Министерство образования и науки Российской Федерации

Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых

Кафедра управления и информатики в технических
и экономических системах

Программирование однокристальных микроЭВМ ADuC816

Методические указания к лабораторным работам по дисциплине

МИКРОКОНТРОЛЛЕРЫ И УСТРОЙСТВА СОПРЯЖЕНИЯ С ОБЪЕКТОМ

Составители:
Кочуров О. М.
Кокорин С. А.

Владимир 2011

УДК 004.31

Рецензент

доцент Владимирского государственного университета
В. С. Грибакин

Программирование однокристальных микроЭВМ ADuC816**:** Методические указания к лабораторным работам по дисциплине «Микроконтроллеры и устройства сопряжения с объектом» / Составители: Кочуров О. М., Кококрин С. А. — Владимир: ВлГУ — 2011, 35 с.

Приведены описания лабораторных работ, посвященных программированию распространенных 8-разрядных микроконтроллеров ADuC816 фирмы Analog Devices. Рассматривается архитектура, система команд микроконтроллеров, основные принципы разработки и отладки программ в среде Keil μVision 3. Изучаются приемы использования встроенных таймеров, часов реального времени, аналого-цифрового преобразователя, асинхронного приемопередатчика. Предлагается разработка нескольких программ, представляющих определенных практический интерес, таких как часы, термометр, вольтметр.

Предназначены для студентов специальности 220201 «Управление и информатика в технических системах».

Ил. 10, Табл. 6. Библиогр.: 3 назв.

УДК 004.31

1 Описание лабораторного макета

Лабораторный макет состоит из миниатюрной печатной платы высокого класса плотности монтажа, на которой установлены микроконтроллер, жидкокристаллический индикатор и другие элементы, и интерфейсной платы, предназначенной для сопряжения лабораторного макета с персональным компьютером. Питание макета осуществляется постоянным напряжением 9…24 В. Макет комплектуется сетевым источником питания с выходным напряжением 12 В.

Структурная схема основной платы лабораторного макета показана на рисунке 1. Электрическая принципиальная схема приведена на рисунке 2.

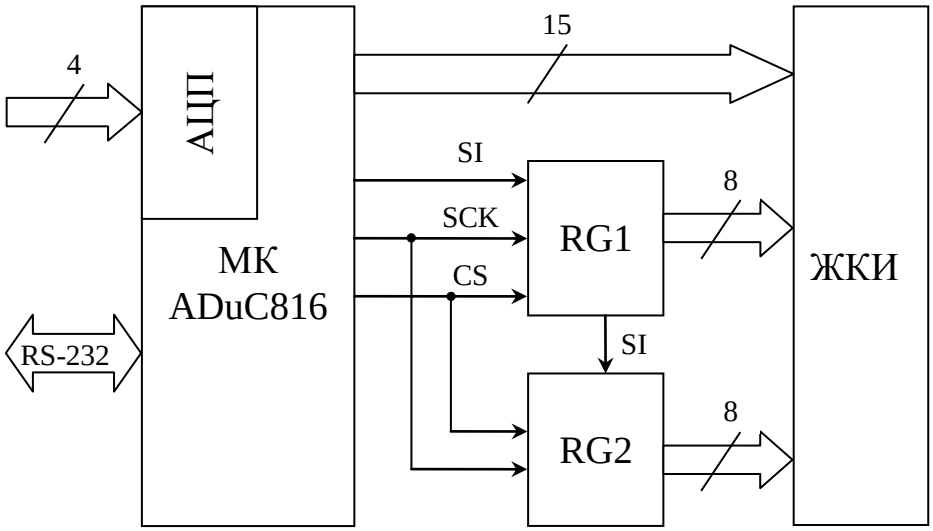

Рисунок 1 – Структурная схема лабораторного макета

Основу устройства составляет восьмиразрядный однокристальный микроконтроллер (МК) ADuC816 фирмы Analog Devices.

Микроконтроллер имеет встроенный АЦП с четырьмя входами для аналоговых сигналов.

Связь микроконтроллера с персональным компьютером осуществляется посредством последовательного интерфейса RS-232. Интерфейс предназначен для загрузки программы в резидентную память МК и обмена данными с персональным компьютером во время работы программы.

К микроконтроллеру подключен жидкокристаллический семисегментный индикатор (ЖКИ). Индикатор имеет четыре разряда. Поскольку для непосредственного управления сегментами индикатора число цифровых выходов МК недостаточно, его параллельный интерфейс расширен с помощью двух микросхем восьмиразрядных регистров (на структурной схеме RG1, RG2, на принципиальной схеме D2, D3).

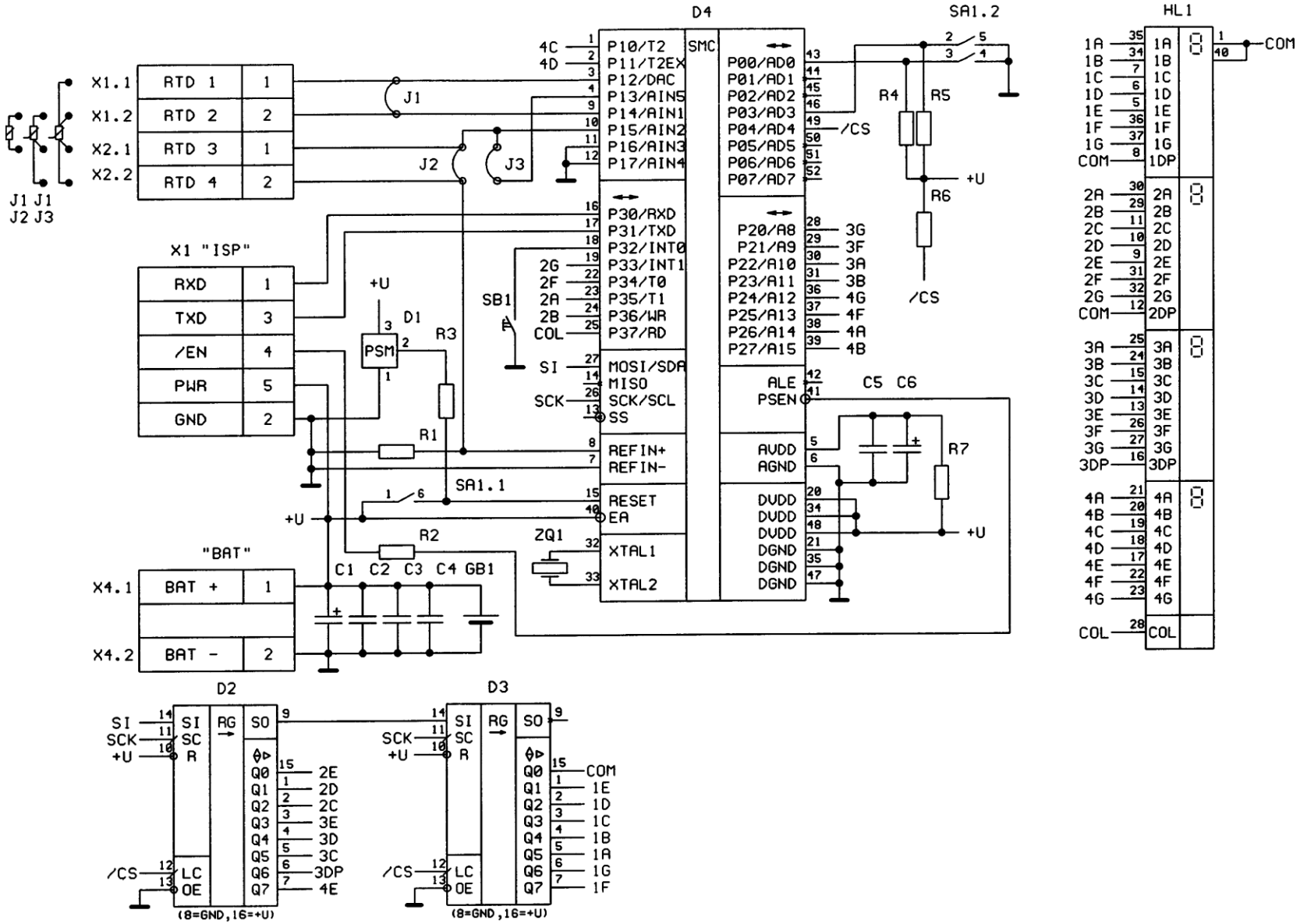

Рисунок 2 – Принципиальная электрическая схема лабораторного макета

Таким образом, часть контактов ЖКИ подключена к МК непосредственно, остальные подключены к выводам регистров. Разводку управляющих входов индикатора следует смотреть на принципиальной схеме.

Запись данных в регистры осуществляется через последовательный интерфейс SPI (сигналы SI, SCK). Схема подключения регистров такова, что загрузка их данными осуществляется поочередно. Первый байт загружается в регистр RG2, следующий — в RG1. Изменение состояний выходов регистров производится только по нарастающему фронту сигнала CS.

На управляющие входы жидкокристаллического индикатора должно подаваться переменное напряжение синусоидальной или прямоугольной формы без постоянной составляющей. Управление постоянным напряжением не допускается, так как это приводит к возникновению электролитического эффекта, что существенно снижает срок службы индикатора.

При управлении от цифровых выходов МК переменное напряжение можно создать, подавая сигнал прямоугольной формы на общий (COM) и управляющий (1A–1G, 2A–2G и т. д.) контакты индикатора. Временные диаграммы, поясняющие управление ЖКИ с помощью МК показаны на рисунке 3.

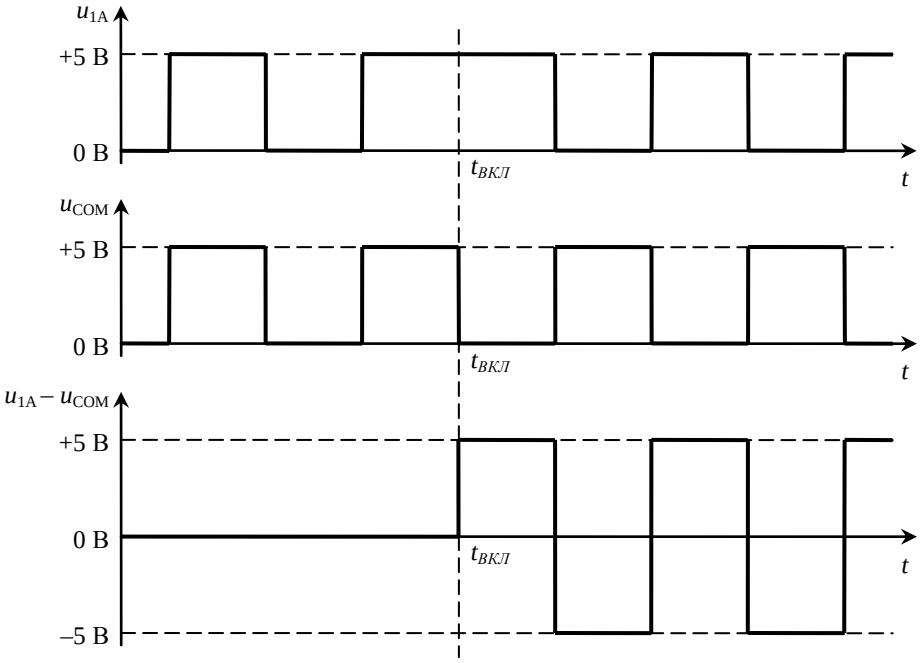

Рисунок 3 – Временные диаграммы напряжений на контактах ЖКИ

До момента времени $t_{ВКЛ}$ на контакт 1A напряжение подается синфазно с общим контактом (COM). При этом сегмент не светится, так как напряжение на 1A относительно COM равно нулю. Если на 1A и COM напряжение подается в противофазе ($t > t_{ВКЛ}$), то сегмент светится, так как

напряжение на 1А не равно нулю и периодически меняет знак по отношению к COM. Частота переменного напряжения обычно составляет 30–1000 Гц.

На макете имеется потенциометр, подключенный к аналого-цифровому преобразователю МК (см. рисунок 4). Потенциометр позволяет создавать на входе АЦП дифференциальное напряжение –2,5…+2,5 В.

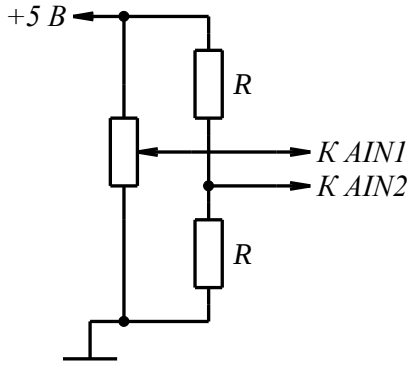

Рисунок 4

Для управления режимами работы микроконтроллера используются кнопки «Сброс» (черного цвета без фиксации), «Работа / Программирование» (белого цвета с фиксацией). Утопленному положению кнопки соответствует режим «Программирование». Каждый раз после переключения кнопки «Работа / Программирование» необходимо нажимать кнопку «Сброс».

Подготовка макета к работе

1. Поместить лабораторный макет на стол рядом с персональным компьютером.

2. Убедиться, что штырьковый разъем интерфейсной платы надежно подключен к основной плате макет. **Студентам запрещается самостоятельно подключать разъем!**

3. Убедиться, что питание персонального компьютера отключено.

4. Подключить интерфейсный кабель лабораторного макета к свободному разъему RS-232 на задней стенке системного блока персонального компьютера.

5. Включить питание персонального компьютера.

6. При выполнении этапов, связанных с проверкой работоспособности программы, подключить источник питания макета к сети 220 В/50 Гц. Включать питание макета перед началом работы не рекомендуется.

2 Архитектура микроконтроллеров семейства 8051

Схемы программистской модели и организации памяти МК семейства 8051 изображены на рисунке 5.

Программистская модель (на рисунке в центре) включает аккумулятор A (Acc); расширитель аккумулятора B; восемь регистров общего назначения R0–R7, слово состояния процессора PSW, указатель данных DPTR, указатель стека SP.

Архитектуру 8051 можно назвать ориентированной на аккумулятор. Большинство инструкций используют его в качестве одного из операндов,

поскольку не могут воздействовать непосредственно на регистры общего назначения и ячейки памяти. Это значит, что прежде чем произвести ту или иную операцию часто приходится сперва записать данные в аккумулятор, обработать, а затем выгрузить результат в память данных.

Расширитель аккумулятора B используется командами умножения и деления для хранения одного из операндов и части результата.

Регистры общего назначения предназначены для сокращения кода программы за счет уменьшения разрядности поля адреса. Адрес регистра занимает 3 бита, в то время как адрес ячейки памяти — восемь. Регистры R0 и R1, кроме того, используются для хранения исполнительного адреса при косвенной адресации ячеек памяти данных.

Регистр-указатель DPTR — 16-разрядный. Он также используется для хранения исполнительного адреса. Высокая разрядность необходима при обращении к внешней памяти данных или памяти программ.

Особенностью 8051 можно считать удобные команды пересылки. Одной командой возможна пересылка данных из любого источника в любой приемник: аккумулятор, регистр общего назначения, ячейка памяти данных, адресуемая прямо или косвенно, или константа.

В то же время команды арифметических и логических операций требуют размещения одного из аргументов в аккумуляторе. Приемником результата арифметических операций также является аккумулятор.

Рассмотрим организацию памяти данных микроконтроллера 8051. Младшие 128 адресов (00h–7Fh) соответствуют памяти данных (см. рисунок 5 слева). Регистры общего назначения (R0–R7) отображаются на один из банков памяти, расположенных по адресам 00h–07h, 08h–0Fh, 10h–17h, 18h–1Fh. Выбор банка памяти осуществляется через биты RS1:RS0 слова состояния процессора PSW.

Регистрам специальных функций (рисунок 5 справа) соответствуют старшие 128 адресов (80h–FFh). Обращение к ним возможно лишь с применением прямого метода адресации. Аккумулятор A, расширитель аккумулятора B, регистр DPTR, слово состояния процессора PSW и указатель стека SP включены в пространство адресов регистров специальных функций.

В микроконтроллерах 8052 встроенная память данных расширена до 256 байт. Адреса верхних 128 байт совпадают с адресами регистров специальных функций. Обращение к регистрам специальных функций производится прямым методом, а к расширенной памяти — косвенным.

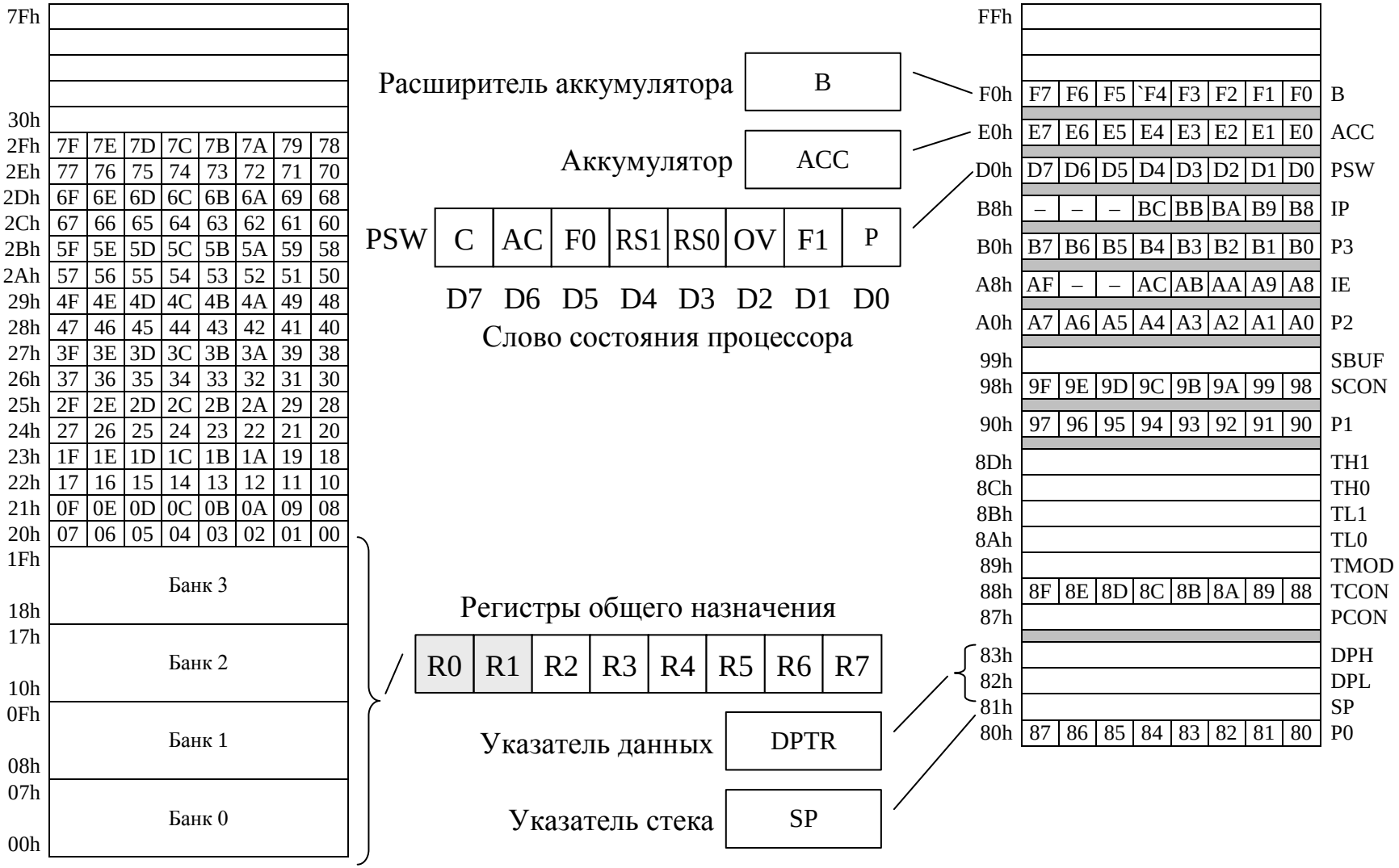

Рисунок 5 – Архитектура микроконтроллера семейства 8051

Память данных микроконтроллеров 8051 может быть расширена при помощи внешних микросхем. Обращение к внешней памяти данных осуществляется при помощи специальной команды **movx**. Адресация внешней памяти производится только косвенно через регистр DPTR. С учетом разрядности этого регистра можно адресовать 64 кбайта памяти данных. Существуют микроконтроллеры, в которых «внешняя» память данных размещена на кристалле микроконтроллера, поэтому является фактически внутренней, но для программиста обращение к ней производится, как к внешней.

К некоторым ячейкам памяти данных можно обращаться в битовом режиме. Для этого предназначены специальные команды. Биты имеют «сквозную» нумерацию от младшего бита ячейки памяти 20h (бит 00h) до старшего бита ячейки 2Fh (бит 7Fh). Многие из регистров специальных функций также могут быть изменены в битовом режиме (адреса битов 80h – FFh).

Слово состояния процессора содержит шесть флагов и два управляющих бита. Перечислим их и укажем назначение.

C — флаг переноса, устанавливается в единицу при переносе из старшего (седьмого) разряда АЛУ;

AC — дополнительный флаг переноса, устанавливается в единицу при переносе из третьего разряда АЛУ;

OV — флаг переполнения, устанавливается в единицу при переносе из шестого разряда АЛУ; означает потерю знака;

F0, F1 — определяются пользователем;

RS0, RS1 — биты выбора банка для регистров общего назначения.

P — бит четности; устанавливается в единицу, если аккумулятор содержит нечетное число единиц.

Приведенное описание передает лишь основные назначения флагов. Подробное описание влияния команд на слово состояния процессора приведено в книге [1]. Перечень команд, влияющих на регистр PSW, приведен в таблице 1.

Таблица 1 – Команды, влияющие на флаги состояния процессора

Команда	Флаги	Команда	Флаги	Команда	Флаги
ADD	C, OV, AC	DA	C	CPL C	C=C′
ADDC	C, OV, AC	RRC	C	ANL C	C
SUBB	C, OV, AC	RLC	C	ORL C	C
MUL	C=0, OV	SETB C	C=1	MOV C	C
DIV	C=0, OV	CLR C	C=0	CJNE	C

Таблица 2 – Система команд микроконтроллеров семейства 8051

Мнемоническое обозначение	Описание		Методы адресации				Время
			Прям.	Косв.	Рег.	Неп.	
Группа команд пересылки							
MOV A, *Источник*	A = *Источник*		✓	✓	✓	✓	1
MOV *Приемник*, A	*Приемник* = A		✓	✓	✓		1
MOV *Приемник*, *Источник*	*Приемник* = *Источник*		✓	✓	✓	✓	2
MOV DPTR, #*const16*	DPTR = *const16*					✓	2
PUSH *Источник*	SP = SP + 1, [SP] = *Приемник*		✓				2
POP *Приемник*	*Приемник* = [SP], SP = SP – 1		✓				2
XCH A, *Байт*	A ⇔ *Байт*		✓	✓	✓		1
XCHD A, *Байт*	A<3:0> ⇔ *Байт*<3:0>			✓			1
MOVX A, *Источник*	A = *Источник*	Обращение к внешней памяти данных		✓			2
MOVX *Приемник*, A	*Приемник* = A			✓			2
MOVX A, @DPTR	A = [DPTR]		Косвенная через указатель				2
MOVX @DPTR, A	[DPTR] = A		Косвенная через указатель				2
MOVC A, @A+DPTR	A = [A + DPTR]	Чтение из памяти программ	Косвенная DPTR и аккум.				2
MOVC A, @A+PC	A = [A + PC]		Косв. через PC и аккум.				2
Группа команд арифметических операций							
ADD A, *Байт*	A = A + *Байт*		✓	✓	✓	✓	1
ADDC A, *Байт*	A = A + *Байт* + C		✓	✓	✓	✓	1
SUBB A, *Байт*	A = A – *Байт* – C		✓	✓	✓	✓	1
INC A	A = A + 1		Только аккумулятор				1
INC *Байт*	*Байт* = *Байт* + 1		✓	✓	✓		1
INC DPTR	DPTR = DPTR + 1		Только указатель				2
DEC A	A = A – 1		Только аккумулятор				1

Таблица 2 – Система команд микроконтроллеров семейства 8051 (продолжение)

Мнемоническое обозначение	Описание	Методы адресации				Время
		Прям.	Косв.	Рег.	Неп.	
Группа команд арифметических операций (продолжение)						
DEC *Байт*	*Байт* = *Байт* – 1	✓	✓	✓		1
MUL AB	B:A = B × A	Только аккумулятор				4
DIV AB	A = int(A/B)[1]; B = mod(A/B)[2]	Только аккумулятор				4
DA A	Десятичная коррекция A	Только аккумулятор				1
Группа команд логической обработки						
ANL A, *Байт*	A = A ∧ *Байт*	✓	✓	✓	✓	1
ANL *Байт*, A	*Байт* = *Байт* ∧ A	✓	✓	✓	✓	1
ANL *Байт*, #*const8*	*Байт* = *Байт* ∧ *const8*	✓				
ORL A, *Байт*	A = A ∨ *Байт*	✓	✓	✓	✓	1
ORL *Байт*, A	*Байт* = *Байт* ∨ A	✓	✓	✓	✓	1
ORL *Байт*, #*const8*	*Байт* = *Байт* ∨ *const8*	✓				
XRL A, *Байт*	A = A ⊕ *Байт*	✓	✓	✓	✓	1
XRL *Байт*, A	*Байт* = *Байт* ⊕ A	✓	✓	✓	✓	1
XRL *Байт*, #*const8*	*Байт* = *Байт* ⊕ *const8*	✓				
CLR A	A = 0	Только аккумулятор				1
CPL A	A = A′	Только аккумулятор				1
RL A	Сдвиг A влево	Только аккумулятор				1
RLC A	Сдвиг A влево через флаг переноса	Только аккумулятор				1
RR A	Сдвиг A вправо	Только аккумулятор				1
RRC A	Сдвиг A вправо через флаг переноса	Только аккумулятор				1
SWAP A	A<3:0> ⇔ A<7:4>	Только аккумулятор				1

Таблица 2 – Система команд микроконтроллеров семейства 8051 (продолжение)

Мнемоническое обозначение	Описание	Методы адресации				Время
		Прям.	Косв.	Рег.	Неп.	
Группа команд операций с битами						
ANL C, *Бит*	C = C ∧ *Бит*	✓				2
ANL C, /*Бит*	C = C ∧ *Бит′*	✓				2
ORL C, *Бит*	C = C ∨ *Бит*	✓				2
ORL C, /*Бит*	C = C ∨ *Бит′*	✓				2
MOV C, *Бит*	C = *Бит*	✓				1
MOV *Бит*, C	*Бит* = C	✓				1
CLR C	C = 0	✓				1
CLR *Бит*	*Бит* = 0	✓				1
CPL C	C = C′	✓				1
CPL *Бит*	*Бит* = *Бит′*	✓				1
SETB C	C = 1	✓				1
SETB *Бит*	*Бит* = 1	✓				1
Группа команд безусловных переходов						
JMP *Адрес*	PC = *Адрес*				✓	2
JMP @A+DPTR	PC = [A + DPTR]		✓			2
CALL *Адрес*	[SP] = PC, PC = *Адрес*, SP = SP + 1				✓	2
RET	PC = [SP], SP = SP − 1					2
RETI	Возврат из прерывания					2
Группа команд условных переходов						
JZ *Адрес*	PC = *Адрес*, если A = 0	Только аккумулятор				2
JNZ *Адрес*	PC = *Адрес*, если A ≠ 0	Только аккумулятор				2
DJNZ *Байт*, *Адрес*	*Байт* = *Байт*−1; PC = *Адрес*, если *Байт* ≠ 0	✓		✓		2

Таблица 2 – Система команд микроконтроллеров семейства 8051 (продолжение)

Мнемоническое обозначение	Описание	Методы адресации				Время
		Прям.	Косв.	Рег.	Неп.	
Группа команд условных переходов (продолжение)						
`CJNE A,` *Байт*`,` *Адрес*	`PC =` *Адрес*, если `A ≠` *Байт*	✓			✓	2
`CJNE` *Байт*`, #`*const8*`,` *Адрес*	`PC =` *Адрес*, если *Байт* ≠ *const8*		✓	✓		2
`JC` *Адрес*	`PC =` *Адрес*, если `C = 1`	✓				2
`JNC` *Адрес*	`PC =` *Адрес*, если `C = 0`	✓				2
`JB` *Бит*`,` *Адрес*	`PC =` *Адрес*, если *Бит* `= 1`	✓				2
`JNB` *Бит*`,` *Адрес*	`PC =` *Адрес*, если *Бит* `= 0`	✓				2
`JBC` *Бит*`,` *Адрес*	`PC =` *Адрес* и *Бит* `= 0`, если *Бит* `= 1`	✓				2

Примеры команд

```
        jmp   Label1    ; Записать в счетчик команд адрес, обозначенный меткой
        …
Label1: clr   23h       ; Сбросить бит с адресом 23h
        mov   A, 7Fh    ; Прямая адресация. Записать в аккумулятор содержимое
                        ; ячейки памяти с шестнадцатеричным адресом 7Fh
        add   A, R3     ; Регистровая адресация. Записать в аккумулятор
                        ; содержимое регистра общего назначения R3
        anl   A, @R0    ; Косвенная адресация. Выполнить логическое умножение
                        ; аккумулятора и ячейки памяти, адрес которой хранится в R0
        subb  A, #-12   ; Непосредственная адресация. Вычесть из аккумулятора
                        ; десятичное число -12
```

[1] `int(A/B)` — целая часть от деления `A/B`

[2] `mod(A/B)` — остаток от деления `A/B`

В стеке сохраняются адреса возврата при вызове подпрограмм и обработке прерываний. В микроконтроллерах 8051 стек организуется в ячейках памяти данных. Указателем вершины стека служит регистр SP. Указатель стека получает приращение автоматически перед записью в стек. По умолчанию он ссылается на ячейку памяти с адресом 07h, то есть стек начинается с адреса 08h.

Остальные регистры специальных функций предназначены для управления периферийными устройствами, встроенными в МК. В базовой версии — это четыре порта ввода-вывода, два 16-разрядных таймера, последовательный приемопередатчик. В версии 8052 к ним добавлен третий таймер и массив программируемых таймеров. Описание этих регистров выходит за рамки методических указаний. Для изучения обращайтесь к [1].

3 Лабораторная работа №1. Изучение основных приемов работы с интегрированной средой Keil µVision 3

Все операции, связанные с программированием микроконтроллеров выполняются с помощью популярной среды разработки Keil µVision 3 Среда включает текстовый редактор, ассемблер, СИ-компилятор и отладчик. В лабораторных работах используется демонстрационная версия программного обеспечения. Единственным ограничением является объем исполнимого кода, формируемого ассемблером — он не может превышать 2 кбайт.

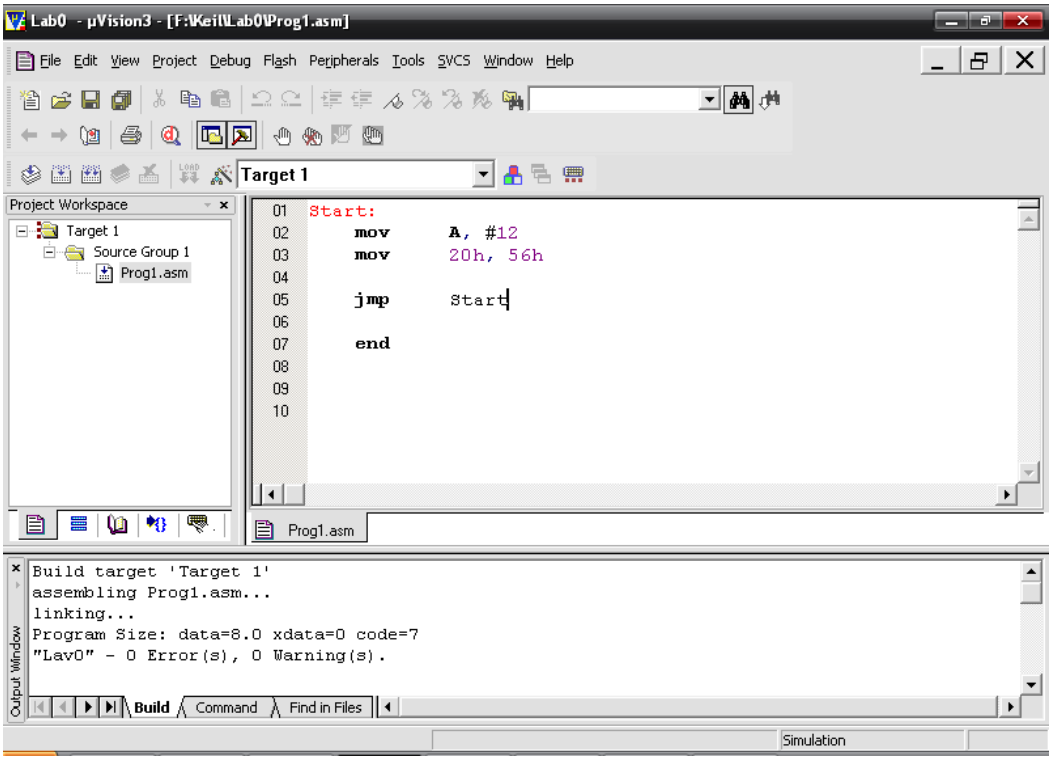

Рисунок 6 – Основное окно Keil µVision 3 в режиме программирования

Создание проекта

В современных средах разработки все файлы, относящиеся к создаваемой программе, объединяются в проекты. Фактически, проект — это файл, хранящий информацию о размещении на диске всех компонентов программы, параметры настройки среды разработки и т. п. Все файлы проекта должны быть размещены в одном каталоге.

Перед началом работы требуется создать рабочий каталог на жестком диске, в котором будут размещены файл с исходным текстом программы, объектный файл и другие файлы, создаваемые средой разработки.

Для создания нового проекта необходимо:

а) Воспользоваться пунктом меню **Project ⇒ New µVision Project**. При этом откроется стандартное окно сохранения файла.

б) Перейти в папку проекта и сохранить проект.

в) Выбрать тип имитируемого контроллера (см. рисунок 8). В нашем случае — **Analog Devices ⇒ ADuC816**. На вопрос «**Copy Analog Devices Startup Code to Project Folder and Add File to Project?**» следует ответить «**Нет**».

Создание файла программы

Файл программы представляет собой обычный текстовый файл с расширением *.ASM. Для создания файла необходимо выполнить следующие операции:

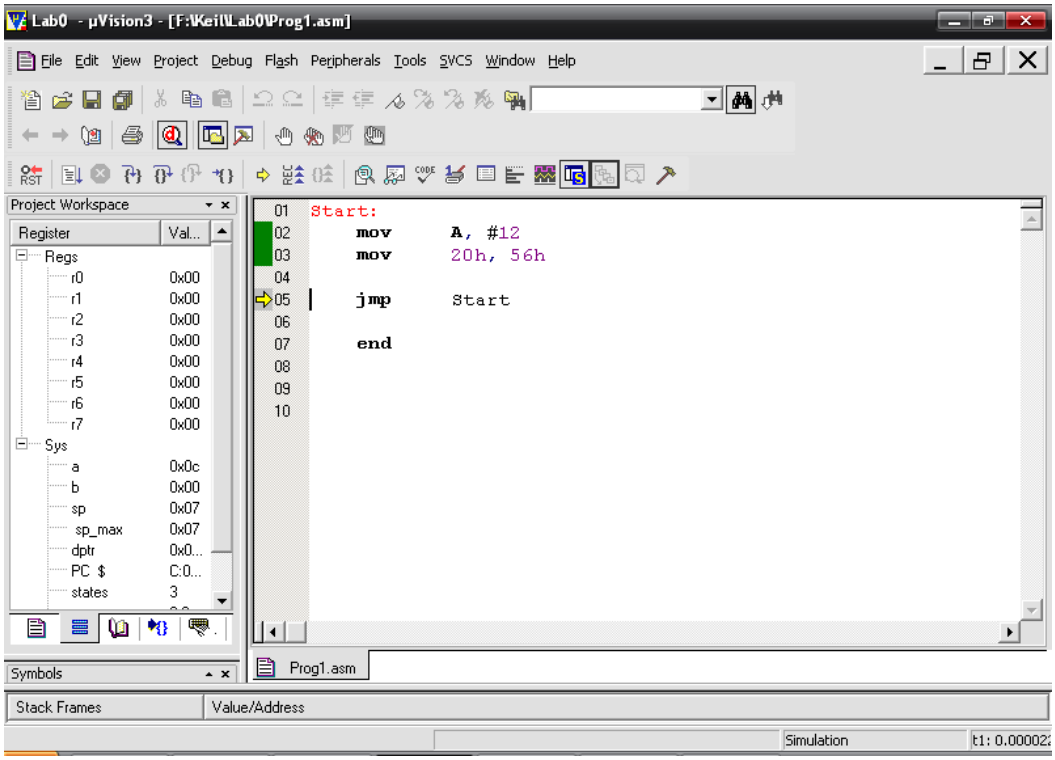

Рисунок 7 – Основное окно Keil µVision 3 в режиме отладки

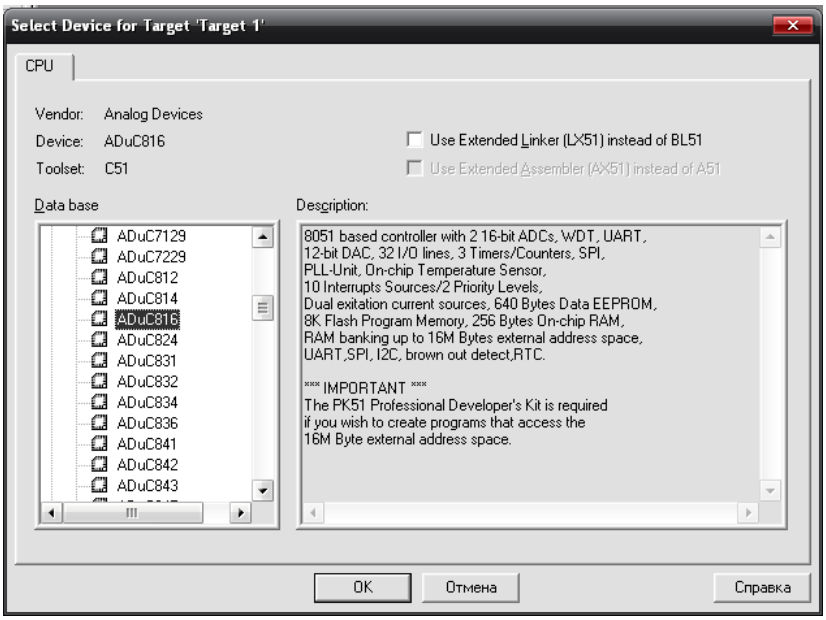

Рисунок 8 – Окно выбора микроконтроллера

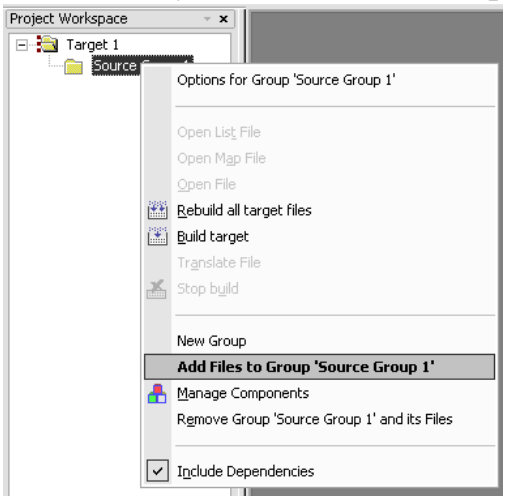

Рисунок 9

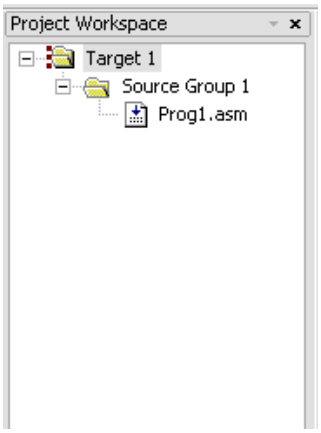

Рисунок 10

а) Создать новый файл командой меню **File** ⇒ **New.**

б) Сохранить пока еще пустой файл командой **File** ⇒ **Save**. Сохранять файл необходимо в папку проекта.

в) Щелчком правой кнопки мыши по пункту **Source Group 1** окна проекта (см. рисунок 9) открыть его контекстное меню.

г) Воспользоваться пунктом **Add Files to Group…** и выбрать файл программы (*.ASM) в открывшемся диалоговом окне. Имя файла программы должно появиться в окне проекта (см. рисунок 10).

Набор текста программы

Текст программы на ассемблере принято оформлять в четыре колонки. Каждая строка состоит из четырех полей: меток, операторов, опе-

рандов, комментариев. Поля отделяются друг от друга символом «табуляция» (кнопка TAB).

В данной лабораторной работе предлагается использовать простейшую программу генератора прямоугольного сигнала частотой 1 кГц.

Листинг программы приведен ниже.

```
        jmp   Start              ; Переход на точку входа
        org   0x0B               ; Вектор прерывания от
                                 ; таймера 0
        jmp   T0Int              ; Переход на процедуру
                                 ; обработки прерывания
Start:  mov   TMOD, #00010001b   ; Настройка таймера
        setb  TCON.4             ; Включение таймера
        mov   IE, #10000010b     ; Разрешение прерывания
                                 ; от таймера
        jmp   $                  ; Вечный цикл

T0Int:  push  PSW                ; Сохр. состояние (PSW)
        mov   TL0, #0xC5         ; Загрузка таймера
        mov   TH0, #0xFA         ; начальным значением
        xrl   P0, #00000001b     ; Инверсия вывода P0.0
        pop   PSW                ; Восстановить состояние

        reti                     ; Возврат из процедуры
                                 ; обработки прерывания
        end
```

Ассемблирование программы

На данном этапе будет создан машинный код, готовый к записи в память программ микроконтроллера. Ассемблирование выполняется нажатием кнопки F7. При этом формируется отчет об ассемблировании, который можно просмотреть щелчком по вкладке **Output Window** в нижней части окна программы. В отчете прежде всего следует обратить внимание на число ошибок (**Errors**) и предупреждений (**Warnings**) (см. рисунок 6).

При наличии ошибок их необходимо исправить. Строка отчета, информирующая об ошибке, содержит имя файла программы, номера́ ошибочных строк, коды и описания ошибок. Двойным щелчком мыши по сообщению об ошибке можно быстро переместиться на строку программы, в которой эта ошибка содержится.

Отладка программы

На этапе отладки моделируется работа программы. Среда разработки полностью имитирует поведение реального контролера. Можно наблюдать за пошаговым выполнением программы, за состоянием регистров и течением модельного времени. Переключение между режимом редактирования и отладки производится комбинацией клавиш Ctrl+F5.

Назначение кнопок наиболее часто используемых в режиме отладки приведены в таблице 3. Способы вызова диалоговых окон, необходимых для слежения за состоянием микроконтроллера и его периферийных устройств приведены в таблице 4.

Таблица 3 – Основные команды отладчика Keil μVision 3

Запуск программы	F5
Остановка программы	Debug ⇒ Stop Running
Системный сброс	Peripherals ⇒ Reset CPU
Пошаговое выполнение программы	F10
Пошаговое выполнение с заходом в подпрограммы	F11
Точка останова	Insert/Remove Breakpoint (в контекстном меню)

Таблица 4 – Основные диалоговые окна отладчика Keil μVision 3

Ячейки внутренней памяти данных	View ⇒ Memory Window
Содержимое памяти программ	View ⇒ Disassembly Window
Состояние объявленных переменных в памяти данных	View ⇒ Watch & Call Stack Window
Виртуальный терминал асинхронного приемопередатчика	View ⇒ Serial Window
Состояние встроенных периферийных устройств	Peripherals ⇒ …
Хронометраж	View ⇒ Performance Analyzer Window

В ходе отладки рекомендуется придерживаться следующего порядка действий.

а) В пошаговом режиме (F10, F11) проследить за переходом на основную программу (метка Start), выполнением команд и «зацикливанием» программы. Указатель в виде желтой стрелки слева отмечает команду, на которую ссылается счетчик команд микроконтроллера, то есть команду, выполнение которой произойдет на следующем этапе.

б) Убедиться, что таймер 0 запускается и инкрементируется в каждом машинном цикле (**Peripherals** ⇒ **Timer** ⇒ **Timer 0**).

в) Уставить точку останова в начале процедуры обработки прерываний от таймера (после метки T0Int). Убедиться, что процедура обработки прерываний запускается.

г) Убедиться в том, что содержимое РСФ PSW заносится в стек (по команде **`push`**), что указатель стека SP автоматически инкрементируется. Состояние указателя стека отображается в левой колонке (см. рисунок 7). Содержимое стека просматривается через окно памяти данных (**View** ⇒ **Memory Window**). Для просмотра содержимого памяти данных следует в поле **Address** ввести **`D:0`**. Поскольку инициализация стека в программе не производится, он располагается с адреса 08h памяти данных.

д) Пронаблюдать присваивание начальных значений регистрам таймера (**Peripherals** ⇒ **Timer** ⇒ **Timer 0**).

е) Проследить за изменением портовой линии P0.0 (**Peripherals** ⇒ **I/O-Ports** ⇒ **Port 0**).

ж) Проконтролировать корректный возврат из подпрограммы обработки прерывания. Проследить за изменением указателя стека.

з) Измерить период вызова процедуры обработки прерываний; сравнить с желаемым временем (**View** ⇒ **Performance Analyzer Window**).

Запись объектного кода в память программ микроконтроллера

Прежде чем приступать к загрузке кода программы во внутреннюю память программ микроконтроллера, необходимо выполнить настройку среды.

а) Вызвать диалоговое окно **Project** ⇒ **Options for Target '…'**.

б) На вкладке **Utilities** в поле **Use Target Driver for Flash Programming** установить значение **ADI Monitor Driver**.

в) В диалоге, вызываемом кнопкой **Settings**, выбрать номер порта RS-232, к которому подключен лабораторный макет.

Перед загрузкой программы необходимо включить питание лабораторного макета. Затем при помощи кнопки белого цвета перевести МК в режим программирования (кнопка утоплена), нажать кнопку «Сброс» (черного цвета).

Загрузка программы осуществляется при помощи пункта меню **Flash** ⇒ **Download** или соответствующей кнопки на панели инструментов. В случае успешной загрузки автоматически произойдет запуск программы. В случае ошибки появится сообщение **TARGET SYSTEM DOES NOT**

RESPOND. Можно либо отменить операцию кнопкой **Stop Debugging**, либо устранить неисправность и повторить загрузку кнопкой **Try Again**.

Прежде чем загружать новую версию программы потребуется снова нажать кнопку «Сброс».

4 Лабораторная работа №2. Индикация статического изображения

Цель работы

Разработать программу, которая выполнит подсветку всех сегментов жидкокристаллического индикатора.

Описание программы

Для подсветки всех сегментов необходимо изменять уровни на каждом управляющем входе индикатора с выбранной частотой, причем в противофазе с уровнем общего входа (контакт COM).

Формирование временных интервалов предлагается осуществлять с помощью таймера 0 и прерывания по переполнению таймера.

Процедура обработки прерывания должна содержать команды изменения состояния портовых линий МК, непосредственно соединенных с ЖКИ, а также команды передачи байтов управления ЖКИ во внешние регистры.

Изменение уровней на противоположные можно выполнять командой сложения по модулю 2.

Так как МК не имеет обратной связи с внешними регистрами, и считать их состояние невозможно, необходимо хранить в памяти данных МК состояние внешних регистров. Назовем эти ячейки «зеркалом» регистров.

Дальнейшее описание программы будет сопровождаться ссылками на страницы технического описания МК ADuC816.

1. Перед текстом основной программы необходимо поместить директивы замены описания регистров специальных функций базового МК 8051 на описание РСФ МК ADuC816

```
$NOMOD51
$include (C:\Keil\C51\INC\ADI\ADUC816.h)
```

2. Объявить две ячейки памяти — «зеркала» регистров. Их адреса не должны совпадать с адресами регистров общего назначения (см. рисунок 5). Использовать директиву

Имя **data** *Адрес*

3. Программа должна начинаться с команды безусловного перехода (**jmp**) на точку входа. Например, обозначим ее меткой **Start**.

4. По соответствующему вектору разместить команду безусловной передачи управления на процедуру обработки прерывания от таймера 0. Саму процедуру обработки прерываний расположить в конце программы (стр. 67, таблица XXXIV).

```
org     Адрес
jmp     Метка
```

5. Основную программу начать с метки, обозначающей точку входа, например

```
Start:
```

6. Провести настройку умножителя внутренней тактовой частоты на основе ФАПЧ (регистр PLLCON) (стр. 42). Частоту можно выбрать произвольно. Для этого записать в регистр PLLCON числовую константу. Ниже приведены равнозначные примеры.

```
mov     PLLCON, #00000011b
mov     PLLCON, #03h
mov     PLLCON, #0x03
mov     PLLCON, #3
```

7. Инициализировать указатель стека. Записать в РСФ SP адрес вершины стека. SP должен ссылаться на область старших адресов памяти, например 70h, чтобы не перекрыть хранимые в памяти данные (рисунок 5). Применить команду **mov**.

8. Провести инициализацию всех четырех портов (P0…P3). Предлагается во все линии, к которым подключен индикатор, записать нули; в остальные — единицы.

9. Записать нулевые значения во все биты «зеркал» регистров, которые соответствуют управляющим входам индикатора и единицу в разряд, соответствующий общему контакту индикатора (COM).

10. Провести настройку интерфейса SPI через регистр SPICON (стр. 49). Включить SPI в режиме ведущего на максимальной скорости. Биты выбора полярности и фазы синхросигнала оставить сброшенными.

11. Настроить таймер 0 через РСФ TMOD и TCON (стр. 54–55). Включить таймер 0 в режиме 16-разрядного таймера без стробирования.

12. Разрешить прерывание от таймера 0 через регистр IE (стр. 66). Здесь же установить бит общего разрешения прерываний.

13. Зациклить программу командой

```
jmp     $
```

14. Процедура обработки прерываний начинается с метки.

15. Процедура обработки прерываний должна содержать блок команд перезагрузки таймера. Здесь необходимо присвоить таймеру начальное значение через регистры TH0 и TL0. Начальное значение таймера может быть рассчитано по формуле:

$$N = 2^{16} - \frac{T/2}{T_0} + N_0 = 65536 - \frac{T/2}{12/F_0} + N_0,$$

где T — период формируемого сигнала; T_0 — длительность машинного цикла (12 тактов); F_0 — тактовая частота; N_0 — поправка, учитывающая затраты времени на выполнение переходов и команд процедуры обработки прерываний (подбирается в ходе отладки программы). Число N необходимо разбить на старший и младший байты, переведя предварительно в шестнадцатеричную систему счисления, и загрузить в соответствующие регистры TH0 и TL0.

На время загрузки таймера его рекомендуется остановить при помощи соответствующего бита регистра TCON (стр. 55).

16. Записать команды инверсии (**`xrl`**) всех битов портов и «зеркал» внешних регистров, соответствующих управляющим входам ЖКИ.

17. Загрузить данные из «зеркал» во внешние регистры. SPI-передатчик начинает работать после записи байта данных в регистр SPI-DAT. Необходимо соблюдать последовательность загрузки регистров: сначала RG2, затем RG1 (на принципиальной схеме D3, D2). Следует дожидаться окончания загрузки каждого регистра, циклически опрашивая состояние бита готовности в регистре SPICON (стр. 49).

18. После загрузки обоих регистров сформировать строб CS для изменения состояния выходов регистров. Для сброса и установки бита можно использовать команды **`clr`** и **`setb`**.

19. Подпрограмма обработки прерывания завершается командой

`reti`

20. Файл завершается директивой

`end`

Отладка

В ходе отладки рекомендуется придерживаться следующего порядка действий:

а) проверить присваиваются ли регистрам P0–P3 верные значения (**Peripherals ⇒ I/O-Ports ⇒ Port X**);

б) убедиться, что таймер 0 запускается и инкрементируется в каждом машинном цикле (**Peripherals ⇒ Timer ⇒ Timer 0**);

в) пронаблюдать «зацикливание» программы;

г) уставить точку останова в начале процедуры обработки прерываний от таймера; убедиться, что процедура обработки прерываний запускается и вместе с тем обеспечивается корректный возврат в вечный цикл.

д) проконтролировать начальное значение, присваиваемое таймеру;

е) убедиться, что регистры портов и «зеркала» регистров инвертируются, причем всегда бит COM не равен остальным управляющим битам (**View ⇒ Watch & Call Stack Windows ⇒ Watch #1**);

ж) измерить период вызова процедуры обработки прерываний; сравнить с желаемым временем (**View ⇒ Performance Analyzer Window**).

Тестирование

Для окончательной проверки работоспособности программы ее необходимо испытать на лабораторном макете. Порядок действий для загрузки программы во внутреннюю память микроконтроллера и ее запуска подробно описан в лабораторной работе №1

5 Лабораторная работа №3. Формирование изображения с возможностью изменения

Цель работы

Разработать программу для индикации четырех десятичных чисел. Программа должна обеспечить корректное формирование изображения на индикаторе.

Описание программы

За основу берется программа, составленная в ходе работы №2.

В данной программе вводится возможность изменять изображение на индикаторе. Предлагается следующий алгоритм. Используется группа ячеек памяти Port_1–Port_3, Reg_1–Reg_2. В эти ячейки заносятся единицы в те разряды, которым соответствуют подсвеченные сегменты. Значения из этих ячеек будут переноситься в порты P1–P3 и «зеркала» регистров только в тот момент, когда общий контакт COM = 0. В самом деле, в эти моменты времени высокие уровни на контактах индикатора будут соответствовать свечению сегментов.

Для преобразования четырехразрядного десятичного числа в код управления семисегментным индикатором с учетом разводки его контактов предлагается воспользоваться готовой подпрограммой Decoder.

Подпрограмма использует таблицу символов аналогичную популярным драйверам индикации MAX7219 (таблица 5).

Подпрограмма содержится в файле Indic, который необходимо подключить к программе. В файле Indic объявлены несколько ячеек памяти. Комментарии см. в таблице 6.

Входными параметрами при вызове подпрограммы Decoder являются ячейки памяти Num_1–Num_4, выходными — Port_1–Port_3, Reg_1–Reg_2, где Num_1–Num_4 содержат двоично-десятичные коды индицируемых цифр.

Таблица 5 – Коды управления жидкокристаллическим индикатором

Код	Изображение	Код	Изображение
0 (0x00)		8 (0x08)	
1 (0x01)		9 (0x09)	
2 (0x02)		10 (0x0A)	
3 (0x03)		11 (0x0B)	
4 (0x04)		12 (0x0C)	
5 (0x05)		13 (0x0D)	
6 (0x06)		14 (0x0E)	
7 (0x07)		15 (0x0F)	

Таблица 6 – Описание ячеек памяти, объявленных в файле Indic

Имя	Шестнадцатеричный адрес	Назначение
Reg_1	23h	Содержат коды управления индикатором с учетом разводки. Единицы в этих ячейках памяти соответствуют подсвеченным сегментам индикатора, нули — погашенным.
Reg_2	24h	
Port_1	25h	
Port_2	26h	
Port_3	27h	
Num_1	28h	Ячейки содержат десятичные значения разрядов индикатора.
Num_2	29h	
Num_3	2Ah	
Num_4	2Bh	
Seg1	2Ch	Ячейки содержат коды управления сегментами (1 — подсвечен, 0 — погашен) без учета разводки в формате: A-B-C-D-E-F-G-dp.
Seg2	2Dh	
Seg3	2Eh	
Seg4	2Fh	

Подготовка к работе

Скопировать файл Indic в рабочий каталог, где размещена программа, составленная в работе №2.

Порядок работы

1. Команды инициализации портов и «зеркал» регистров заменить командами сброса в нуль ячеек Port_1–Port_3, Reg_1–Reg_2.

2. Добавить команды присваивания некоторых значений ячейкам памяти, в которых хранятся индицируемые цифры. Например, Num_1 присвоить значение «1», а Num_4 — значение «4».

3. Вызвать подпрограмму Decoder командой **`call`**.

4. Внести изменения в подпрограмму обработки прерывания от таймера. Непосредственно после команд инициализации таймера организовать проверку условия: «COM = 0» (команда **`jb`** или **`jnb`**). Если условие истинно, выполнить пересылку содержимого ячеек Port_1–Port_3 в РСФ P1–P3, а ячеек Reg_1–Reg_2 в соответствующие «зеркала» регистров.

5. Инвертировать разряд порта, соответствующий двоеточию (контакт COL) командой **`cpl`**.

6. Выти из процедуры обработки прерывания командой **`reti`**.

7. В конце программы (непосредственно перед директивой **`end`**) разместить директиву подключения файла Indic:

```
$include(indic)
```

Отладка

В данной работе в ходе отладки рекомендуется проверить выполнение ветвления по условию «COM = 0».

6 Лабораторная работа №4. Разработка программы счетчика событий

Цель работы

Разработать программу для подсчета и индикации числа нажатий на кнопку.

Описание программы

За основу берется программа, составленная в ходе работы №3.

На основной плате лабораторного макета имеется кнопка, подключенная к портовой линии P3.2 (INT0). Эта линия может использоваться для запроса внешнего прерывания. В зависимости от настроек прерывающим событием может являться либо отрицательный перепад напряжения, либо уровень логического нуля на линии. Поскольку кнопки, установленные на

макете практически не дают дребезга, принимать меры по подавлению дребезга не требуется.

Процедура обработки прерываний по отрицательному перепаду уровня на линии P3.2 (INT0) должна содержать инструкции для подсчета числа нажатий, перевода зарегистрированного числа в десятичную систему счисления и обновления разрядов индикатора. Хранить число событий рекомендуется в РОН R2. Регистры R0 и R1 использовать нельзя, поскольку они используются подпрограммой Decoder.

Порядок работы

1. К программе, разработанной в работе №3, добавить команду передачи управления на процедуру обработки прерывания от внешнего прерывания INT0 (стр. 67). Адрес команды задается директивой **org** соответственно таблице векторов прерываний.

2. Установить режим прерывания по отрицательному перепаду сигнала на контакте P3.2 через РСФ TCON (стр. 55).

3. Разрешить прерывание от внешнего сигнала (INT0) при помощи регистр IE (стр. 66).

4. Перед вечным циклом следует размесить инструкции обнуления ячеек памяти Num_1–Num_4 и вызова подпрограммы Decoder.

5. Далее обнулить ячейку памяти, выбранную для хранения числа нажатий кнопки. Выше было рекомендовано использовать для этого РОН R2.

6. Создать процедуру обработки прерывания от внешнего сигнала. Процедуру следует начать с инкремента регистра R2 (команды **inc**).

Здесь же необходимо запрограммировать перевод двоичного значения R2 в десятичный код. Преобразование в десятичную систему счисления производится путем целочисленного деления исходной величины на 10. При этом будем предусматривать корректный счет лишь до 99, то есть разряда сотен и тысяч будут оставаться нулевыми.

Для деления служит команда **div**. Делимое располагается в аккумуляторе ACC, делитель — в расширителе аккумулятора B. Команда возвращает результат целочисленного деления в ACC, остаток от деления в B.

Результаты необходимо разместить в ячейках Num_3, Num_4.

7. Вызвать подпрограмму Decoder.

8. Выти из процедуры обработки прерывания командой **reti**.

9. В конце программы (непосредственно перед директивой **end**) разместить директиву подключения файла Indic:

```
$include(indic)
```

Отладка

Отладку рекомендуется проводить по следующей методике:

а) установить точку останова в процедуре обработки внешнего прерывания; убедиться, что прерывание вырабатывается при переходе состояния портовой линии P3.2 из единицы в ноль. (**Peripherals ⇒ I/O-Ports ⇒ Port X**);

б) проконтролировать увеличение на единицу регистра R2;

в) проверить корректность перевода числа в десятичную систему счисления; при необходимости занести тестовое значение в регистр R2. Изменить РОН можно в правой колонке (Project Workspace) при помощи кнопки F2;

г) убедиться в том, что обеспечивается корректный возврат управления в основную программу.

7 Лабораторная работа №5. Разработка программы часов

Цель работы

Разработать программу для индикации минут и секунд текущего времени. Двоеточие, разделяющее минуты и секунды должно мигать с частотой 1 Гц.

Описание программы

За основу берется программа, составленная в ходе работы №4.

Микроконтроллер ADuC816 имеет встроенные часы реального времени. Значения секунд, минут и часов могут быть считаны из соответствующих регистров специальных функций: SEC, MIN, HOUR.

Необходимо при помощи прерывания от часов реального времени формировать интервалы длительностью 0,5 с. Процедура обработки прерываний должна содержать инструкции обновления разрядов индикатора и вывода мигающего двоеточия.

Порядок работы

1. К программе, разработанной в работе №4, добавить команду передачи управления на процедуру обработки прерывания от часов реального времени (стр. 67). Вызов процедуры обработки внешнего прерывания и саму процедуру исключить.

2. Добавить команды настройки часов реального времени (стр. 44). Включить временно́й счет и счетчик временного интервала (TIMECON); выбрать временную базу 1/128 секунды. Занести границу временного интервала в регистр (INTVAL). Для выбранной временной базы значение 64 соответствует интервалу 0,5 с.

3. Разрешить прерывание от часов реального времени через регистр IEIP2 (стр. 66).

4. Создать процедуру обработки прерывания от часов реального времени. Здесь необходимо обеспечить перевод двоичных значений РСФ MIN и SEC в десятичный код. Преобразуемое число не может быть больше 59, то есть разряд сотен отсутствует.

Результаты необходимо разместить в регистрах Num_1–Num_4.

5. Вызвать подпрограмму Decoder командой **`call`**.

6. Инвертировать разряд порта, соответствующий двоеточию (контакт COL) командой **`cpl`**.

7. Выти из процедуры обработки прерывания командой **`reti`**.

8. В конце программы (непосредственно перед директивой **`end`**) разместить директиву подключения файла Indic:

```
$include(indic)
```

Отладка

Отладку рекомендуется проводить по следующей методике:

а) установить точку останова в процедуре обработки прерывания от часов реального времени; убедиться, что прерывание вырабатывается с периодичностью 0,5 с (**View** ⇒ **Performance Analyzer Window**);

б) проверить корректность перевода чисел в десятичную систему счисления; при необходимости занести тестовые значения в регистры MIN и SEC через меню **Peripherals** ⇒ **Timer** ⇒ **TIC**.

в) проконтролировать периодическое изменение портовой линии, отвечающей за индикацию двоеточия;

8 Лабораторная работа №6. Разработка программы термометра

Цель работы

Разработать программу для индикации показаний встроенного датчика температуры.

Описание программы

За основу берется программа, составленная в ходе работы №3.

Для преобразования десятичных чисел в код семисегментного индикатора воспользоваться подпрограммой Decoder из файла Indic.

В микроконтроллер ADuC816 встроено два независимых сигма-дельта АЦП разрядностью 16 бит (основной и дополнительный).

К входу дополнительного АЦП может быть подключен встроенный датчик температуры.

Результат преобразования дополнительного АЦП размещается в паре РСФ ADC1H, ADC1L.

Необходимо включить прерывание от АЦП Процедура обработки прерывания должна содержать инструкции обновления разрядов индикатора.

Порядок работы

1. Добавить команду передачи управления на процедуру обработки прерывания от АЦП (стр. 67). Команду, относящуюся к прерыванию от часов реального времени исключить.

2. Добавить команды настройки дополнительного АЦП. Через регистр ADCMODE включить дополнительный АЦП в режиме циклического преобразования.

В регистре ADC1CON включить внутренний источник опорного напряжения (ИОН); подключить к входу АЦП датчик температуры; включить биполярное кодирование (стр. 19, 21).

3. Команды настройки часов реального времени исключить.

4. Разрешить прерывание от АЦП через регистр IE (стр. 66).

5. Создать процедуру обработки прерывания от АЦП. Здесь необходимо ввести поправку смещения нуля в результат АЦП. Для этого скопировать содержимое байта ADC1H в аккумулятор и прибавить к нему содержимое РСФ поправки OF1H (offset calibration 1 high) командой **`add`**.

6. Перевести откорректированные данные в десятичную систему счисления при помощи команды **`div`**. Предположим, что температура лежит в диапазоне 0–100°C. Поэтому достаточно получить два десятичных разряда и знак не учитывать. Результат разместить в регистрах Num_2, Num_3.

7. Погасить разряды 1 и 4 индикатора путем записи соответствующих значений в ячейки Num_1 и Num_4 (см. таблицу 5).

8. Вызвать подпрограмму Decoder командой **`call`**.

9. Записать соответствующий код в ячейку Port_2 так, чтобы в младшем разряде индикатора получить изображение символа «°».

10. Сбросить флаг прерывания от дополнительного АЦП в регистре ADCSTAT командой **`clr`** (стр. 18).

Отладка

Рекомендации по отладке:

а) установить точку останова в начале процедуры обработки прерывания от АЦП; убедиться, что прерывание вырабатывается с корректным возвратом;

б) проконтролировать правильность перевода чисел в десятичную систему счисления; при необходимости занести в аккумулятор тестовое число.

9 Лабораторная работа №7. Разработка программы вольтметра

Цель работы

Разработать программу для индикации значения постоянного напряжения на аналоговом входе.

Описание программы

За основу берется программа, составленная в ходе работы №3.

Основной АЦП микроконтроллера ADuC816 имеет дифференциальный вход AIN+, AIN–. Это значит, что напряжение на AIN+ может быть как положительным по отношению к AIN–, так и отрицательным. Однако напряжения на каждом из входов должны быть не меньше AGND и не больше AVDD.

В лабораторном макете на вход AIN– (AIN2) основного АЦП подается напряжение 2,5 В с делителя, а на вход AIN+ (AIN1) — напряжение, регулируемое в диапазоне 0–5 В (см. рисунок 4). Таким образом, дифференциальное напряжение изменяется от –2,5 до +2,5 В.

Порядок работы

1. В программе термометра откорректировать команды настройки АЦП. Включить основной АЦП в режиме циклического преобразования (ADCMODE). Дополнительный АЦП отключить.

Включить внутренний источник опорного напряжения (ADC0CON); подключить к входу АЦП выводы AIN1, AIN2; включить биполярное кодирование; выбрать самый широкий диапазон входного напряжения (стр. 19, 21).

2. Изменить команды считывания результата АЦП в процедуре обработки прерывания. Скопировать в аккумулятор и ввести поправку в результат основного АЦП (OF0H).

3. Погасить старший разряд индикатора (разряд знака) путем записи соответствующего кода в ячейку Num_1 (см. таблицу 5).

4. Организовать проверку знака откорректированного результата АЦП. Если результат отрицательный, то преобразовать его из дополнительного кода в прямой и записать в регистр Num_1 код, соответствующий изображению знака «–».

5. Преобразовать прямой код результата АЦП в десятичную систему счисления. Здесь исходное число может быть больше 100, поэтому команду **`div`** необходимо применять дважды.

6. Вызвать подпрограмму Decoder.

7. Сбросить флаг прерывания от основного АЦП в регистре ADCSTAT командой **`clr`** (стр. 18).

Отладка

Рекомендации по отладке программы:

а) убедиться, что программа корректно распознает знак числа. При необходимости занести в аккумулятор тестовое число.

б) проконтролировать правильность перевода трехзначных чисел в десятичную систему счисления.

10 Лабораторная работа №8. Использование асинхронного приемопередатчика

Цель работы

Разработать программу, обеспечивающую прием данных с персонального компьютера по интерфейсу RS-232 и индикацию десятичного кода принятого байта.

Описание программы

Для связи с персональным компьютером используется встроенный асинхронный приемопередатчик. Во время тестирования рекомендуется пользоваться программами Hyper Terminal, TTY или любой другой программой-терминалом, функционирующей в среде Windows.

Порядок работы

1. Добавить команду передачи управления на процедуру обработки прерывания от асинхронного приемопередатчика (стр. 67). Команду перехода на обработчик прерывания от АЦП требуется исключить.

2. Исключить настройку АЦП.

3. Настроить модуль таймера 2 при помощи регистра T2CON (стр. 58). Установить биты, отвечающие за включение таймера и использование его в качестве источника синхросигнала для приемника.

Установить модуль счета, соответствующий выбранной скорости передачи, в регистрах RCAP2H, RCAP2L. Скорость выбрать из ряда 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600 бит/с. Значение регистров RCAP2H, RCAP2L вычислить по формуле

$$RCAP2H : RCAP2L = 65536 - \frac{F_0}{32R_B},$$

где F_0 — тактовая частота микроконтроллера; R_B — битовая скорость приемопередатчика. Результат следует округлить до ближайшего целого. Примеры настроек скорости см. стр. 64 описания микроконтроллера.

4. Выполнить настройку приемника через РСФ SCON (стр. 61). Разрешить работу приемника; выбрать восьмибитный режим UART с переменной скоростью обмена; бит SM2 и бит разрешения приемника установить.

5. Включить прерывание от приемника. Отключить прерывание от АЦП.

6. В процедуре обработки прерывания от приемника скопировать принятый байт из РСФ SBUF в аккумулятор и преобразовать его в десятичную форму по аналогии с работой №4.

7. Сбросить флаг прерывания в регистре SCON (стр. 61).

Отладка

Во время отладки:

а) при помощи инструмента **Peripherals** ⇒ **Serial** убедиться, что скорость приемника установлена с погрешностью, не превышающей 5%.

б) установить точку останова в начале процедуры обработки прерывания от приемника; проверить, что прерывание вырабатывается, пользуясь имитацией терминала (**View** ⇒ **Serial Window #1**);

в) проверить перевод числа в десятичную форму.

11 Контрольные вопросы и задания

Для защиты лабораторных работ студент должен быть знаком с основами архитектуры МК 8051 по крайней мере в объеме параграфа 2 методических указаний. Следует знать организацию памяти данных; назначение регистров специальных функций, к которым приходилось обращаться в ходе работ; назначение использованных команд.

Требуется детальное понимание алгоритмов составленных программ, их реакций на те или иные события; очередности выполнения команд; назначения каждой команды.

Необходимо владение основными приемами работы с интегрированной средой разработки, поскольку в ходе защиты может быть предложено внести простейшие изменения в одну из программ и продемонстрировать результат в компьютерной имитации или на макете.

Подготовку к защите рекомендуется проводить прежде всего по источнику [1] и технического описания ADuC816, на которе неоднократно ссылались выше.

Для самостоятельного контроля знания рекомендуется следующий перечень вопросов.

1. В чем особенность Гарвардской архитектуры ЭВМ?
2. Какие периферийные устройства встроены в базовый вариант микроконтроллера семейства 8051?
3. Чем отличается микросхема ADuC816 от базовой версии?
4. Поясните назначение следующих регистров специальных функций: A, B, PSW, SP, DPTR.
5. Объясните назначение флагов состояния процессора C, AC, OV, P. Опишите условия, при которых каждый из флагов изменяется, приведите примеры.
6. Изобразите упрощенную схему организации памяти микроконтроллера семейства 8051. Обозначьте на ней область регистров общего назначения, адресов памяти данных, область памяти данных с битовой адресацией, область регистров специальных функций.
7. Какие методы адресации поддерживаются микроконтроллером семейства 8051? Объясните суть каждого метода адресации. Что служит исполнительным адресом, где он хранится (для разных методов)?
8. Какие методы адресации применяются для доступа к регистрам специальных функций, для доступа к внешней памяти данных, чтения памяти программ?
9. Каков максимальный объем внешней памяти данных?

10. Что такое стек? Для чего он предназначен?

11. Опишите процесс обработки прерывания в микроконтроллере. В каком случае инициируется этот процесс? Что происходит при возврате из прерывания?

12. Что такое приоритеты прерываний? Для чего предназначена поддержка многоприоритетных прерываний?

13. При помощи блок схемы изобразите структуру программы, использующей прерывания. Покажите на схеме основную программу, векторы прерывания и процедуры обработки прерываний.

14. Объясните назначение таймеров микроконтроллера. Расскажите об основных режимах их работы.

15. В чем отличие директив ассемблера от мнемонических команд?

16. Объясните назначение директив ассемблера EQU, ORG, DB.

17. Что необходимо предпринимать для обработки данных разрядностью больше восьми?

Список литературы

1. Сташин В. В. и др., Проектирование цифровых устройств на однокристальных микроконтроллерах. — М.: Энергоатомиздат, 1990. — 224 с.

2. Каган Б. М., Электронно-вычислительные машины и системы: Учеб. пособие для вузов — 3-е изд. перераб. и доп. — М.: Энергоатомиздат, 1991. — 592 с.

3. Микропроцессоры, в 3-х кн., под ред. Л. Н. Преснухина, кн. 1, Архитектура и проектирование микро-ЭВМ: Учеб. для вузов. — М.: Высш. шк., 1986. — 495 с.

Содержание