преподаватель кафедры электрических систем и сетей

Ответственный за выпуск: Казимир Владимир Викторович, заведующий кафедрой информационных и компьютерных систем, доктор технических наук, профессор

Рецензент: Вервейко Александр Иванович, кандидат технических наук, доцент, доцент кафедры информационных и компьютерных систем Черниговского государственного технологического университета

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В последнее время однокристальные микроконтроллеры находят самое разнообразное применение во многих областях человеческой деятельности, причём границы их применения расширяются с каждым днём. С появлением первых микроконтроллеров их сложность постоянно возрастает за счёт разработки более совершенных архитектур, новых решений в области полупроводниковых технологий, добавления новых инструкций. Для решения задач с высоким уровнем сложности применяются микроконтроллерные системы.

Значительных успехов в области создания 16/32-разрядных микроконтроллерных ядер достигла британская компания ARM (Advanced RISC Machines). Благодаря высокой производительности и выгодному соотношению цена/качество эти микроконтроллеры приобрели широкую популярность в различных областях человеческой деятельности. Фактически, микроконтроллеры с ядром ARM выделились в отдельный класс микроконтроллеров для встраиваемых приложений.

Компания ARM не имеет собственных мощностей полупроводникового производства, поэтому микроконтроллерные ядра, поставляемые ею в различных видах (описанные на языках VHDL или Verilog, в электрических схемах или топологической макроячейке), реализуются в дальнейшем такими крупными производителями, как Atmel, Philips, Samsung, Sony, Texas Instruments и т.д.

Согласно рекомендациям производителя, микроконтроллеры с ядром ARM предназначены для использования в автомобилестроении, медицине, промышленной и бытовой электронике (в том числе и с батарейным питанием), компьютерных сетях и системах. Это возможно благодаря высоким показателям надёжности, относительно низкой мощности потребления и наличию различных режимов функционирования.

Микроконтроллеры с ядром ARM различных версий поддерживаются многими программными продуктами – отладчики, компиляторы C/C++, эмуляторы, операционные системы реального времени, драйверы низкого уровня, высокоуровневые приложения – различных всемирно известных производителей: Keil Software, Microsoft, Philips и другие.

Производитель определяет ядро ARM версии 7 как универсальное ядро 32-разрядного RISC-микроконтроллера с малым энергопотреблением, предназначенное для использования в различных заказных и специализированных интегральных схемах.

Данные методические указания предназначены для ознакомления и начального изучения основ программирования микроконтроллеров с ядром ARM7 и систем на их основе с помощью языка C, а также для выполнения цикла лабораторных работ по дисциплине “Микроконтроллерные системы автоматики и управления”. Они не охватывают всех доступных возможностей и нюансов обращения с микроконтроллерами данного семейства, поэтому при необходимости следует обращаться и к другим литературным источникам и ресурсам.

1 ВЗАИМОДЕЙСТВИЕ МИКРОКОНТРОЛЛЕРОВ С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ

Ввиду наличия большого количества периферийных устройств, в данном разделе методических указаний будут рассматриваться особенности внутренней организации, архитектурного и структурного построения лишь тех периферийных устройств, программирование работы которых предусмотрено практическим курсом:

– жидкокристаллический индикаторный модуль;
– приёмопередатчик SPI-интерфейса;
– аналого-цифровой преобразователь;
– цифро-аналоговый преобразователь.

Информацию, относящуюся к таким периферийным устройствам микроконтроллеров с ядром ARM7, как контроллер векторов прерываний, модуль управления линиями ввода/вывода следует искать в методических указаниях к самостоятельной и расчётно-графической работам.

1.1 Жидкокристаллический индикаторный модуль

В настоящее время для отображения информации широко используются алфавитно-цифровые жидкокристаллические индикаторы (ЖКИ), которые являются одними из основных средств вывода информации в современных микроконтроллерных системах. Они представляют собой дешевое и удобное решение, позволяющее экономить время и ресурсы при разработке новых изделий, обеспечивают отображение большого объема информации при хорошей различимости и низком энергопотреблении, благодаря чему широко используются в измерительных приборах, медицинском оборудовании, промышленном оборудовании, информационных системах, аппаратуре с автономным питанием.

Такой индикатор представляет собой одну или несколько строк, каждая из которых состоит из нескольких знакомест. Каждое знакоместо, в свою очередь, представляется массивом точек (как правило, 5x7). На сегодняшний день существует несколько различных стандартных форматов ЖКИ-модулей (символов x строк): 8x2, 16x1, 16x2, 16x4, 20x1, 20x2, 20x4, 24x2, 40x2, 40x4. Встречаются и менее распространенные форматы: 8x1, 12x2, 32x2 и др. Принципиальных ограничений на комбинации и количество отображаемых символов контроллер ЖКИ-модуля не накладывает – модуль может иметь любое количество символов от 1 до 80, но в некоторых комбинациях программная адресация символов может оказаться крайне неудобной. В учебно-отладочном стенде LPC2148 Education Board используется ЖКИ, который содержит 2 строки по 16 символов (знакомест).

Для управления работой алфавитно-цифровым ЖКИ используется специальный модуль, который имеет следующие внешние выводы:

– Vcc – питание модуля;
– Gnd – общий контакт (земля);
– Vlcd – управление контрастом;

– RS – выбор регистра команд или данных;
– E – строб записи/чтения;
– R/W – сигнал записи/чтения;
– DB7 .. DB 0 – двунаправленные линии обмена командами/данными;
– LED_A – анод светодиодной подсветки;
– LED_C – катод светодиодной подсветки.

Используемый в лабораторных работах ЖКИ-модуль имеет светодиодную подсветку. Вывод R/W заземлён, поэтому модуль доступен только для операций записи команд/данных.

Временная диаграмма записи команд/данных в ЖКИ-модуль показана на рисунке 1.1.

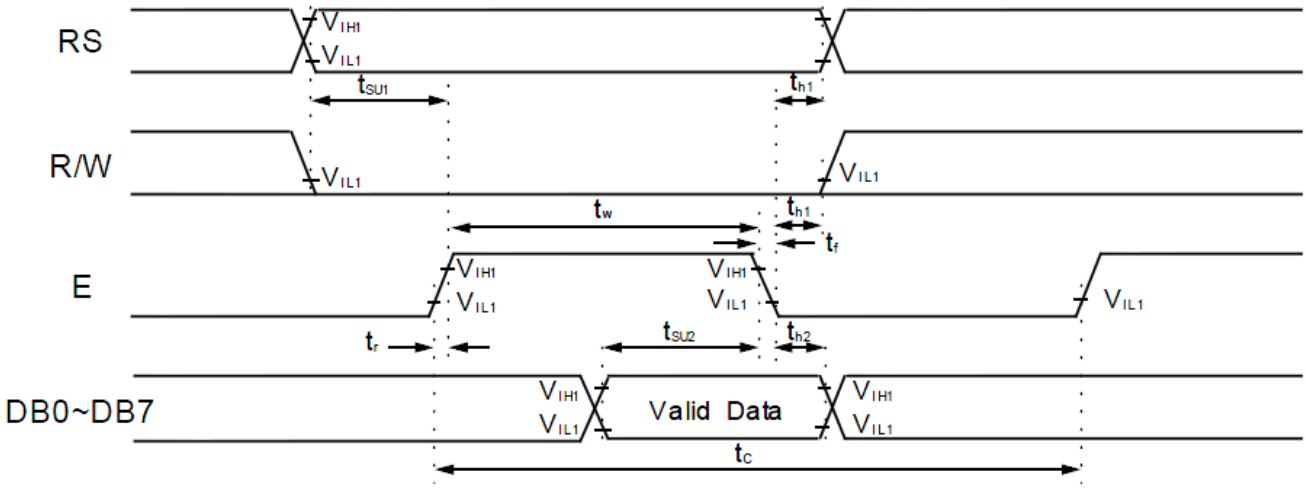

Рисунок 1.1 – Временная диаграмма записи информации в ЖКИ-модуль

Все указанные на диаграмме уровни и длительности сигналов следует в точности выдерживать. В лабораторном стенде процессом обмена данными с ЖКИ-модулем управляет микроконтроллер, он же и формирует приведённую временную диаграмму обмена.

Контроллер HD44780 фирмы Hitachi фактически является промышленным стандартом и широко применяется при производстве алфавитно-цифровых ЖКИ-модулей. Аналоги этого контроллера или совместимые с ним по интерфейсу и командному языку микросхемы, выпускают множество фирм, среди которых: Epson, Toshiba, Sanyo, Samsung, Philips. Одним из таких аналогов является контроллер KS0066U.

Контроллер HD44780 потенциально может управлять 2-мя строками по 40 символов в каждой (для модулей с 4-мя строками по 40 символов используются два однотипных контроллера), при использовании матрицы символа 5x7 точек. Контроллер также поддерживает символы с матрицей 5x10 точек, но в последние годы ЖКИ-модули с такой матрицей практически не встречаются.

Упрощённая внутренняя структура ЖКИ-модуля показана на рисунке 1.2.

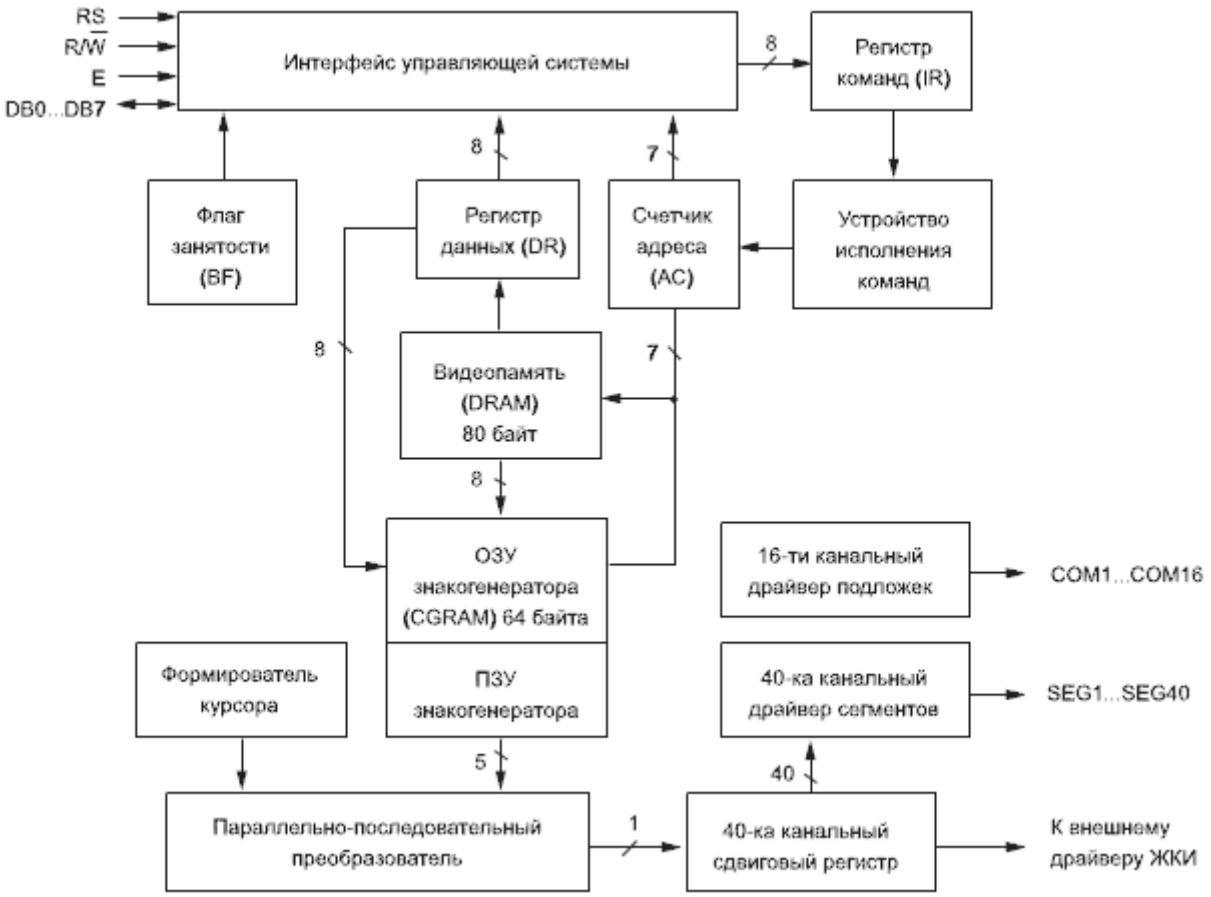

Рисунок 1.2 – Упрощённая внутренняя структура ЖКИ-модуля

Используемый в лабораторном стенде ЖКИ-модуль, имеет 80 ячеек видеопамяти DDRAM (для хранения кодов символов сообщения). Для первой строки модуля зарезервировано 40 ячеек памяти DDRAM с адресами от 00h до 27h. Для второй строки модуля также зарезервировано 40 ячеек с адресами от 40h до 67h (таким образом, в адресах ячеек памяти между первой и второй строкой имеется “дырка”). При последовательной записи информации в DDRAM внутренний управляющий контроллер ЖКИ-модуля автоматических переходит с последней ячейки первой строки (27h) на первую ячейку второй строки (40h), если включен соответствующий режим автоматического увеличения адресов. Следует помнить, что на экране ЖКИ-модуля отображается содержимое лишь 16-ти ячеек для 1-й строки и 16-ти ячеек для второй строки. При выполнении команды сдвига экрана содержимое ячеек DDRAM не меняется, меняются лишь указатели начала отображаемых элементов строк. Например, после инициализации модуля, мы видим в первой его строке содержимое ячеек DDRAM с адресами от 00h до 0Fh, а во второй – с адресами от 40h до 4Fh (смотри рисунок 1.3).

For 16×2 or 8×2 Display

Character	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DD RAM Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Рисунок 1.3 – Взаимное соответствие знакомест и ячеек памяти DDRAM для ЖКИ-модуля после сброса

После выполнения команды сдвига экрана влево в первой строке будет отображаться содержимое ячеек с адресами от 01h до 10h, а во второй – от 41h до 50h. Аналогичные действия выполняются и при сдвиге экрана вправо.

Чтение регистра DR приводит к загрузке содержимого DDRAM или CGRAM, в зависимости от текущего режима, при этом курсор смещается на одну позицию как и при записи. Чтение регистра IR возвращает 8 значащих разрядов, причем в 7-ми младших содержится текущее значение счетчика AC (7 разрядов, если адресуется DDRAM, и 6 – если CGRAM), а в старшем – флаг занятости BF. Этот флаг имеет значение 1, когда контроллер занят и 0 – когда свободен. Необходимо учитывать, что большинство операций, выполняемых контроллером, занимают значительное время, около 40 мкс, а время выполнения некоторых доходит до единиц миллисекунд, поэтому цикл ожидания снятия флага BF должен обязательно присутствовать в программах драйвера ЖКИ-модуля и предшествовать совершению любой операции (естественно, кроме операции проверки флага BF).

ЖКИ-модуль также содержит ПЗУ знакогенератора (CGROM) и ОЗУ знакогенератора (CGRAM). ПЗУ содержит предустановленные производителем символы для отображения модулем. Для вывода такого предустановленного символа достаточно записать в соответствующую ячейку видеопамяти DDRAM его код (или адрес в памяти CGROM). Кодовая таблица ЖКИ-модуля, содержащего кириллические символы, показана на рисунке 1.4.

ОЗУ знакогенератора (CGRAM) может использоваться для создания символов пользователя.

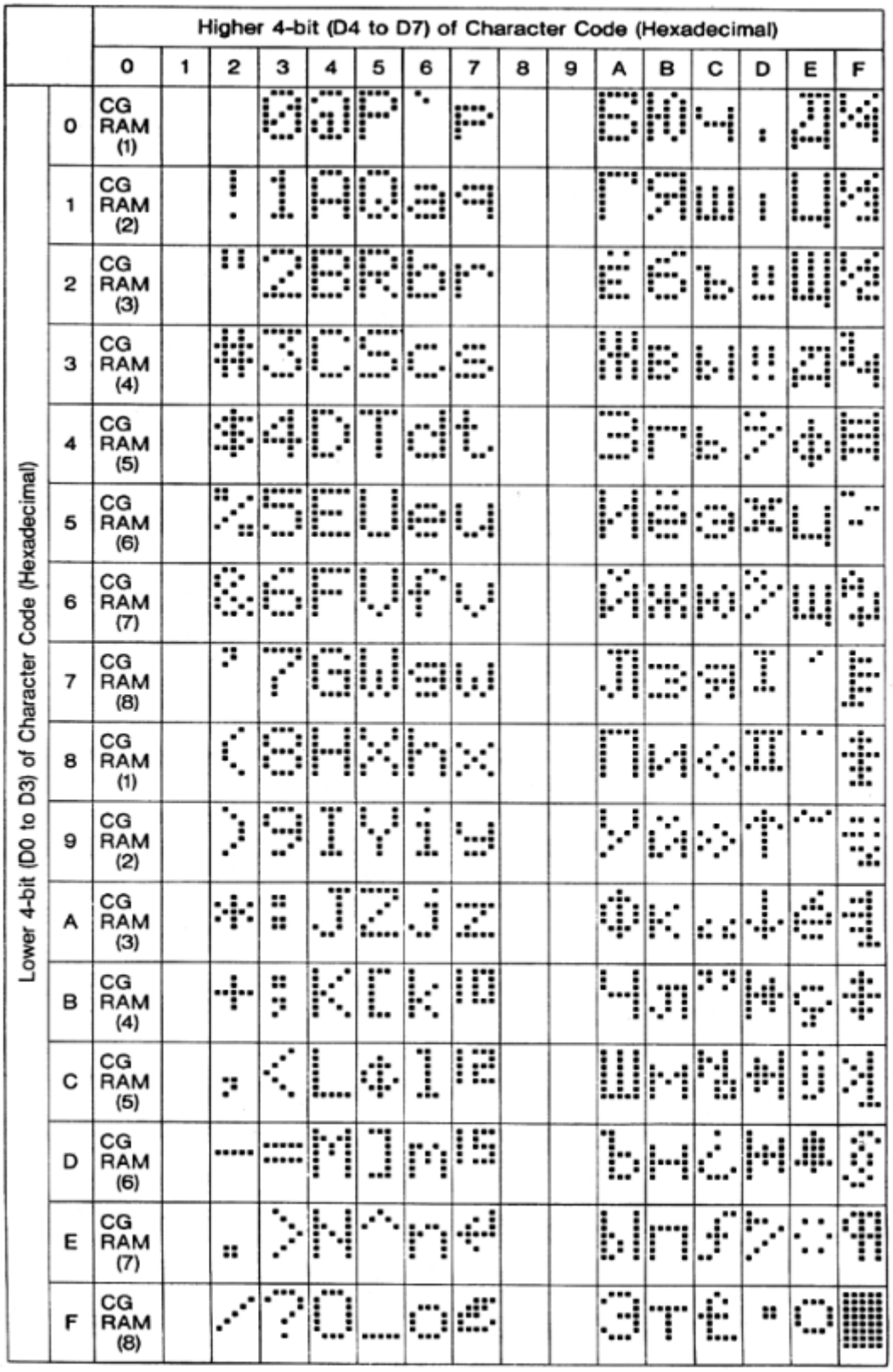

Рисунок 1.4 – Кодовая таблица ЖКИ-модуля

У контроллера HD44780 существует набор внутренних флагов, определяющих режимы работы различных его узлов. Переопределение значений флагов производится специальными командами, записываемыми в регистр IR (смотри рисунок 1.2), при этом комбинации старших битов определяют группу

флагов или команду, а младшие содержат собственно флаги. Назначение и формат всех команд управления показано ниже на рисунке 1.5.

COMMAND	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DESCRIPTION	Executing time fosc=250khz
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears Display & Returns to Address 0.	1.64ms
Cursor at Home	0	0	0	0	0	0	0	0	1	x	Returns Cursor to Address 0. Also returns the display being shifted to the original position. DDRAM contents remain unchanged.	1.64ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	I/D: Set Cursor Moving Direction I/D=1: Increment I/D=0: Decrement S: Specify Shift of Display S=1: The display is shifted S=0: The display is not shifted	40µs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Display D=1: Display on D=0: Display off Cursor C=1: Cursor on C=0: Cursor off Brink B=1: Brink on B=0: Brink off	40µs
Cursor / Display Shift	0	0	0	0	0	1	S/C	R/L	x	x	Moves cursor or shifts the display w/o changing DD RAM contents S/C=0: Cursor Shift (RAM unchanged) S/C=1: Display Shift (RAM unchanged) R/L=1: Shift to the Right R/L=0: Shift to the Left	40µs
Function Set	0	0	0	0	1	DL	N	F	x	x	Sets data bus length (DL), # of display lines (N), and character fonts (F). DL=1: 8 bits F=0: 5x7 dots DL=0: 4 bits F=1: 5x10 dots N=0: 1 line display N=1: 2 lines display	40µs
Set CG RAM Address	0	0	0	1	Character Generator (CG) RAM Address						Sets CG RAM address. CG RAM data is sent and received after this instruction.	40µs
Set DD RAM Address	0	0	1	Display Data (DD) RAM Address / Cursor Address							Sets DD RAM address. DD Ram data is sent and received after this instruction.	40µs
Busy Flag / Address Read	0	1	BF	Address counter used for both DD & CG RAM address							Reads Busy Flag (BF) and address counter contents.	40µs
Write Data	1	0	Write Data								Writes data into DDRAM or CGRAM.	46µs
Read Data	1	1	Read Data								Reads data from DDRAM or CGRAM.	46µs

x: Don't Care

Рисунок 1.5 – Назначение и формат команд ЖКИ-модуля

Следует обратить внимание, что после подачи очередной команды или байта данных необходимо выдержать паузу длительностью не менее чем указа-

но в поле Executing time на рисунке 1.5. В течение этого промежутка времени контроллер ЖКИ-модуля обрабатывает предыдущий принятый байт и может некорректно отреагировать на вновь поступивший байт. Значение времени выполнения каждой из команд на рисунке 1.5 указано для случая, когда внутренняя тактовая частота работы контроллера ЖКИ-модуля составляет 250 кГц. На практике встречаются ЖКИ-модули, работающие на тактовых частотах 250 кГц или 270 кГц. При частоте работы в 270 кГц задержки на выполнение команд/обработку данных будут немного меньше. На практике, с целью обеспечения совместимости с любыми модулями, рекомендуется использовать задержки, которые несколько превышают указанные на рисунке 1.5 значения.

После включения питания программисту необходимо произвести инициализацию ЖКИ-модуля. В лабораторном стенде может использоваться как 8-битный режим обмена между микроконтроллером и ЖКИ-модулем, так и 4-битный. Инициализация ЖКИ-модуля в первом режиме осуществляется согласно алгоритму, показанному на рисунке 1.6. Временные задержки на выполнение алгоритма инициализации указаны для ЖКИ-модуля с внутренней тактовой частотой 270 кГц. При использовании более “медленного” ЖКИ-модуля, например, с частотой 250 кГц, все указанные задержки следует умножать на коэффициент 270/250.

В случае если используется не 8-битный, а 4-х битный режим обмена, пересылка или приём команд или данных осуществляется тетрадами (по 4 бита). Этот режим выбирается при посылке команды Function Set (смотри рисунок 1.5) с определённым значением бита DL (должен быть равен 0). В этом случае используются только выводы DB7 ... DB4 ЖКИ-модуля. Поскольку любой элемент информации (команда или код символа) имеет разрядность не более байта, поэтому обмен осуществляется в 2 этапа: на первом этапе происходит передача старшей тетрады байта (с учётом порядка следования битов), на втором – передаётся младшая из тетрад. Несложно заметить, что при использовании 4-х битного режима все задержки на обмен данными или командами с ЖКИ-модулем увеличиваются более, чем в 2 раза. Поэтому такой режим применяется крайне редко, хотя он и позволяет сэкономить количество используемых выводов микроконтроллера (которые участвуют в обмене) с 11 до 7, что было критично в те времена, когда микроконтроллеры обладали малым количеством выводов (например, выполнялись в 20-выводном корпусе).

1) 8-bit interface mode (Condition: fosc = 270KHZ)

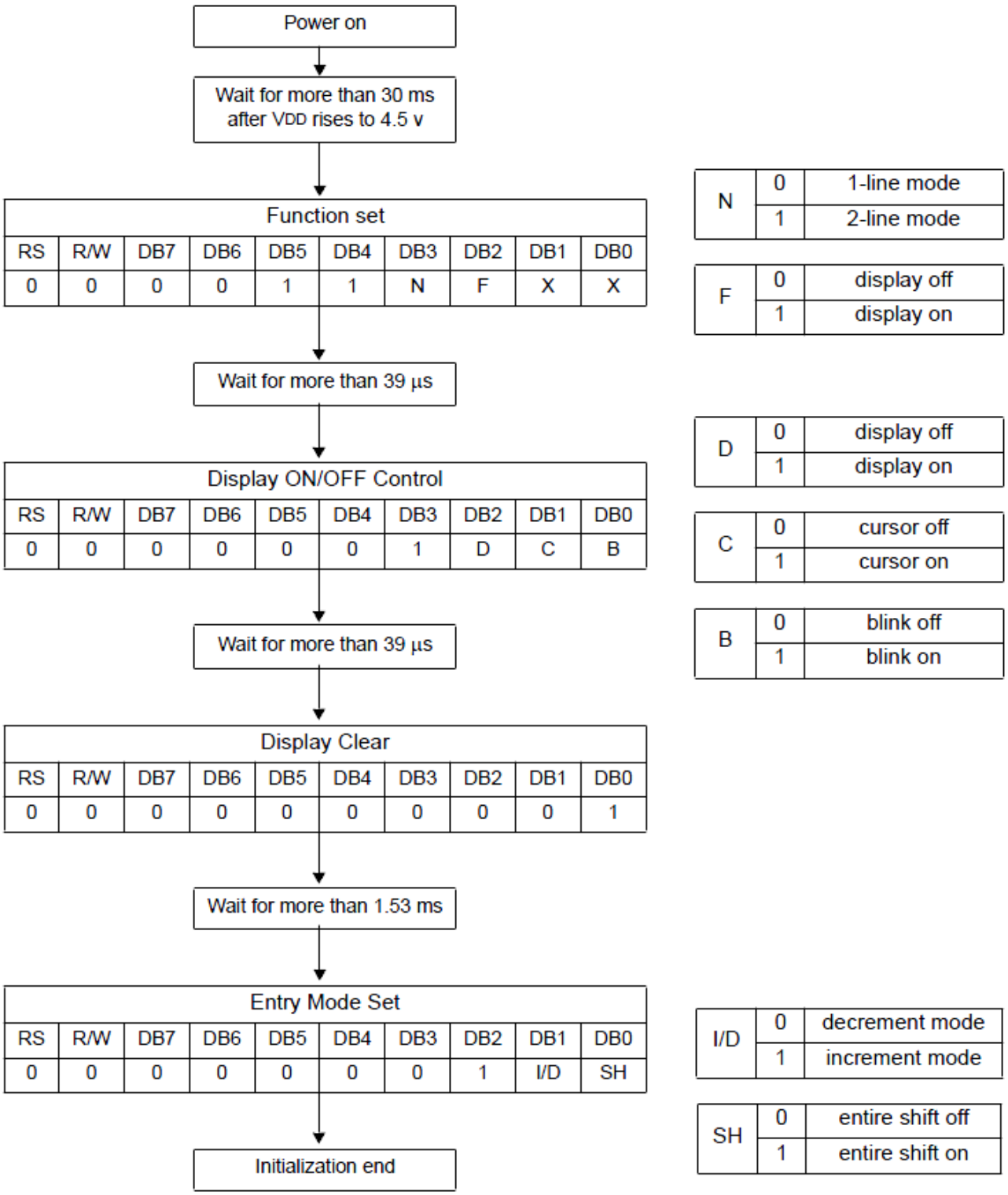

Рисунок 1.6 – Алгоритм инициализации ЖКИ-модуля в 8-битном режиме обмена

Пример широко распространенной последовательности для инициализации ЖКИ-модуля: $38, $OC, $06 (знак "$" указывает на шестнадцатеричное основание системы счисления). $38 устанавливает режим отображения 2-х строк с

матрицей 5x7 точек и работу с 8-ми разрядной шиной данных. $OC включает отображение символов на индикаторе, без отображения курсора. $06 устанавливает режим автоматического перемещения курсора слева-направо после вывода каждого символа.

1.2 Приёмопередатчик интерфейса SPI

Реализация интерфейса SPI для микроконтроллеров семейства LPC2000 имеет следующие особенности:

– 2 полнофункциональных и независимых модуля интерфейса SPI;

– полная поддержка стандартной спецификации Serial Peripheral Interface;

– синхронное последовательное полнодуплексное соединение;

– поддержка режимов работы: ведущий и ведомый;

– максимальная частота обмена составляет 1/8 от тактовой частоты.

Структурная схема модуля интерфейса SPI для микроконтроллеров семейства LPC2000 приведена ниже на рисунке 1.7. По такой схеме построены оба интерфейсных модуля – SPI0 и SPI1.

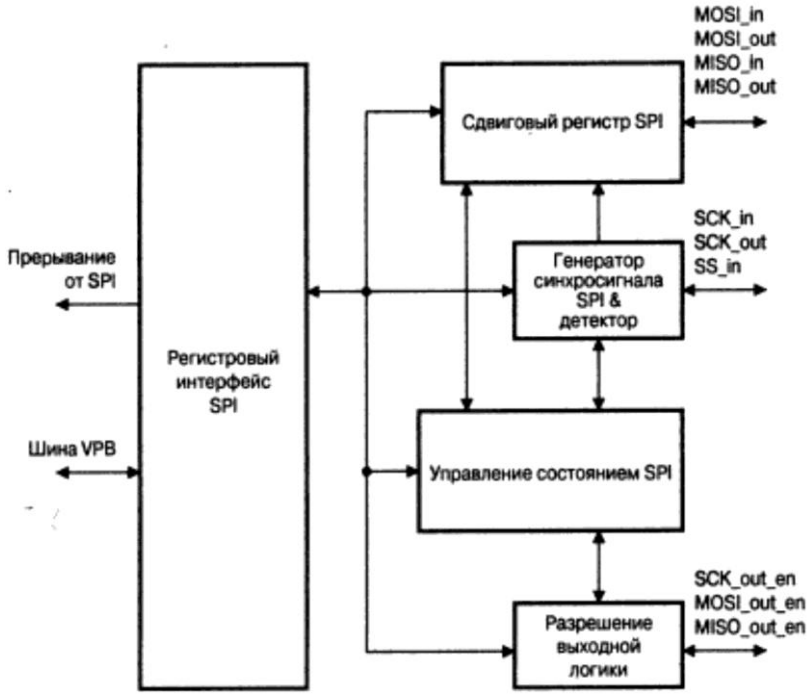

Рисунок 1.7 – Структурная схема модуля интерфейса SPI

1.2.1 Исключительные ситуации в работе интерфейса SPI

В процессе обмена данными по интерфейсу SPI возможно возникновение исключительной ситуации, т.е. ситуации, когда происходит некоторая некорректная операция, приводящая к потере данных или другим нежелательным последствиям. Существуют следующие виды исключительных ситуаций:

– переполнение при чтении;

– конфликт при записи;

– ошибка выбора режима;

– аварийное прекращение работы ведомого.

Переполнение при чтении может произойти тогда, когда внутренний приемный буфер модуля интерфейса SPI содержал данные, которые не были прочитаны микроконтроллером до завершения следующей операции передачи данных. Признаком того, что приемный буфер содержит достоверные данные, служит текущее активное состояние бита SPIF в регистре состояния. Когда передача данных завершается, модуль интерфейса SPI автоматически перемещает принятые данные в приемный буфер. Если в этот момент бит SPIF находится в активном состоянии, то при перемещении новых данных в приемный буфер находящиеся там старые данные будут “затерты” (т.е. потеряны). На переполнение при чтении указывает бит ROVR в регистре состояния, который при этом перейдет в активный уровень.

Как видно из рисунка 1.7, между регистровым интерфейсом модуля SPI и внутренним сдвиговым регистром отсутствует какой-либо аппаратный буфер. В связи с этим, данные не должны записываться в соответствующий регистр в то время, когда через интерфейс SPI производится передача данных. Интервал времени, в течение которого новые данные нельзя записывать в регистр данных, отсчитывается от момента начала передачи, т.е. от момента последней записи в регистр данных до момента чтения регистра состояния, в котором бит SPIF находился в активном уровне. Если запись в регистр данных будет произведена в указанный интервал, записываемые данные будут потеряны, а в регистре состояния установится бит WCOL, который будет свидетельствовать о наличии конфликта записи.

Внешний сигнал на входе SSEL устройства, модуль интерфейса SPI которого сконфигурирован как ведущий, всегда должен иметь неактивный (высокий) уровень. Если сигнал на входе SSEL ведущего устройства станет активным, то это означает, что другой ведущий выбрал данное устройство в качестве ведомого. Такое состояние для выбранного устройства называется ошибкой выбора режима. В случае его возникновения, взводится бит ошибки выбора режима MODF в регистре состояния. При этом драйверы сигналов модуля интерфейса SPI деактивируются, а режим работы данного устройства изменяется с ведущего на ведомый.

Передача данных по интерфейсу SPI ведомым устройством считается прерванной, если внешний сигнал SSEL переходит в неактивный (высокий) уровень прежде, чем передача будет закончена. В случае аварийного прекращения работы ведомого передаваемые и принимаемые данные текущего обмена будут потеряны, а бит аварийного прекращения работы ведомого ABRT в регистре состояния ведомого устройства перейдет в активное состояние.

1.2.2 Описание выводов модуля интерфейса SPI

Преобладающее большинство микроконтроллеров семейства LPC2000 содержат в своём составе 2 абсолютно одинаковых модуля интерфейса SPI (обычно их нумеруют цифрами 0 и 1). Описание выводов модулей интерфейса SPI приведено ниже в таблице 1.1.

Таблица 1.1 – Описание выводов модулей интерфейса SPI

Название	Направление	Описание
1	2	3
SCK0, SCK1	Вход/выход	Сигнал синхронизации передачи данных по интерфейсу SPI. Всегда генерируется ведущим устройством и принимается ведомым. Может быть сконфигурирован с активным высоким или низким уровнем сигнала. Сигнал на этом выходе изменяется только во время передачи данных, в любое другое время выход может быть в неактивном или высокоимпендансном состоянии (Z-состоянии)
SSEL0, SSEL1	Вход/выход	Сигнал выбора ведомого устройства, имеет низкий активный уровень, указывающий, что ведомое устройство в текущий момент выбрано, чтобы участвовать в обмене данными. Каждое ведомое устройство имеет отдельный вход SSEL, т.е. ведущий может выбирать одного из нескольких ведомых, подключенных к одной и той же шине SPI. Сигнал на входе ведомого должен перейти в низкий уровень до того, как начнется обмен данными, и оставаться таковым (в нормальном режиме) в течение всего времени обмена. Если сигнал переходит в высокий уровень в любой момент при передаче данных, то передача считается прерванной. В этом случае, ведомое устройство возвращается в режим ожидания, а все данные теряются. Выход SSEL ведущего не имеет аппаратного управления от модуля SPI, а управляется программно, поэтому ведущий микроконтроллер должен формировать сигнал SSEL посредством использования линий ввода/вывода общего назначения. Для микроконтроллеров семейства LPC2000 на вход SSEL модуля SPI ведущего устройства необходимо подать внешний высокий уровень. Иначе, возникнет исключительная ситуация
MISO0, MISO1	Вход/выход	Вход ведущего, выход ведомого. Линия сигнала MISO – однонаправленная линия, по которой последовательные данные передаются от ведомого к ведущему. Если устройство является ведомым, последовательные данные выводятся через эту линию, а если ведущим – поступают через неё. Если ведомое устройство не выбрано, этот его выход находится в Z-состоянии

Продолжение таблицы 1.1

1	2	3
MOSI0, MOSI1	Вход/выход	Выход ведущего, вход ведомого. Линия сигнала MOSI – однонаправленная линия, по которой последовательные данные передаются от ведущего к ведомому. Если устройство является ведущим, последовательные данные выводятся через эту линию, а если ведомым – поступают через неё.

1.2.3 Регистры модуля интерфейса SPI

Любой из двух модулей интерфейса SPI микроконтроллеров семейства LPC2000 включает в себя пять регистров, перечисленных в таблице 1.2. Все они доступны как байт, полуслово или полное слово.

Таблица 1.2 – Описание регистров модулей интерфейса SPI

Адрес	Название	Описание	Доступ
0xE0020000	S0SPCR	Регистр управления операциями модуля интерфейса SPI	R/W
0xE0030000	S1SPCR		
0xE0020004	S0SPSR	Регистр, отображающий состояние модуля интерфейса SPI	RO
0xE0030004	S1SPSR		
0xE0020008	S0SPDR	Регистр, обеспечивающий обмен данными по интерфейсу SPI. Передача данных начинается после записи в регистр. Приём данных осуществляется путём чтения этого регистра	R/W
0xE0030008	S1SPDR		
0xE002000C	S0SPCCR	Регистр счётчика синхроимпульсов, генерируемых в сеансе обмена	R/W
0xE003000C	S1SPCCR		
0xE002001C	S0SPINT	Регистр, содержащий флаг прерывания по интерфейсу SPI	R/W
0xE003001C	S1SPINT		

Регистр управления SPCR (SPI Control Register) осуществляет управление операциями модуля интерфейса SPI в соответствии с заданными значениями битов конфигурации. Битовая структура данного регистра представлена ниже в таблице 1.3. Значения битов регистра должны быть установлены до начала сеанса обмена.

Таблица 1.3 – Битовая структура регистра SPCR

Номер бита	Название бита	Функция бита	Значение после сброса
2:0	Зарезер-вированы	Пользовательское ПО не должно производить запись в зарезервированные биты. Чтение зарезервированных битов возвращает неопределенное значение	Неопре-деленное
3	CPHA	Бит управления фазой синхросигнала. Определяет временное соотношение между синхросигналом и данными при передаче, а также моменты начала и окончания передачи для ведомого устройства. Если бит установлен, выборка данных производится по срезу синхроимпульса SCK. Если бит сброшен, выборка данных производится по фронту синхроимпульса SCK. Передача начинается и заканчивается, соответственно, в моменты активации и отпускания сигнала SSEL для данного ведомого устройства	0
4	CPOL	Бит управления полярностью синхросигнала. Если бит установлен, сигнал SCK имеет низкий активный уровень. Если бит сброшен, сигнал SCK имеет высокий активный уровень	0
5	MSTR	Бит выбора режима ведущего. Если бит установлен, модуль интерфейса SPI работает в режиме ведущего. Если бит сброшен, модуль интерфейса SPI работает в режиме ведомого	0
6	LSBF	Бит управления порядка передачи битов. Если бит установлен, то первым передается LSB (бит 0). Если бит сброшен, то первым передается MSB (бит 7)	0
7	SPIE	Разрешение генерации прерывания от SPI. Если бит установлен, каждый переход в активный уровень бита SPIF или бита MODF регистра SPSR генерирует аппаратное прерывание. Если бит сброшен, то генерация прерываний запрещена	0

Регистр состояния SPSR (SPI Status Register) отображает текущее состояние модуля интерфейса SPI. Главное предназначение этого регистра состоит в том, чтобы обнаруживать моменты завершения передачи данных. Он доступен только для чтения. Битовая структура данного регистра приведена ниже в таблице 1.4.

Таблица 1.4 – Битовая структура регистра SPSR

Номер бита	Название бита	Функция бита	Значение после сброса
2:0	Зарезервированы	Пользовательское ПО не должно производить запись в зарезервированные биты. Чтение зарезервированных битов возвращает неопределенное значение	Неопределенное
3	ABRT	Бит аварийного прекращения работы ведомого. Установка бита сигнализирует об аварийном прекращении работы ведомого. Бит сбрасывается чтением регистра SPSR	0
4	MODF	Бит ошибки выбора режима. Установка бита сигнализирует об ошибке выбора режима. Бит сбрасывается чтением регистра SPSR и последующей записи в регистр SPCR	0
5	ROVR	Бит переполнения при чтении. Установка бита сигнализирует о переполнении при чтении. Бит сбрасывается чтением регистра SPSR	0
6	WCOL	Бит конфликта записи. Установка бита сигнализирует о конфликте записи. Бит сбрасывается чтением регистра SPSR и последующей записи в регистр SPDR	0
7	SPIF	Флаг окончания передачи данных. Установка флага сигнализирует об окончании процесса передачи данных. Если устройство – ведущий, флаг устанавливается в конце последнего цикла передачи. Если устройство – ведомый, флаг устанавливается по фронту или срезу (в зависимости от настроек модуля) сигнала SCK, которым осуществляется выборка последнего бита данных. Флаг сбрасывается при первом чтении регистра SPSR и последующем обращении к регистру SPDR. Данный флаг не является флагом прерывания, т.к. последний находится в регистре SPINT.	0

Регистр данных SPDR (SPI Data Register) является двунаправленным регистром, т.е. обеспечивает передачу и прием данных по интерфейсу SPI. Данные, подлежащие передаче, записываются в регистр, а принятые – могут быть прочитаны из него. Если устройство является ведущим, запись в регистр инициирует передачу данных. Однако запись в этот регистр не приведет к началу передачи в том случае, если бит SPIF в регистре SPSR был установлен и регистр состояния не был прочитан. Битовая структура данного регистра приведена ниже в таблице 1.5.

Таблица 1.5 – Битовая структура регистра SPDR

Номер бита	Название бита	Функция бита	Значение после сброса
7:0	Данные	Двунаправленный буферизированный регистр обмена данными по интерфейсу SPI	0

Регистр счётчика синхроимпульсов SPCCR (SPI Clock Counter Register) управляет значением частоты синхросигнала ведущего устройства. Регистр хранит количество циклов сигнала pclk, которое составляет один цикл синхросигнала SCK. Значение в этом регистра всегда должно быть четным, т.е. бит 0 всегда сброшен, также оно должно быть больше или равно 8. Несоблюдение этих условий может привести к непредсказуемости поведения модуля интерфейса SPI. Битовая структура данного регистра представлена в таблице 1.6.

Абсолютное значение частоты сигнала SCK может быть рассчитано как: pclk/значение регистра SPCCR, где значение pclk равно: cclk/значение регистра делителя VPBDIV (шины VPB).

Таблица 1.6 – Битовая структура регистра SPCCR

Номер бита	Название бита	Функция бита	Значение после сброса
7:0	Счётчик	Значение счётчика синхроимпульсов интерфейса SPI	0

Регистр прерывания SPINT (SPI Interrupt) предназначен для хранения флага прерывания интерфейса SPI. Битовая структура данного регистра представлена ниже в таблице 1.7.

Таблица 1.7 – Битовая структура регистра SPINT

Номер бита	Название бита	Функция бита	Значение после сброса
0	Флаг прерывания интерфейса SPI	Флаг прерывания интерфейса SPI. Флаг устанавливается аппаратно в случае возникновения прерывания. Сброс флага осуществляется программно путём записи в него логической 1 (после чего он автоматически сбросится в 0). Флаг будет установлен один раз, когда происходит установка бита SPIE и хотя бы одного из битов SPIF или WCOL. Но лишь в том случае, когда бит прерывания установлен, и соответствующее прерывание разрешено в VIC. Такое прерывание может быть обработано программным способом.	0
7:1	Зарезервированы	Пользовательское ПО не должно производить запись в зарезервированные биты. Чтение зарезервированных битов возвращает неопределенное значение	Неопределенное

1.3 Модуль аналого-цифрового преобразования

Модуль аналого-цифрового преобразования (АЦП) микроконтроллеров семейства LPC2000 обладает следующими основными характеристиками:

– 10-разрядный АЦП последовательного приближения;

– количество независимых каналов АЦП составляет 4 или 8;

– поддержка энергосберегающего режима Power down;

– диапазон входного измеряемого напряжения 0 ... 3 В;

– время одного 10-битного преобразования не более 2.44 мкс;

– режим преобразования Burst для одного или нескольких входов;

– возможность осуществлять преобразование по изменению внешнего сигнала на цифровом входе или по сигналу совпадения таймера.

Синхронизация модуля аналого-цифрового преобразования обеспечивается синхросигналом VPB. Для преобразования частоты этого сигнала в частоту 4.5 МГц, которая необходима для осуществления аналого-цифрового преобразования, служит программируемый делитель. Длительность времени аналого-цифрового преобразования входного напряжения с полной точностью (при 10-разрядном выходном коде) занимает 11 периодов сигнала с частотой 4.5 МГц. (поскольку АЦП, работающий по принципу последовательного приближения, требует в процессе преобразования на 1 период тактового сигнала больше, чем разрядность выходного кода, в который он преобразует заданное входное

напряжение). В таком случае 10-разрядный выходной код появится за время, равное 11 * $1/4{,}5*10^6 = 2{,}444$ мкс.

В микроконтроллере LPC2148, помимо модуля ADC0, также имеется модуль ADC1 со сходной архитектурой. Для управления АЦП, интегрированным в микроконтроллер LPC2148, используется набор регистров, описанных ниже в таблице 1.8.

Таблица 1.8 – Описание регистров модуля АЦП

Адрес	Название	Описание	Доступ
0xE0034000	AD0CR	Регистр управления соответствующим модулем АЦП	R/W
0xE0060000	AD1CR		
0xE0034004	AD0GDR	Регистр, содержащий преобразованное значение для текущего канала, а также признак окончания преобразования	R/W
0xE0060004	AD1GDR		
0xE0034030	AD0STAT	Регистр, содержащий информацию о текущем состоянии	RO
0xE0060030	AD1STAT		
0xE003400C	AD0INTEN	Регистр управления генерацией прерываний модуля АЦП	R/W
0xE006000C	AD1INTEN		
0xE0034010	AD0DR0	Регистр, содержащий преобразованное значение для канала 0	R/W
0xE0060010	AD1DR0		
0xE0034014	AD0DR1	Регистр, содержащий преобразованное значение для канала 1	R/W
0xE0060014	AD1DR1		
0xE0034018	AD0DR2	Регистр, содержащий преобразованное значение для канала 2	R/W
0xE0060018	AD1DR2		
0xE003401C	AD0DR3	Регистр, содержащий преобразованное значение для канала 3	R/W
0xE006001C	AD1DR3		
0xE0034020	AD0DR4	Регистр, содержащий преобразованное значение для канала 4	R/W
0xE0060020	AD1DR4		
0xE0034024	AD0DR5	Регистр, содержащий преобразованное значение для канала 5	R/W
0xE0060024	AD1DR5		
0xE0034028	AD0DR6	Регистр, содержащий преобразованное значение для канала 6	R/W
0xE0060028	AD1DR6		
0xE003402C	AD0DR7	Регистр, содержащий преобразованное значение для канала 7	R/W
0xE006002C	AD1DR7		

Конфигурирование работы модуля АЦП осуществляется с помощью соответствующего регистра управления ADCR (A/D Control Register). Битовая структура данного регистра представлена ниже в таблице 1.9.

Таблица 1.9 – Битовая структура регистра ADCR

Номер бита	Название бита	Функция бита	Значение после сброса
1	2	3	4
7:0	CHN	Биты выбора рабочего канала АЦП. Если некоторый бит установлен в “1”, то соответствующий порядковому номеру бита канал и будет осуществлять преобразование (т.е., 0x01 выбирает канал 0; 0x02 выбирает канал 1; 0x80 – канал 7)	Неопределенное
15:8	CLKDIV	Биты коэффициента деления частоты для шины периферийных устройств. Если значения всех битов равны 0, то коэффициент деления равен 10, т.е. частота работы АЦП – 4,5 МГц (при частоте синхронизации шины VPB – 45 МГц)	0
16	REPEAT	Бит режима выполнения преобразования. Установка бита указывает на то, что преобразование будет осуществляться модулем автоматически. Сброс бита указывает на необходимость генерации запроса на преобразование вручную.	0
19:17	SIZE	Биты разрядности преобразования. Устанавливают разрядность результата преобразования в следующем виде: при 000 – 10 бит; при 001 – 9 бит; при 010 – 8 бит; при 011 – 7 бит; при 100 – 6 бит; при 101 – 5 бит; при 110 – 4 бита; при 111 – 3 бита;	0
20	Зарезервирован	Пользовательское ПО не должно производить запись в зарезервированный бит. Чтение зарезервированного бита возвращает неопределенное значение	Неопределенное
21	POWER	Бит включения питания модуля АЦП. При установке бита разрешается подача питания на соответствующие входы модуля АЦП, а сброс бита отключает питание модуля АЦП.	0
23:22	Зарезервированы	Пользовательское ПО не должно производить запись в зарезервированные биты. Чтение зарезервированных битов возвращает неопределенное значение	Неопределенное

Продолжение таблицы 1.9

1	2	3	4
26:24	EVENT	Биты условия начала преобразования. Если бит 16 регистра ADCR установлен в 0, эти биты определяют событие, по наступлению которого будет генерироваться запрос на выполнение преобразования модулем АЦП.	0
27	FRONT	Бит условия начала преобразования. При установке бита преобразование будет осуществляться только по переднему фронту соответствующего управляющего сигнала. При сбросе бита – по заднему фронту соответствующего управляющего сигнала.	0
31:28	Зарезервированы	Пользовательское ПО не должно производить запись в зарезервированные биты. Чтение зарезервированных битов возвращает неопределенное значение	Неопределенное

После окончания выполнения аналого-цифрового преобразования, считывание результата осуществляется при помощи регистра ADGDR (A/D Global Data Register). Битовая структура данного регистра представлена ниже в таблице 1.10.

Таблица 1.10 – Битовая структура регистра ADGDR

Номер бита	Название бита	Функция бита	Значение после сброса
1	2	3	4
5:0	Зарезервированы	Пользовательское ПО не должно производить запись в зарезервированные биты. Чтение зарезервированных битов возвращает неопределенное значение	Неопределенное
15:6	RESULT	Биты, содержащие результат выполнения преобразования над входной аналоговой величиной	0
26:24	CHN	Биты номера канала. Кодируют номер канала, для которого был получен результат преобразования	0

Продолжение таблицы 1.10

1	2	3	4
30	WARN	Бит сигнализации о потере данных. Установка бита свидетельствует о том, что результат хотя бы одного из преобразований не был своевременно считан, а был уничтожен после перезаписи данных. Сброшенный бит свидетельствует о нормальном ходе выполнения операций преобразования	0
31	DONE	Бит занятости модуля АЦП. Установка бита свидетельствует о том, что в данный момент модуль АЦП занят выполнением преобразования. Запись или чтение данных в/из него может привести к непредсказуемым последствиям. Сброса бита свидетельствует об окончании выполнения преобразования.	0

Результат выполнения преобразования над входным аналоговым сигналом для некоторого независимого канала модуля АЦП сохраняется как в соответствующем регистре ADDR, так и в регистре ADGDR.Однако, в силу некоторых особенностей организации регистров ADDR, осуществлять чтение результатов преобразования, а также проверку занятости модуля лучше путём чтения и анализа содержимого регистра ADGDR.

В учебно-отладочном стенде LPC2148 Education Board регулирование величины напряжения, подаваемого на входы модуля АЦП, обеспечивается с помощью потенциометра – резистора с регулируемым сопротивлением – и находиться в диапазоне от 0 до 3,3 В). Для обеспечения электрического контакта между выходом потенциометра и входом модуля необходимо установить перемычку J28 в соответствующее положение.

1.4 Модуль цифро-аналогового преобразования

Цифро-аналоговые преобразователи (ЦАП) вырабатывают напряжение или ток, функционально связанный с управляющим кодом. Применяются ЦАП для формирования выходных аналоговых сигналов в цифровых измерительных и вычислительных устройствах. Для преобразования двоичного кода в аналоговый сигнал обычно формируются токи, пропорциональные весам разрядов кода, и затем суммируются те из токов, которые соответствуют ненулевым разрядам входного кода.

Применяются в основном два метода выполнения цифро-аналогового преобразования: суммирование единичных эталонных величин и суммирование эталонных величин, веса которых различаются. В первом при формировании выходной аналоговой величины используется только одна эталонная величина

весом в один квант. Во втором методе применяются эталонные величины с весами, зависящими от номера разряда, и в суммировании участвуют только те эталонные величины, для которых в соответствующем разряде входного кода установлена единица.

В случае использования на входе двоичного позиционного кода, значения всех разрядов поступают одновременно, и работа таких ЦАП описывается выражением:

$$X = \frac{P}{2^b - 1}(a_0 \cdot 2^0 + a_1 \cdot 2^1 + a_2 \cdot 2^2 + ... + a_{b-1} \cdot 2^{b-1}), \quad (1.1)$$

где X – аналоговая величина,

a_i – коэффициенты соответствующих двоичных разрядов, которые принимают дискретные значения единица или ноль,

P – опорный сигнал,

b – число разрядов.

В преобразователях из опорного напряжения формируются эталонные величины, соответствующие значениям разрядов входного кода, которые суммируются и образуют дискретные значения выходной аналоговой величины.

Также существующие ЦАП могут быть классифицированы по таким признакам:

– способ формирования выходной величины (с суммированием напряжений, с делением напряжений, с суммированием токов);

– природа выходного сигнала (с токовым выходом, с выходом по напряжению).

– полярности выходного напряжения (постоянное, переменное).

При интегральном исполнении микросхем ЦАП в качестве преобразовательных элементов могут использоваться следующие структуры:

– ЦАП со взвешенными резисторами в цепях эмиттеров;

– ЦАП со взвешенными резисторами в цепях нагрузки;

– ЦАП с лестничной матрицей R-2R в цепях эмиттеров транзисторов источников токов;

– ЦАП с выходной лестничной матрицей R-2R.

К основным параметрам ЦАП относят:

– число разрядов n управляющего кода;

– номинальный выходной ток;

– время установления выходного сигнала после изменения входного управляющего кода;

– погрешность полной шкалы;

– погрешность линейности;

– дифференциальная нелинейность.

Погрешности ЦАП могут быть выражены в процентах или других относительных единицах, а также в долях кванта.

Упрощенная схема цифро-аналогового преобразования входного кода в ток (или напряжение) определенной величины изображена на рисунке 1.8.

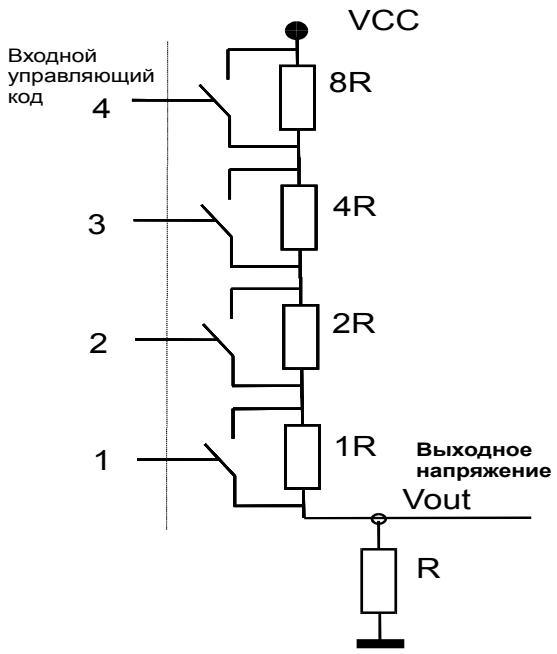

Рисунок 1.8 – Упрощенная схема цифро-аналогового преобразования

Если соотношение резисторов ЦАП как 8R, 4R, 2R, 1R, то при включении всех коммутаторов, в соответствии с законом Ома напряжение в точке OUT будет равняться напряжению VCC. Если включить все коммутаторы резисторов, кроме 1R, напряжение в точке OUT будет равно VCC/2. Аналогично можно знать значение напряжения, при различных комбинациях входного кода.

Схема реализации цифро-аналогового преобразования в учебно-отладочном стенде LPC2148 Education Board изображена на рисунке 1.9.

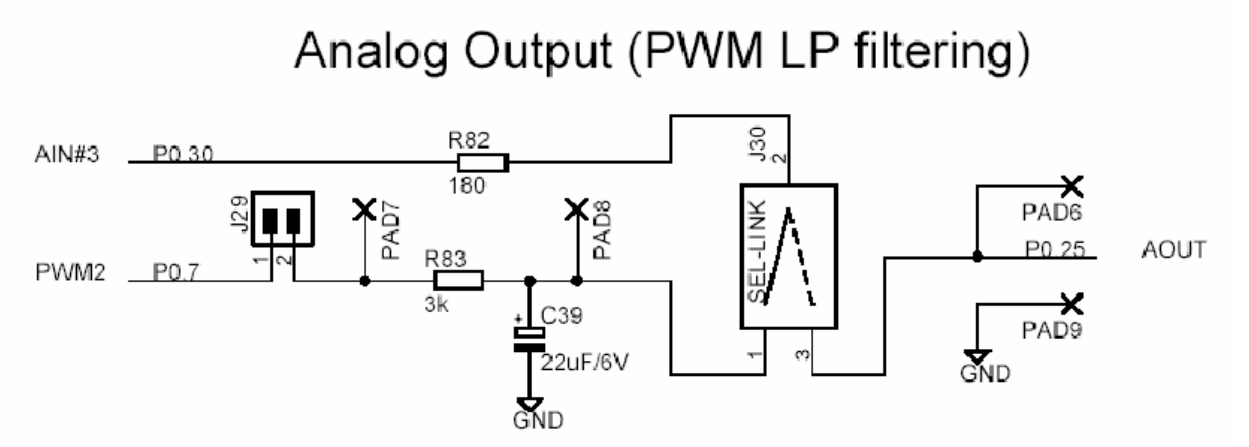

Рисунок 1.9 – Схема коммутации каналов ЦАП в учебно-отладочном стенде

Микроконтроллеры семейства LPC2000 содержат интегрированный (встроенный в микроконтроллер) модуль ЦАП, который имеет следующие особенности:

– цифро-аналоговый преобразователь с разрядностью кода в 10 бит;
– архитектура коммутируемой резистивной матрицы;
– буферизированный выход;
– поддержка энергосберегающего режима Power Down;
– время преобразования выбирается в зависимости от потребляемой мощности.

Описание выводов, которые относятся к модулю цифро-аналогового преобразования, входящего в состав микроконтроллера LPC2148 приведено в таблице 1.11.

Таблица 1.11 – Описание выводов модуля ЦАП

Наименование вывода	Функциональное назначение	Описание
A_{OUT}	Выход аналогового сигнала	По истечении интервала времени преобразования, напряжение на этом выводе (относительно V_{SSA}) становится равным величине: (VALUE/1024)* V_{REF}, где VALUE – значение кода, записанного в регистр DACR
V_{REF}	Вход опорного напряжения	Формирует опорное напряжение для каналов ЦАП
V_{DD}, V_{SS}	Вход цифрового напряжения питания, цифровая земля	Входы питания, необходимые для работы цифровых (логических) цепей модуля цифро-аналогового преобразования сигналов
V_{DDA}, V_{SSA}	Вход аналогового напряжения питания, аналоговая земля	Входы питания, необходимые для работы аналоговых цепей модуля цифро-аналогового преобразования сигналов, они должны быть гальванически развязаны с цифровыми входами питания с целью минимизации уровня шумов и предотвращения ошибок преобразования

Для управления работой модуля ЦАП микроконтроллеров семейства LPC2000 используется только один специальный регистр – регистр DACR.

В микроконтроллере LPC2148, этот регистр расположен по адресу 0xE006C000. Битовая структура регистра приведена ниже в таблице 1.12. В этом регистре, доступном как для чтения, так и для записи, содержится значение кода, которое и будет преобразовано в аналоговое напряжение, а также бит, который задает время установления сигналов модуля ЦАП. Биты 5:0 этого ре-

гистра зарезервированы для модулей ЦАП с более высоким разрешением, которые будут предлагаться производителем в перспективе.

Таблица 1.12 – Битовая структура регистра DACR

Номер бита	Название бита	Функция бита	Значение после сброса
5:0	Зарезервированы	Пользовательское ПО не должно производить запись в зарезервированные биты. Чтение зарезервированного бита возвращает неопределенное значение	0
15:6	VALUE	По истечении интервала времени преобразования, после того, как в регистр записано новое значение, напряжение на этом выводе (относительно V_{SSA}) становится равным величине: (VALUE /1024)* V_{REF}, где VALUE – значение кода, записанного в регистр DACR	0
16	BIAS	Если бит установлен в 0, максимальное время установления ЦАП составляет 1 мкс, а максимальный ток потребления – 700 мкА. Если бит установлен в 1, максимальное время установления ЦАП составляет 2.5 мкс, а максимальный ток потребления – 350 мкА	0
31:17	Зарезервированы	Пользовательское ПО не должно производить запись в зарезервированные биты. Чтение зарезервированного бита возвращает неопределенное значение	0

Модуль ЦАП может получить управление выводом P0.25/AD0.4/A_{OUT}, когда значение битов 19:18 в специальном регистре PINSEL1 равно 10b. При этом активизируется модуль ЦАП. У микроконтроллеров, не имеющих встроенного модуля ЦАП, значение 10b указанных битов является зарезервированным.

Значения времени установления ЦАП, задаваемые битом BIAS, справедливы, если входной импеданс (сопротивление) нагрузки, подключенной к выводу A_{OUT}, превышает определенную заданную величину. Невыполнение этого условия приведет к увеличению времени установления.

2 ПРИМЕРЫ ФУНКЦИЙ ДЛЯ УПРАВЛЕНИЯ ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ

В данном разделе будут приведены примеры функций, связанных с конфигурированием и управлением различными периферийными устройствами учебно-отладочного стенда.

2.1 Работа с полупроводниковым светодиодным индикатором

Данный полупроводниковый светодиодный индикатор состоит из восьми светодиодов, расположенных в линию. Схема подключения (реализованная в учебно-отладочном стенде) светодиодного индикатора к микроконтроллеру показана на рисунке 2.1.

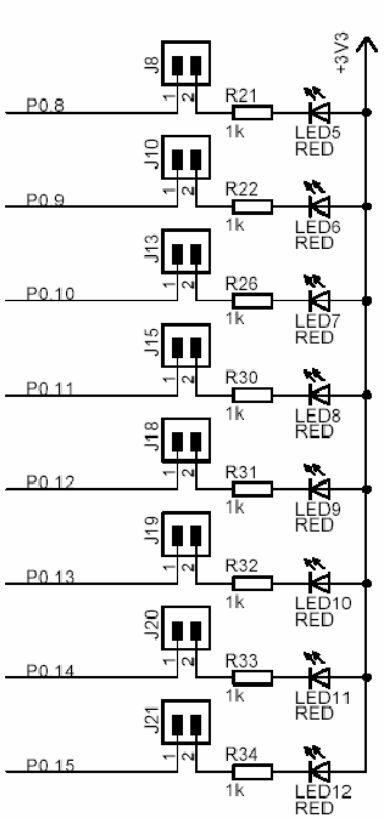

Рисунок 2.1 – Схема подключения светодиодного индикатора

Пример программы для зажигания всей линейки светодиодов:

```
#include <lpc21xx.h>

int main(void) {
     IODIR0 |= 0x0000FF00;
     IOCLR0 |= 0x0000FF00;
     while(1) {;}
}
```

2.2 Работа с кнопкой SW1

Данная кнопка обеспечивает возможность для управления некоторым устройством посредством механизма опроса её состояния, также она может являться источником внешнего прерывания EINT1. Схема подключения кнопки к микроконтроллеру показана на рисунке 2.2. При работе с кнопкой не следует забывать о существовании дребезга контактов и применять методы по его подавлению.

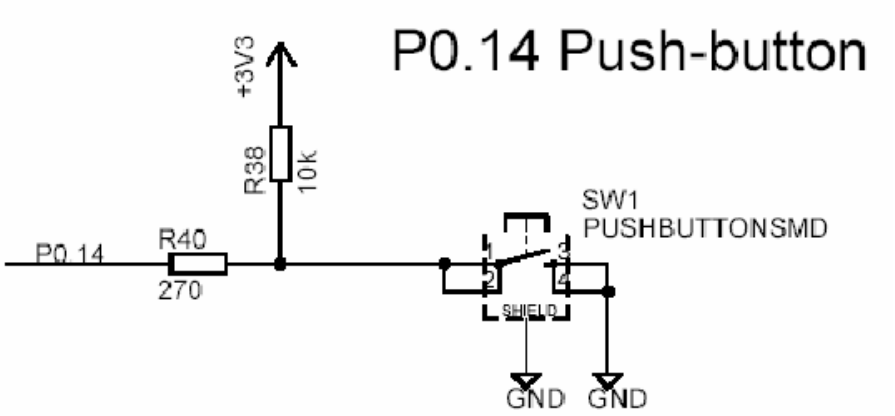

Рисунок 2.2 – Схема подключения кнопки к микроконтроллеру

Пример программы для опроса состояния кнопки. При каждом нажатии происходит мигание светодиодного индикатора:

```
#include <lpc21xx.h>

int delay(int _slp) {
     int i, j = 0;
     for (i = 0; i < _slp * 1000; i++)
          j = i + 1;
     return j;
}

int main(void) {
     IODIR0 |= 0x0000FF00;
     IOSET0 |= 0x0000FF00;
     while(1) {
          if (!(IOPIN0 & (1<<14))) {
               IOCLR0 |= 0x0000FF00;
               delay(800);
               IOSET0 |= 0x0000FF00;
          }
     }
}
```

2.3 Работа с джойстиком SW2

Кнопочный контакт джойстика может находиться в одном из пяти положений: сдвинут влево, вправо, вверх или вниз, нажат как кнопка. Он может быть использован для управления другими устройствами, однако не может является источником прерываний. Схема подключения джойстика к микроконтроллеру показана на рисунке 2.3.

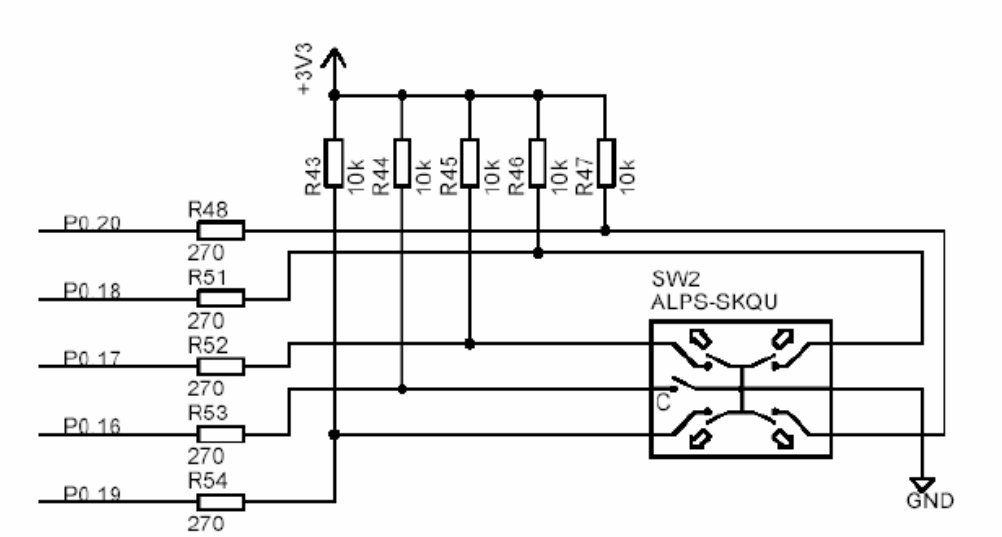

Рисунок 2.3 – Схема подключения джойстика к микроконтроллеру

Пример программы для опроса состояния джойстика и выполнения некоторых необходимых действий при каждом из нажатий:

```
#include <lpc21xx.h>

int delay(int _slp) {
     int i, j = 0;
     for (i = 0; i < _slp * 1000; i++)
          j = i + 1;
     return j;
}

int main(void) {
     IODIR0 |= 0x0000FF00;
     IOSET0 |= 0x0000FF00;
     while(1) {
          switch ((IOPIN0 & (0x1F<<16)) >> 16) {
               case 0x1D: <выполняемые действия> break;
               case 0x0F: <выполняемые действия> break;
               case 0x17: <выполняемые действия> break;
               case 0x1B: <выполняемые действия> break;
               default: break;
          }
     }
}
```

2.4 Работа с RGB-светодиодом

Трёхцветный светодиод (R-красный, G-зелёный, B-голубой) может применяться в качестве устройства отображения. По своей сути он представляет 3 полупроводниковых светодиода с разным цветом горения, размещённых в одном корпусе. Зажигание и потухание каждого из этих светодиодов может управляться отдельно, что позволяет генерировать различные цветовые комбинации. Схема подключения RGB-светодиода к микроконтроллеру показана на рисунке 2.4.

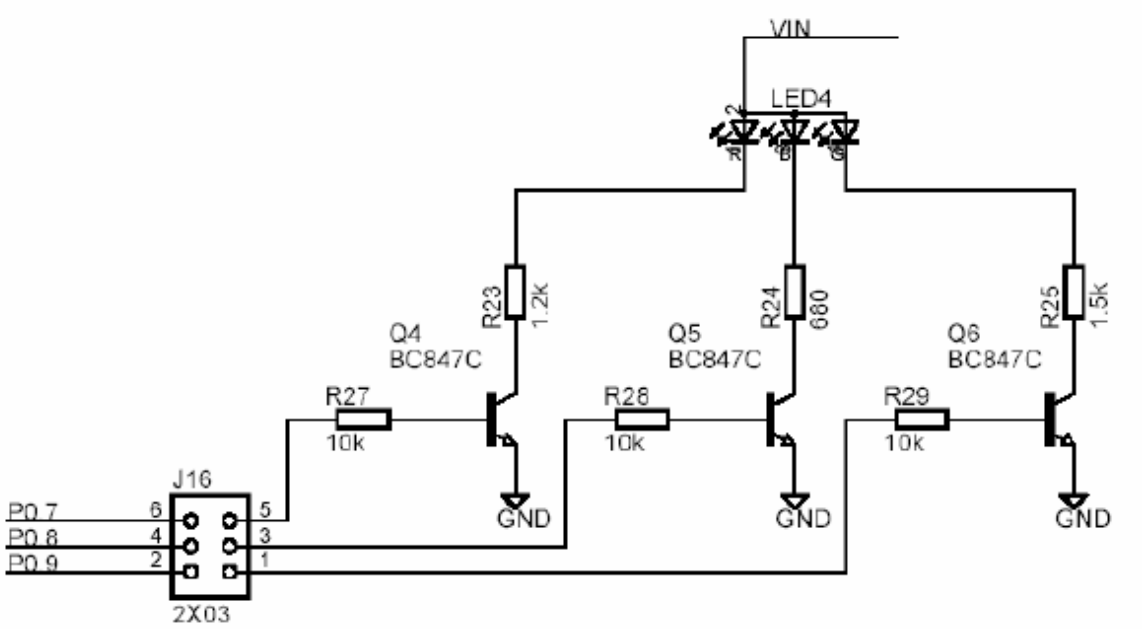

Рисунок 2.4 – Схема подключения RGB-светодиода к микроконтроллеру

Пример программы по управлению RGB-светодиодом, в которой происходит поочерёдное раздельное зажигание каждого из цветов:

```
#include <lpc21xx.h>

int delay(int _slp) {
     int i, j = 0;
     for (i = 0; i < _slp * 1000; i++)
          j = i + 1;
     return j;
}

int main(void) {
     IODIR0 |= 0x0000380;
     IOCLR0 |= 0x0000380;
     while(1) {
          IOSET0 |= (1<<7);
          delay (1000);
          IOCLR0 |= 0x0000380;
          IOSET0 |= (1<<8);
          delay (1000);
```

```
                IOCLR0 |= 0x0000380;
                IOSET0 |= (1<<9);
                delay (1000);
                IOCLR0 |= 0x0000380;
           }
}
```

2.5 Работа с пьезоэлектрическим динамиком

Пьезоэлектрический динамик предназначен для генерации звука определённой частоты и уровня. Идея его работы основывается на пьезоэффекте. Схема подключения RGB-светодиода к микроконтроллеру показана ниже на рисунке 2.5. Частота генерируемого звука будет тем выше, чем большее количество раз за период будет переключаться транзисторный ключ.

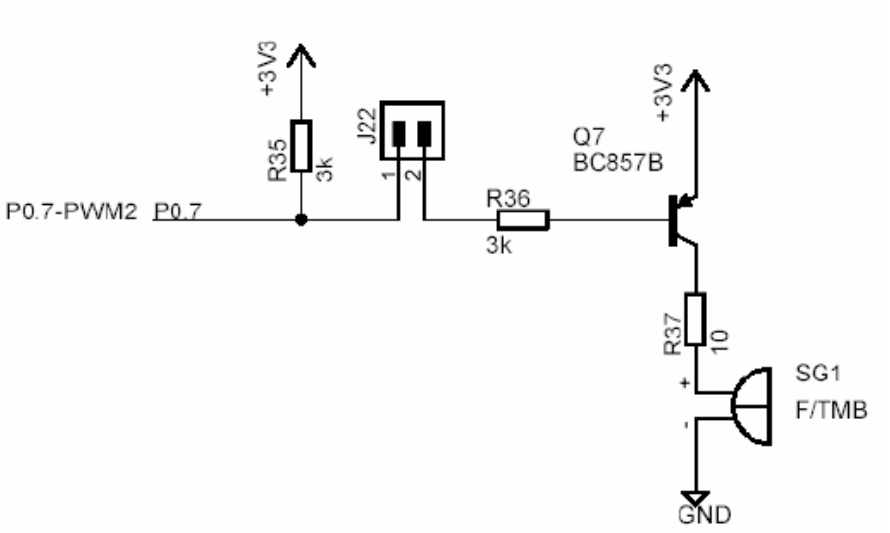

Рисунок 2.5 – Схема подключения пьезоэлектрического динамика к микроконтроллеру

Пример программы работы с динамиком:

```
#include <lpc21xx.h>

int delay(int _slp) {
     int i, j = 0;
     for (i = 0; i < _slp * 1000; i++)
          j = i + 1;
     return j;
}

int main(void) {
     IODIR0 |= (1<<7);
     IOSET0 |= (1<<7);
     while(1) {
          IOCLR0 |= (1<<7);
          delay(30);
          IOSET0 |= (1<<7);
```

```
            delay(20);
        }
    }
```

2.6 Работа с БДПТ

Бесколлекторный двигатель постоянного тока (БДПТ) применяется для вращения валов различных механизмов. В учебно-отладочном стенде он применяется для вращения лопастей вентилятора. Схема подключения БДПТ к микроконтроллеру показана ниже на рисунке 2.6. В зависимости от того, с какой частотой будут подаваться управляющие сигналы, БДПТ будет вращаться быстрее или медленнее.

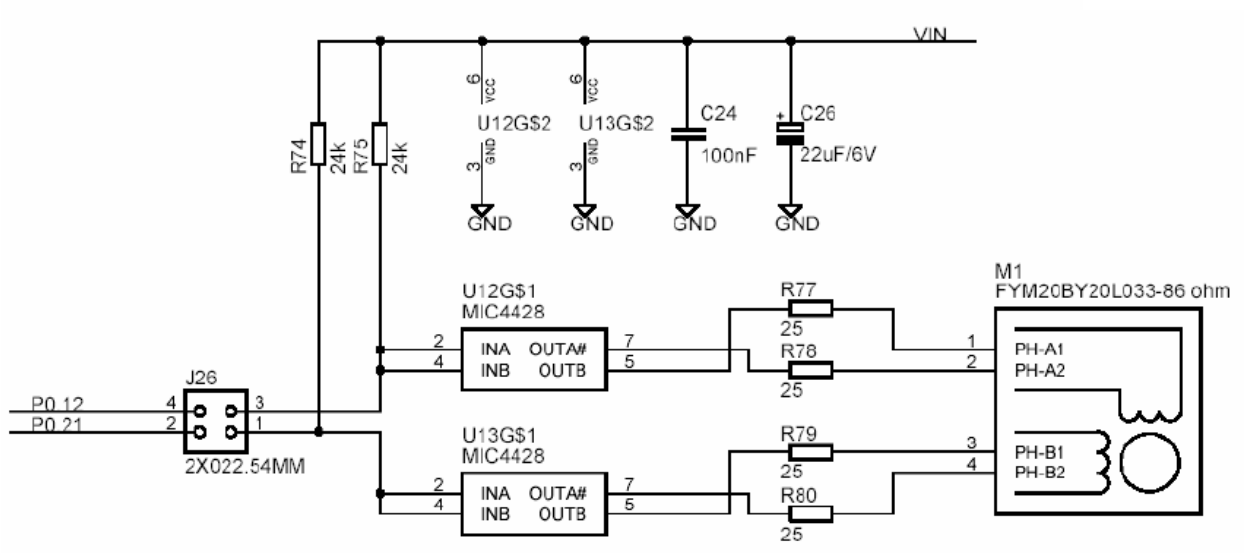

Рисунок 2.6 – Схема подключения БДПТ к микроконтроллеру

Управление направлением вращения ротора БДПТ осуществляется так же по сигнальным линиям P0.12 и P0.21 (смотри рисунок 2.7).

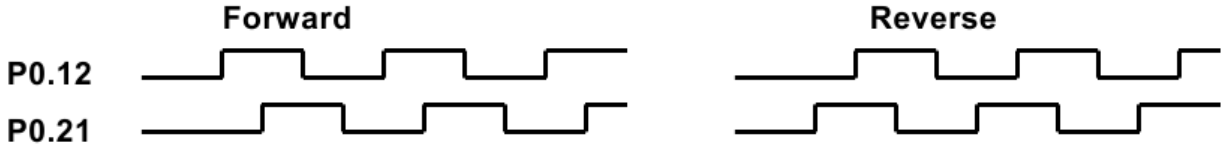

Рисунок 2.7 – Временные диаграммы генерации управляющих сигналов для вращения БДПТ по (слева) или против часовой стрелки (справа)

Пример программы управления БДПТ, в которой происходит его вращение по часовой стрелке:

```
#include <lpc21xx.h>

int delay(int _slp) {
     int i, j = 0;
```

```
     for (i = 0; i < _slp * 1000; i++)
          j = i + 1;
     return j;
}

int main(void) {
     unsigned char flag = 0;
     IODIR0 |= (1 << 12) | (1 << 21);
     IOCLR0 |= (1 << 12) | (1 << 21);
     while(1) {
               IOSET0 |= (1<<12);
               delay (125);
               IOSET0 |= (1<<21);
               delay (125);
               IOCLR0 |= (1<<12);
               delay (125);
               IOCLR0 |= (1<<21);
               delay (125);
     }
}
```

2.7 Конфигурация и обработка прерываний от таймера

Пример программы конфигурации и обработки прерывания по совпадению для таймера TIMER0:

```
#include <lpc21xx.h>

int led_count = 8;
//обработчик прерывания от Timer0
__irq void Timer0ISR(void) {
     if (++led_count == 16)
          led_count = 8;
     IOSET0 |= 0x0000FF00;
     T0IR = 0x01;
     VICVectAddr = 0;
     return;
}

int delay(int _slp) {
     int i, j = 0;
     for (i = 0; i < _slp * 1000; i++)
          j = i + 1;
     return j;
}

void InitTimer0(void) {
     /*****************************************************/
     /*              Инициализация VIC                    */
     /*****************************************************/
     VICDefVectAddr = (unsigned int) &Timer0ISR;
     VICIntEnable = 0x10; //Channel#4 is the Timer0
```

```
     VICIntSelect = 0x00; //all interrupts are IRQs
     /*************************************************/
     /*          Инициализация Timer0                 */
     /*************************************************/
     T0MR0 = 4500000; //Timer match (~ 0.1 second)
     T0MCR = 0x03; //Interrupt on Match0, reset timer on match
     T0PC = 0x01; // Prescaler to 2
     T0TC = 0x00; // reset Timer counter
     T0TCR = 0x01; // enable Timer
     return;
}

int main(void) {
     IODIR0 |= 0x0000FF00;
     IOSET0 |= 0x0000FF00;
     InitTimer0();
     while(1) {
          IOCLR0 |= (1 << led_count);
     }
}
```

2.8 Конфигурация и обработка внешнего прерывания EINT1

Пример программы конфигурации и обработки внешнего прерывания EINT1, которое генерируется по нажатию кнопки SW1:

```
#include <lpc21xx.h>

unsigned char flag = 0;

int delay(int _slp) {
     int i, j = 0;
     for (i = 0; i < _slp * 1000; i++)
          j = i + 1;
     return j;
}

__irq void eint1(void) {
     EXTINT |= 0x02;
     VICVectAddr = 0;
     flag = 1;
}

void init_eint1(void) {
     PINSEL0 |= 0x20000000;
     VICVectCntl0 = 0x0000002f;
     VICVectAddr0 = (unsigned)&eint1;
     VICIntEnable = 0x00008000;
}

int main(void) {
     IODIR0 |= 0x0000FF00;
```

```
      IOSET0 |= 0x0000FF00;
      init_eint1();
      while(1) {
           if (flag == 1) {
                IOCLR0 |= 0x0000FF00;
                delay(200);
                IOSET0 |= 0x0000FF00;
                flag = 0;
           }
      }
}
```

2.9 Работа с алфавитно-цифровым ЖКИ-модулем

ЖКИ-модуль применяется для отображения информации в символьном виде. Схема подключения ЖКИ-модуля к микроконтроллеру, реализованная в учебно-отладочном стенде LPC2148 Education Board, приведена ниже на рисунке 2.8.

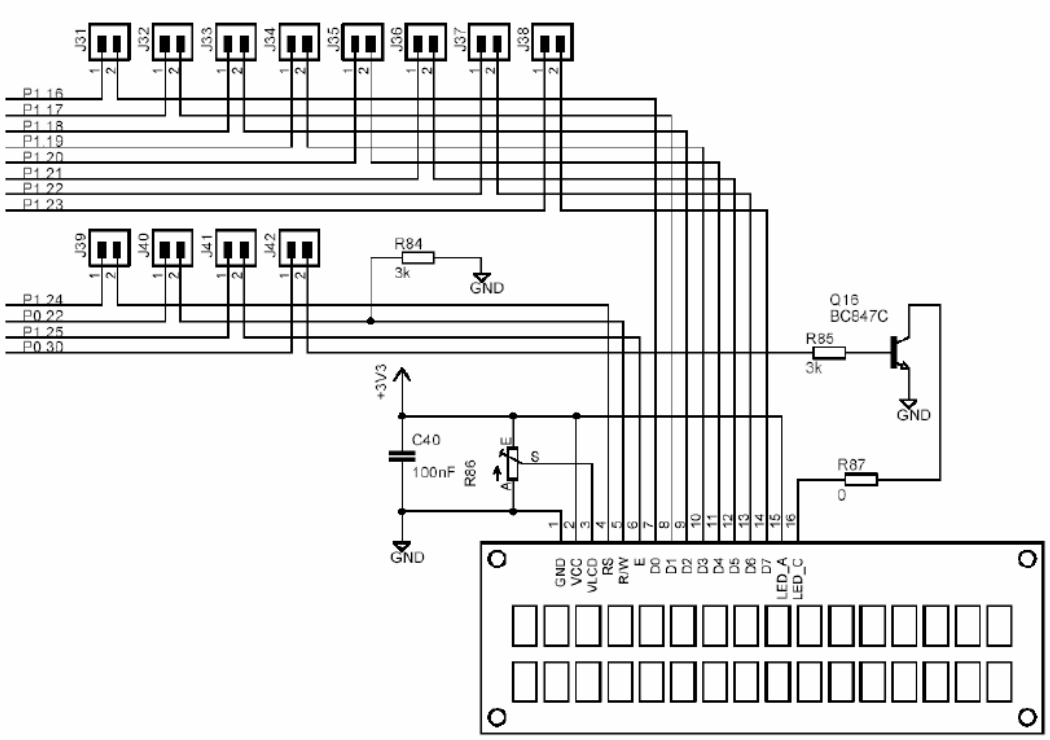

Рисунок 2.8 – Схема подключения ЖКИ-модуля к микроконтроллеру

Пример функции отправки команды в ЖКИ-модуль:

```
void SetCommLCD(uchar  bT) {
      IODIR1 |= (0xFF << 16); //Set data bus as outputs
      IODIR1 |= (1 << 24); //Set RS as output
      IODIR1 |= (1 << 25); //Set E as output
      IODIR0 |= (1 << 30); //Set Backlight control as output
      IODIR0 |= (1 << 22); //Set R/W as output
      IOCLR1 = (1 << 25); //E down
```

```
	IOCLR1 = (1 << 24); //RS = 0
	IOCLR0 = (1 << 22); //RW = 0
	IOCLR1 = (0xFF) << 16; //Clear data bus
	IOSET1 = (bT) << 16; //Write data to bus
	IOSET1 = (1 << 25); //E up
	Sleep (10); //Wait
	IOCLR1 = (1 << 25); //E down
}
```

Пример функции отправки данных (кода символа) в ЖКИ-модуль:

```
void SetDataLCD(uchar  bT) {
	IODIR1 |= (0xFF << 16); //Set data bus as outputs
	IODIR1 |= (1 << 24); //Set RS as output
	IODIR1 |= (1 << 25); //Set E as output
	IODIR0 |= (1 << 30); //Set Backlight control as output
	IODIR0 |= (1 << 22); //Set R/W as output
	IOCLR1 = (1 << 25); //E down
	IOSET1 = (1 << 24); //RS = 1
	IOCLR0 = (1 << 22); //RW = 0
	IOCLR1 = (0xFF) << 16; //Clear data bus
	IOSET1 = (bT) << 16; //Write data to bus
	IOSET1 = (1 << 25); //E up
	Sleep (10); //Wait
	IOCLR1 = (1 << 25); //E down
}
```

2.10 Работа со знакосинтезирующим индикатором при помощи интерфейса SPI

Знакосинтезирующий индикатор представляет собой поле, состоящее из полупроводниковых светодиодов. Все светодиоды собраны в группы или строки. В учебно-отладочном стенде таких строк насчитывается 8. Каждая строка содержит по 8 полупроводниковых светодиодов, объединённых по схеме с общим катодом. Однако известно, что для зажигания светодиода нужно приложить к его выводам разность потенциалов правильной полярности (открывающую). Аноды всех светодиодов строки собраны в группу (или столбец), таких столбцов тоже 8. Таким образом, для того, чтобы зажечь светодиод, необходимо установить соответствующие уровни напряжений на его выводах, которые, в свою очередь подключены к выводам двух последовательно-параллельных регистров. Эти регистры обмениваются информацией с микроконтроллером при помощи интерфейса SPI. Как видим, здесь реализовано каскадное (или последовательное) подключение ведомых устройств по интерфейсу SPI. Схема подключения знакосинтезирующего индикатора к микроконтроллеру посредством интерфейса SPI, реализованная в учебно-отладочном стенде LPC2148 Education Board, приведена ниже на рисунке 2.9.

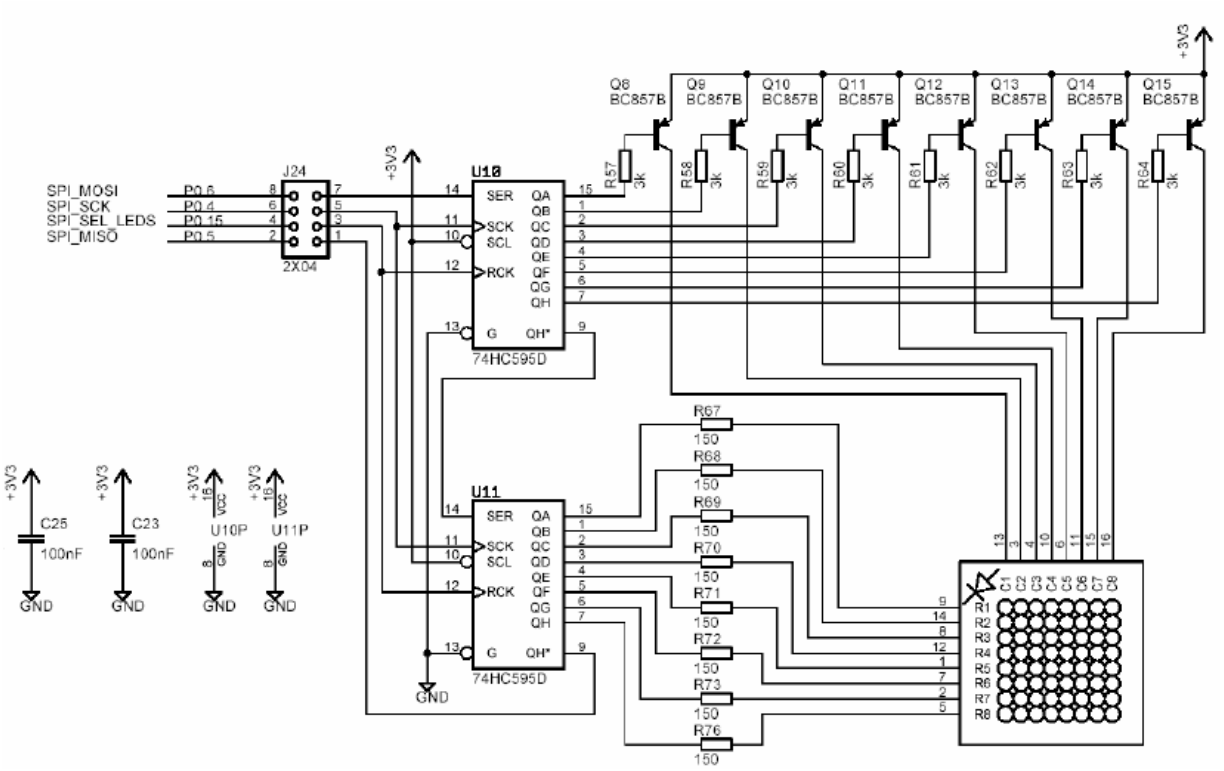

Рисунок 2.9 – Схема подключения знакосинтезирующего индикатора к микроконтроллеру посредством интерфейса SPI

Пример функции зажигания светодиода в некоторой строке и столбце индикатора:

```
void SPIPutDot (int x, int y) {
     IOSET0 = (1 << 15);
     S0SPDR = ~(1 << (y - 1));
     while ((S0SPSR & (1 << 7)) == 0);
     S0SPDR = ~(1 << (8 - x));
     while ((S0SPSR & (1 << 7)) == 0);
     IOCLR0 = (1 << 15);
}
```

2.11 Работа с аналого-цифровым преобразователем

Аналого-цифровой преобразователь (АЦП) предназначен для перевода сигнала из аналоговой формы в цифровую. Схема подключения аналого-цифрового преобразователя к микроконтроллеру, реализованная в учебно-отладочном стенде LPC2148 Education Board, приведена ниже на рисунке 2.10. Хотя АЦП и содержит 8 каналов для преобразования, в стенде используются только 2 канала.

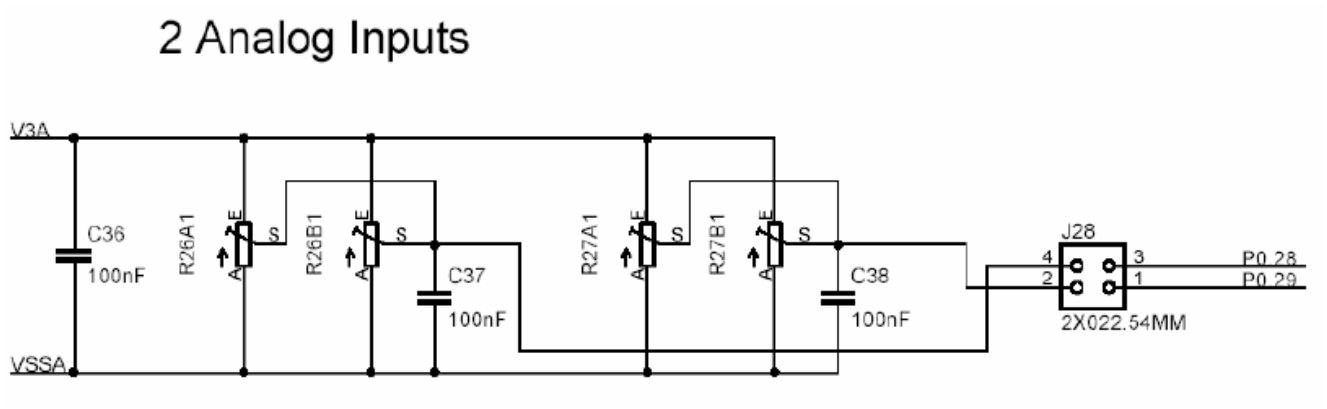

Рисунок 2.10 – Схема подключения аналого-цифрового преобразователя к микроконтроллеру

Пример функции для инициализации АЦП:

```
void InitADC (void) {
     PINSEL1 |= (1 << 24);
     /*Write 01 to 25:24 bits of PINSEL1 to select the ADC-
     input function for P0.28 - AD0.1*/
     PINSEL1 |= (1 << 26);
     /*Write 01 to 27:26 bits of PINSEL1 to select the ADC-
     input function for P0.29 - AD0.*/
     AD0CR &= 0x00000000; //Reset ADC0
     //Setup ADC – AD0.1 and AD0.2 input, 11 clocks/10 bits
     AD0CR |= (1 << 2); //Select ADC channel
     AD0CR |= (1 << 16); //Automatically repeat transform
     AD0CR |= (1 << 21); //Power-on ADC
}
```

Пример функции, которая выполняет чтение данных из АЦП после окончания преобразования:

```
int ADCReadValue (void) {
     int ADCValue= 0;
     AD0CR = 0x00210002; //Setup ADC AD0.1 input (P0.28)
     AD0CR |= 0x01000000; //Start of transformation
     //Waiting for the end of transformation
     while (AD0GDR == 0x8000000);
     ADCValue = AD0DR1; //Read all data from ADC
     //Read 10-bits result of transformation
     ADCValue = ((ADCValue >>6) & 0x03FF);
     return ADCValue;
}
```

3 ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРОВ НА БАЗЕ ЯДРА ARM7

Изучение дисциплины “Микроконтроллерные системы автоматики и управления” эффективно лишь тогда, когда наряду с овладением теорией студенты в условиях проведения лабораторного практикума самостоятельно разрабатывают алгоритмы управления различными объектами при помощи языка программирования С, знакомятся с современными интегрированными средами разработки.

Предлагаемые методические указания посвящены изучению языка С для программирования микроконтроллеров, а также интегрированной среды разработки (ИСР) Keil uVision, которая отличается от других известных аналогичных интегрированных сред разработки:

– простотой и удобством в обращении;

– развитой системой библиотек, обеспечивающих поддержку огромного количества микроконтроллерных устройств, а также отладочных плат;

– наличием бесплатных версий;

– системой контроля версий проекта;

– наличием развитых средств отладки программных проектов (в том числе и пошаговой отладки в готовом устройстве);

– развитой и удобной для разработчика системой меню;

– интуитивно понятным графическим интерфейсом пользователя.

В ходе выполнения лабораторных работ студенты должны:

– изучить особенности архитектуры и структурного построения микроконтроллеров семейства ARM7;

– изучить особенности языка С для программирования микроконтроллеров семейства ARM7;

– получить знания о методах практического применения микроконтроллеров и систем на их основе при проектировании и разработке электронных систем различного назначения;

– освоить основные подходы и особенности написания программ для микроконтроллеров;

– изучить некоторые особенности обращения с ИСР Keil uVision, которые обязательно надо учитывать при написании программ на языке С.

Подготовка к лабораторной работе предусматривает обязательное изучение студентами теоретического материала.

Перед выполнением лабораторных работ студенты должны:

– внимательно ознакомиться с теоретическими сведениями, которые относятся к конкретной лабораторной работе;

– изучить и получить навыки обращения с встроенными типовыми компонентами ИСР Keil uVision, которые доступны разработчику;

– произвести анализ полученного в соответствии с установленным вариантом задания алгоритма и разработать схему алгоритма для его реализации на языке С.

Выполнение лабораторной работы рекомендуется начинать с ознакомления с теоретическим материалом, который относятся непосредственно к теме лабораторной работы. Затем следует разрабатывать схему алгоритма по управлению конкретным периферийным устройством с учётом изученных ранее особенностей его функционирования. И уже на завершающем этапе нужно приступать к реализации данного алгоритма доступными средствами языка С, выполняя, при необходимости, отладку разработанной программы с целью поиска и устранения допущенных ошибок.

Отчет о выполнении каждой лабораторной работы обязательно должен содержать такие структурные элементы:

- название лабораторной работы;
- цель работы;
- краткие теоретические сведения;
- порядок выполнения работы;
- результаты выполнения работы:
  - схема разработанного алгоритма управления периферийным устройством отладочного стенда;
  - текст программы, написанной на языке С, представленный с необходимым форматированием и содержащий подробные комментарии;
- особенности функционирования интегрированной среды разработки Keil uVision, выявленные в ходе выполнения лабораторной работы;
- выводы.

4 ЛАБОРАТОРНАЯ РАБОТА №1. ПОРТЫ ВВОДА/ВЫВОДА ОБЩЕГО НАЗНАЧЕНИЯ. УПРАВЛЕНИЕ КОНТРОЛЛЕРОМ ОБРАБОТКИ ВЕКТОРОВ ПРЕРЫВАНИЙ

Цель работы: научиться управлять работой портов ввода/вывода общего назначения, а также изучить основные возможности контроллера обработки прерываний.

4.1 Порядок выполнения работы

4.1.1 Изучите до начала выполнения лабораторной работы основные возможности использования портов ввода/вывода общего назначения, а также работу контроллера прерываний.

4.1.2 Изучите примеры программирования работы портов ввода/вывода общего назначения, а также контроллера обработки прерываний, приведенные в разделе 2 данных методических указаний.

4.1.3 Разработайте схему алгоритма управления светодиодным индикатором с учетом установленного варианта задания по таблице 4.1.

4.1.4 Запустите интегральную среду разработки Keil uVision.

4.1.5 Создайте проект, в котором опишите разработанный алгоритм управления при помощи языка С.

Таблица 4.1 – Варианты заданий

№ варианта	Формулировка задания
1	2
1	Реализовать алгоритм управления светодиодным индикатором при помощи джойстика SW2. При нажатии джойстика влево или вправо некоторый светодиод перемещается в выбранном направлении, при нажатии вверх или вниз – остаётся на прежней позиции. Перемещение светодиода происходит по срабатыванию прерывания от таймера (частота выбирается произвольно).
2	Реализовать алгоритм отображения на светодиодном индикаторе количества обработанных внешних прерываний, которые генерируются при нажатии на кнопку SW1. Дребезг контактов должен устраняться программным способом.
3	Реализовать алгоритм плавного зажигания и потухания светодиодного индикатора.
4	Реализовать алгоритм горения светодиодного индикатора с различной градацией яркости. Соседние светодиоды должны отличаться по яркости на 1 шаг градации – 1/8 от максимальной яркости, т.е. первый светодиод светится с яркостью 1/8, второй – 2/8 и т.д. Последний светодиод – с максимальной яркостью.

Продолжение таблицы 4.1

1	2
5	Реализовать алгоритм управления частотой звука при помощи джойстика. Шаг изменения частоты 100 Гц. Одно нажатие джойстика должно отвечать одному шагу изменения частоты в зависимости от направления движения: вверх – частота увеличивается, вниз – уменьшается.

4.1.6 Откомпилируйте созданный проект. При наличии сообщений об ошибках или предупреждениях вернитесь к предыдущему пункту и внесите необходимые изменения.

4.1.7 Включите учебно-отладочный стенд. При помощи утилиты Flash Magic загрузите файл с расширением *.hex, находящийся в папке проекта, в стенд.

4.1.8 Визуально оцените правильность функционирования разработанного алгоритма.

4.2 Контрольные вопросы

4.2.1 Объясните внутреннюю организацию портов ввода/вывода общего назначения микроконтроллеров с ядром ARM7.

4.2.2 Объясните механизм генерации и обработки прерываний микроконтроллеров с ядром ARM7.

4.2.3 Перечислите и объясните известные методы подавления дребезга контактов.

4.2.4 Объясните внутреннюю организацию таймера-счётчика TIMER0 микроконтроллера LPC2148.

4.2.5 Объясните механизм генерации прерываний таймера-счётчика TIMER0.

4.2.6 Объясните физический принцип работы полупроводникового светоизлучающего диода.

4.2.7 Перечислите основные преимущества и недостатки применения индикаторов на полупроводниковых светодиодах в цифровых устройствах.

5 ЛАБОРАТОРНАЯ РАБОТА №2. ЖИДКОКРИСТАЛЛИЧЕСКИЙ ИНДИКАТОР

Цель работы: изучить внутреннюю структуру, особенности организации памяти модуля жидкокристаллического индикатора, организацию обмена данными с ним, а также научиться отображать символьные данные на жидкокристаллическом индикаторе.

5.1 Порядок выполнения работы

5.1.1 Изучите до начала выполнения работы внутреннее устройство жидкокристаллического индикатора, основные принципы его работы, а также схему подключения ЖКИ-модуля к микроконтроллеру, реализованную в учебно-отладочном стенде LPC2148 Education Board.

5.1.2 Изучите примеры программирования работы ЖКИ-модуля, приведенные в разделе 2 данных методических указаний.

5.1.3 Разработайте схему алгоритма управления ЖКИ-модулем с учетом установленного варианта задания по таблице 5.1.

Таблица 5.1 – Варианты заданий

№ варианта	Формулировка задания
1	Реализовать алгоритм вывода данных на ЖКИ. Данные в виде целых чисел от 0 до 9 циклически поступают и выводятся в произвольном месте индикатора при срабатывании прерывания от таймера TIMER0, частота работы которого – 2 Гц.
2	Реализовать алгоритм вывода данных на ЖКИ. Данные в виде собственноручно созданных символов (не менее 2-х символов) должны выводиться в произвольные знакоместа индикатора.
3	Реализовать алгоритм отображения состояния электронного секундомера (в формате мм:сс) на ЖКИ, используя 4-х битный интерфейс обмена. При этом на ЖКИ должен отображаться текущий ход минут и секунд. Управление работой секундомера производиться с помощью нажатий джойстика SW2: вправо – запуск секундомера, влево – остановка, вниз – остановка и сброс.
4	Реализовать алгоритм управления курсором на ЖКИ. В роли курсора выступает собственноручно созданный символ. Курсор перемещается только в пределах видимой области ЖКИ. Перемещение осуществляется с помощью нажатий джойстика SW2.
5	Реализовать алгоритм вывода бегущей строки, состоящей из фамилий студентов бригады, на ЖКИ. Данные двигаются по индикатору в направлении, которое задаётся смещением джойстика SW2 в соответствующую сторону. Нажатие джойстика вверх ускоряет смещение строки в 2 раза, а вниз – замедляет в 2 раза.

5.1.4 Запустите интегральную среду разработки Keil uVision.

5.1.5 Создайте проект, в котором опишите разработанный алгоритм управления при помощи языка С.

5.1.6 Откомпилируйте созданный проект. При наличии сообщений об ошибках или предупреждениях вернитесь к предыдущему пункту и внесите необходимые изменения.

5.1.7 Включите учебно-отладочный стенд. При помощи утилиты Flash Magic загрузите файл с расширением *.hex, находящийся в папке проекта, в стенд.

5.1.8 Визуально оцените правильность функционирования разработанного алгоритма.

5.2 Контрольные вопросы

5.2.1 Перечислите основные преимущества и недостатки применения жидкокристаллических индикаторов в цифровых устройствах.

5.2.2 Приведите и объясните алгоритм создания символа пользователя для ЖКИ.

5.2.3 Приведите и объясните структуру модуля ЖКИ.

5.2.4 Объясните, каким образом представляется символ в ПЗУ и ОЗУ знакогенератора контроллера ЖКИ.

5.2.5 Объясните схему подключения ЖКИ-модуля к микроконтроллеру, реализованную в учебно-отладочном стенде.

5.2.6 Каким образом можно вычислить максимальную частоту записи команд в ЖКИ-модуль?

5.2.7 Объясните временные диаграммы чтения и записи команд или данных в ЖКИ-модуль.

6 ЛАБОРАТОРНАЯ РАБОТА №3. ИНТЕРФЕЙС SPI

Цель работы: изучить внутреннюю организацию, режимы обмена данными, конфигурацию режимов, а также протокол функционирования синхронного последовательного полнодуплексного интерфейса SPI.

6.1 Порядок выполнения работы

6.1.1 Изучите до начала выполнения работы внутреннюю организацию модуля приёма/передачи данных по интерфейсу SPI, протокол обмена данными, конфигурацию режимов обмена данными, а также схему подключения светодиодного знакосинтезирующего индикатора к микроконтроллеру, реализованную в учебно-отладочном стенде LPC2148 Education Board.

6.1.2 Изучите примеры программирования модуля обмена данными по интерфейсу SPI, приведенные в разделе 2 данных методических указаний.

6.1.3 Разработайте схему алгоритма управления знакосинтезирующим индикатором с учетом установленного варианта задания по таблице 6.1.

Таблица 6.1 – Варианты заданий

№ варианта	Формулировка задания
1	2
1	Реализовать алгоритм свечения/гашения на знакосинтезирующем индикаторе “шахматной доски” при срабатывании внешнего прерывания EINT1, которое генерируется по нажатию кнопки SW1. Дребезг контактов должен устраняться программным способом.
2	Реализовать алгоритм вывода на знакосинтезирующий индикатор последовательности цифр от 0 до 9. Цифры начинают выводиться при срабатывании внешнего прерывания EINT1, которое генерируется по нажатию кнопки SW1. Дребезг контактов должен устраняться программным способом. Повторное нажатие на SW1 приостанавливает процесс вывода, при этом индикатор сохраняет текущее состояние.
3	Реализовать алгоритм управления светодиодом на знакосинтезирующем индикаторе с помощью джойстика SW2. Изначально в произвольном месте индикатора загорается один светодиод. После каждого нажатия джойстика с задержкой 0,2 с зажжённый светодиод смещается в заданном направлении.
4	Реализовать алгоритм перемещения светодиода на знакосинтезирующем индикаторе. Изначально в произвольном месте индикатора загорается светодиод. В дальнейшем он двигается с частотой 2 Гц по индикатору под углом 45°. При достижении края, “мячик” должен отбиваться в противоположную сторону под углом 45°. Перемещение светодиода приостанавливается по нажатию кнопки SW1, при этом выводится количество “ударов” об края знакосинтезирующего индикатора на светодиодный индикатор.

Продолжение таблицы 6.1

1	2
5	Реализовать алгоритм отображения шкалы на знакосинтезирующем индикаторе. В зависимости от количества нажатий джойстика SW2 загорается определённое количество строк индикатора. Каждое нажатие джойстика вверх добавляет по одной светящейся строке, вниз – уменьшает. При достижении предельных состояний (вся шкала заполнена, или вся шкала пуста) соответствующие нажатия джойстика вверх или вниз не оказывают влияния.

6.1.4 Запустите интегральную среду разработки Keil uVision.

6.1.5 Создайте проект, в котором опишите разработанный алгоритм управления при помощи языка С.

6.1.6 Откомпилируйте созданный проект. При наличии сообщений об ошибках или предупреждениях вернитесь к предыдущему пункту и внесите необходимые изменения.

6.1.7 Включите учебно-отладочный стенд. При помощи утилиты Flash Magic загрузите файл с расширением *.hex, находящийся в папке проекта, в стенд.

6.1.8 Визуально оцените правильность функционирования разработанного алгоритма.

6.2 Контрольные вопросы

6.2.1 Приведите и поясните временные диаграммы обмена данными по интерфейсу SPI в режимах 0 и 1.

6.2.2 Приведите и поясните временные диаграммы обмена данными по интерфейсу SPI в режимах 2 и 3.

6.2.3 Объясните механизм конфигурации режимов обмена данными по интерфейсу SPI.

6.2.4 Объясните механизм конфигурации выводов модуля приёмопередачи данных по интерфейсу SPI.

6.2.5 Какой способ подключения ведомых устройств реализован в учебно-отладочном стенде?

6.2.6 Дайте общую характеристику интерфейса SPI.

6.2.7 Охарактеризуйте особенности обмена данными между микроконтроллером и знакосинтезирующим индикатором.

7 ЛАБОРАТОРНАЯ РАБОТА №4. АНАЛОГО-ЦИФРОВОЕ ПРЕОБРАЗОВАНИЕ

Цель работы: научиться выполнять аналого-цифровое преобразование сигналов.

7.1 Порядок выполнения работы

7.1.1 Изучите до начала выполнения работы известные принципы аналого-цифрового преобразования, внутреннюю организацию аналого-цифрового преобразователя, а также схему подключения модуля аналого-цифрового преобразования, входящего в состав микроконтроллера, реализованную в учебно-отладочном стенде LPC2148 Education Board.

7.1.2 Изучите примеры программирования модуля аналого-цифрового преобразования, приведенные в разделе 2 данных методических указаний.

7.1.3 Разработайте схему алгоритма управления аналого-цифровым преобразователем с учетом установленного варианта задания по таблице 7.1.

Таблица 7.1 – Варианты заданий

№ варианта	Формулировка задания
1	Реализовать алгоритм чтения данных из АЦП и вывода их на знакосинтезирующий индикатор в виде шкалы. В зависимости от диапазона, в котором находится текущее значение входного напряжения, загорается определённое количество строк индикатора.
2	Реализовать алгоритм чтения данных из АЦП и вывода их на ЖКИ в формате: "Voltage: *.* V". При увеличении или уменьшении входного напряжения с помощью потенциометра значение, отображаемое на ЖКИ, изменяется.
3	Реализовать алгоритм чтения данных из АЦП и вывода их на светодиодный индикатор в виде шкалы. В зависимости от диапазона, в котором находится текущее значение входного напряжения, загорается определённое количество.
4	Реализовать алгоритм чтения данных из АЦП и вывода их на синтезатор тона. В зависимости от диапазона, в котором находится текущее значение входного напряжения, синтезатор генерирует звук определённой частоты. Если входное напряжение лежит в диапазоне 0 В – 1 В, генерируется звук с частотой 100 Гц; если в диапазоне 1 В – 2 В, генерируется звук с частотой 300 Гц; если в диапазоне 2 В – 3 В, генерируется звук с частотой 500 Гц
5	Реализовать алгоритм чтения данных из АЦП и вывода их на светодиодный индикатор. В зависимости от диапазона, в котором находится текущее значение входного напряжения, все светодиоды индикатора мигают с определённой частотой. Шаг изменения частоты между соседними диапазонами – 1 Гц.

7.1.4 Запустите интегральную среду разработки Keil uVision.

7.1.5 Создайте проект, в котором опишите разработанный алгоритм управления при помощи языка С.

7.1.6 Откомпилируйте созданный проект. При наличии сообщений об ошибках или предупреждениях вернитесь к предыдущему пункту и внесите необходимые изменения.

7.1.7 Включите учебно-отладочный стенд. При помощи утилиты Flash Magic загрузите файл с расширением *.hex, находящийся в папке проекта, в стенд.

7.1.8 Визуально оцените правильность функционирования разработанного алгоритма.

7.2 Контрольные вопросы

7.2.1 Объясните функциональное назначение микросхем аналого-цифрового преобразования.

7.2.2 Приведите статические и динамические параметры аналого-цифровых преобразователей.

7.2.3 Объясните механизм аналого-цифрового преобразования, реализованный в учебно-отладочном стенде.

7.2.4 Какова максимальная частота выполнения аналого-цифрового преобразования в учебно-отладочном стенде?

7.2.5 Какие особенности следует учитывать при использовании микросхем аналого-цифрового преобразования.

7.2.6 Поясните наиболее распространённые из существующих способов осуществления аналого-цифрового преобразования.

7.2.7 Перечислите все регистры, которые используются для управления аналого-цифровым преобразователем в учебно-отладочного стенда.

РЕКОМЕНДОВАННАЯ ЛИТЕРАТУРА

1 LPC2148 Education Board User's Guide . Datasheet. EA2-USG-0601 v2.1 Rev B. © Embedded Artists AB 2005 – 2007. – 51 p.

2 Muresan Radu. Embedded System Development and Labs for ARM / Radu Muresan. – Embest Inc., 2005. – 291 p.

3 Схемотехніка електронних систем: У 3 кн. Кн. 3. Мікропроцесори та мікро-контролери: Підручник/ В.І.Бойко, А.М.Гуржій, В.Я.Жуйков та ін. – 2-ге вид., допов. і переробл. К.: Вища шк., 2004. – 399 с.

4 Васильев А.Е. Микроконтроллеры. Разработка встраиваемых приложений. – СПб.: БХВ-Петербург, 2008. – 304 с.: ил.

5 Мартин Т. Микроконтроллеры ARM7. Семейство LPC2000 компании Philips. Вводный курс/Пер. с англ. – М.: Издательский дом «Додэка-XXI», 2006. – 240 с.: ил.

6 Редькин П.П. Микроконтроллеры ARM7 семейства LPC2000. Руководство пользователя. – М.: Издательский дом «Додэка-XXI», 2007. – 560 с.: ил.

7 Швец В.А., Шестакова В.В., Бурцева Н.В., Мелешко Т.В. Одноплатные микроконтроллеры. Проектирование и применение. – К.: «МК-Пресс», 2005. – 304 с.: ил.