Министерство образования Российской Федерации

Томский политехнический университет

УТВЕРЖДАЮ
Зав. каф. промышленной и
медицинской электроники,
профессор, д-р техн. наук

Г. С. Евтушенко

«24» октября 2003 г.

PIC-КОНТРОЛЛЕРЫ

Методические указания к выполнению лабораторной работы по дисциплине «Основы микропроцессорной техники» для студентов направлений 550.700 «Электроника и микроэлектроника» и 553.400 «Биомедицинская инженерия».

Томск 2004

УДК 681.325.5-181.48.001.63
PIC-КОНТРОЛЛЕРЫ: Методические указания к выполнению лабораторной работы по дисциплине «Основы микропроцессорной техники» для студентов направления 550.700 «Электроника и микроэлектроника» и для студентов направления 553.400 «Биомедицинская инженерия» - Томск: Изд-во ТПУ, 2004. – 60с.

Составители:
Г.С. Воробьева
Д.В. Добровольский
А.В. Крыцкий
Д.В. Пайгин
Д.А. Пестунов

Рецензент доцент, к.т.н. А.И. Солдатов

Методические указания рассмотрены и рекомендованы к изданию методическим семинаром кафедры промышленной и медицинской электроники 23 октября 2003

Зав. кафедрой
Проф., д-р техн. наук Г. С. Евтушенко

Лабораторная работа №1. Обучение работе на внутрисхемном отладчике MPLAB-ICD.

1. ЦЕЛЬ РАБОТЫ

Получить общее представление о работе микроконтроллера с Гарвардской архитектурой на примере PIC16F877. Ознакомиться с интегрированной средой разработки MPLAB IDE. Изучить основные функции и назначение основных пунктов меню. Получить практические навыки по работе с инструментальными средствами отладки микропроцессорных систем MPLAB ICD.

2. ВВЕДЕНИЕ

Современный рынок электронных систем предъявляет все более жесткие требования к разрабатываемым устройствам. Данные системы должны быть просты по своей схемотехнике, надежны, высокотехнологичны, иметь большие ресурсы в части модификации выполняемых функций и иметь низкую стоимость. При этом очень часто при разработке электронных устройств приходится увязывать взаимоисключающие моменты:

простоту схемотехники при большом наборе выполняемых функций;

малое энергопотребление при высоком быстродействии;

малые затраты на подготовку производства для обеспечения серийного выпуска изделий.

Всем вышеперечисленным задачам и требованиям соответствуют устройства, разработанные на базе микроконтроллеров, в частности семейств PIC12/14/16/17/18 фирмы Microchip, которые объединяют все передовые технологии микроконтроллеров: электрически программируемые пользователем ППЗУ, минимальное энергопотребление, высокую производительность, хорошо развитую RISC-архитектуру, функциональную законченность и минимальные размеры.

Все обширное семейство, выпускаемое под торговой маркой PICmicro, объединяет ряд особенностей: Гарвардская архитектура, простой набор RISC команд, высокое быстродействие (до 8,5 MIPS), широкий диапазон питания (2,0- 6,0 В), низкое потребление (от 15 мкА при 3 В/ 32 кГц до 10 мА при 5 В/ 20 МГц); при огромном разнообразии периферийных устройств и корпусов (от SOIC-8 до TQFP-80). Все типы

контроллеров в корпусах с одинаковым количеством выводов имеют и одинаковую цоколевку, что упрощает модификацию уже разработанных изделий и позволяет при переходе к крупносерийному производству безболезненно заменять Flash версии, на дешевые – однократно программируемые. По числу выпущенных микроконтроллеров фирма MICROCHIP вышла на второе место в мире, опередив таких производителей, как Intel, SGS-Thomson, Zilog, Siemens.

Микроконтроллеры PICmicro поддерживаются полным набором программных и аппаратных средств разработки. Фирма Microchip предоставляет бесплатно – интегрированную среду разработки MPLAB IDE.

Авторы благодарят фирму Microchip и ее представителей в России – фирму «Гамма Санкт-Петербург», а так же лично господина Воротынского, за информационную поддержку, и выделение комплектов отладочных средств, для организации цикла лабораторных работ по изучению микроконтроллеров с Гарвардской архитектурой.

3. КРАТКИЕ СВЕДЕНИЯ О PIC16F877

3.1. КРАТКОЕ ОПИСАНИЕ АРХИТЕКТУРЫ

3.1.1. Возможности микропроцессорного ядра:

- Высокопроизводительная RISC архитектура;
- 35 однословных команд, все команды выполняются за один цикл, кроме команд переходов, ветвления и табличного чтения-записи, которые выполняются за 2 машинных цикла;
- Тактовая частота – до 20 МГц, длина машинного цикла – 200нсек;
- 8К x 14 - ти разрядных слов FLASH памяти программ;
- 368 x 8 байт памяти данных (RAM);
- 256 x 8 байт электрически перезаписываемой (EEPROM) памяти данных;
- Восьмиуровневый аппаратный стек;
- До 14 источников прерываний;
- Сброс по включению питания (Power-on Reset – POR);
- Таймер включения питания (PWRT) и таймер запуска генератора (OST);
- Сторожевой таймер (Watchdog timer – WDT) со встроенным на кристалле RC-генератором;
- Биты защиты памяти программ;
- Режим пониженного энергопотребления (SLEEP mode);
- Программируемый выбор генератора;

- Низко потребляемая, высокоскоростная КМОП FLASH/EEPROM технология;
- Полностью статическое устройство (частота работы от 0 до 20 МГц);
- Возможность внутрисхемного программирования (ICSP) и внутрисхемной отладки (ICD) по двум выводам;
- Широкий диапазон напряжений питания (от 2,0 В до 5,5 В);
- Высокая нагрузочная способность: 25 мА.

3.1.2. Описание периферии.

Таймер 0 (TMR0) – 8-разрядный таймер/счетчик с предделителем;

Таймер 1 (TMR1) – 16-разрядный таймер/счетчик с предделителем, который может инкрементироваться от внешнего сигнала в спящем режиме;

Таймер 2 (TMR2) – 8-разрядный таймер/счетчик с предделителем и постделителем, переполнение происходит при достижении таймером значения записанного в регистре PR2;

Два модуля Захвата (Накопления)/Сравнения/ШИМ:

захват (Capture) – 16-разрядный, максимальное разрешение 12,5 нсек;

сравнение (Compare) – 16-разрядный, разрешение до 200 нсек;

ШИМ (PWM) максимальное разрешение 10 бит;

10-разрядный, многоканальный АЦП;

Последовательные синхронные порты: SSP, SPI, I^2C;

Универсальный асинхронный приемопередатчик (UART);

Параллельный ведомый порт (PSP).

Архитектура PIC16F877 показана на рис. 1.

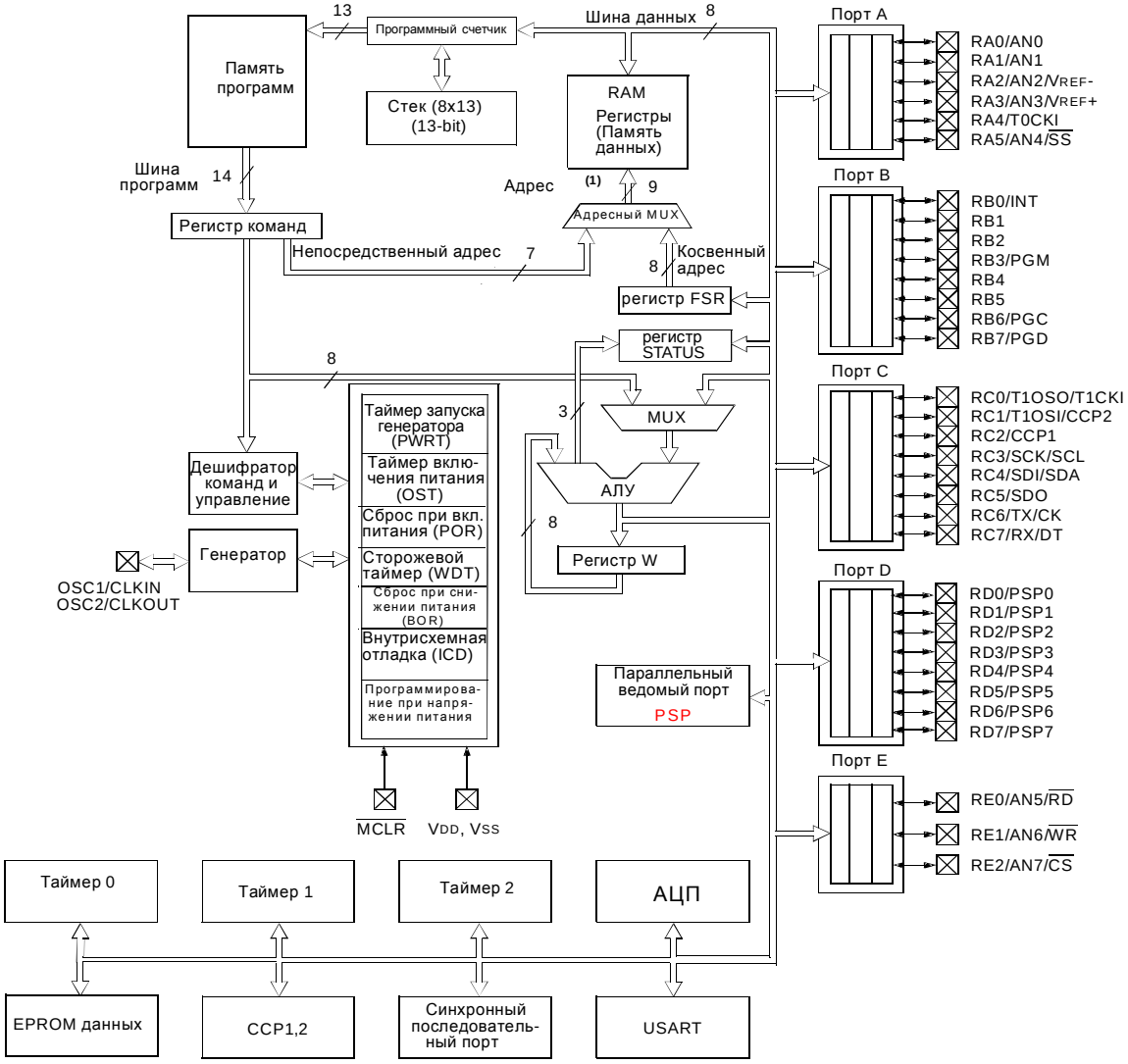


Рис. 1. Архитектура PIC16F877.

3.2. ПАМЯТЬ ПРОГРАММ И СТЕК

Так как микроконтроллеры данного семейства имеют Гарвардскую архитектуру, то память программ и память данных у каждого из них физически и логически разделены. Память в микроконтроллерах PICmicro разделена на два независимых блока: память программ и память данных. Каждый блок имеет собственную шину данных и шину адреса, позволяя организовать одновременный доступ к обоим типам памяти в течение одного машинного цикла.

Микроконтроллер имеет 13-разрядный счетчик программ, способный адресовать 8К памяти. После сброса счетчик устанавливается на 0000h, а любое прерывание вызовет переход на адрес 0004h. Организация памяти программ показана на рис. 2.

Следует знать, что команды CALL и GOTO способны адресовать только 11-разрядный интервал. Это значит, что возможен переход только внутри страницы памяти программ размером 2К. Для того, что

бы адресовать все адресное пространство памяти программа должна переключать разряды выбора страниц, которые находятся в регистре PCLATH (биты 4 и 3). При возврате из подпрограммы или из прерывания из стека выталкивается все 13-битное значение адреса.

Микроконтроллер имеет 13-разрядный аппаратный стек, глубиной 8 уровней. Стек не относится ни к памяти программ, ни к памяти данных. Указатель стека пользователю недоступен. В стеке запоминается содержание программного счетчика при выполнении команды CALL или при вызове прерывания. Извлечение из стека осуществляется командами RETURN, RETFIE.

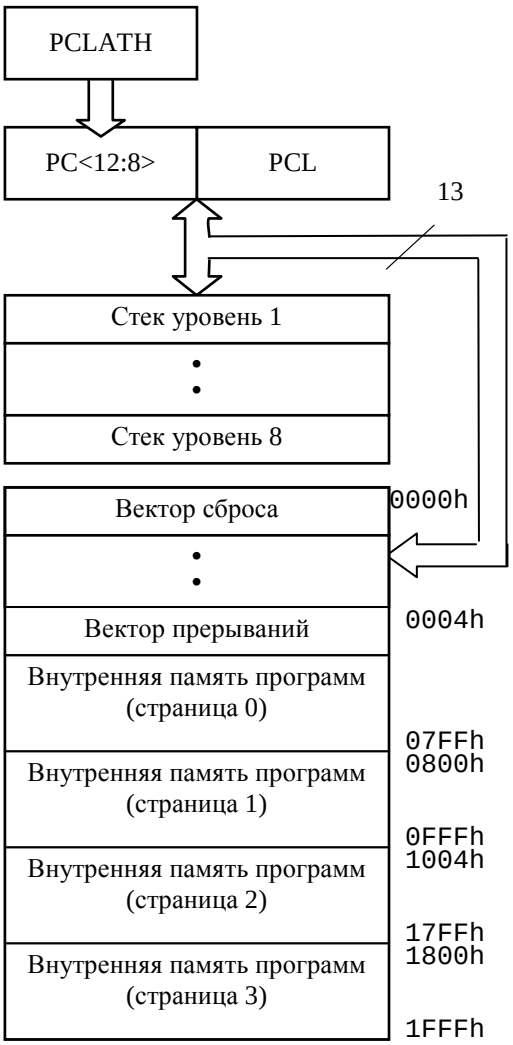


Рис. 2. Карта памяти программ и стек

3.3. ПАМЯТЬ ДАННЫХ

Память данных разбита на четыре так называемых банка, каждый из которых имеет адресуемое пространство 128 байт. Выбор каждого банка осуществляется переключением битов RP1 и RP0 (биты 6 и 5, регистра STATUS).

Младшие ячейки каждого банка зарезервированы для регистров специальных функций, потом расположены универсальные регистры, выполненные как статическая память. Для уменьшения программы и более быстрого доступа некоторые часто используемые регистры специальных функций расположенные в одном банке, могут быть отображены в другом банке. Карта памяти данных PIC16F877 представлена в Приложении 1.

3.4. РЕГИСТР СОСТОЯНИЯ STATUS

Регистр состояния STATUS (см табл. 1) содержит флаги результатов выполнения операций АЛУ, состояния сторожевого таймера и биты выбора банка памяти данных.

Таблица 1

Регистр состояния **STATUS** (адреса **03h, 83h, 103h, 183h**)

№ бита	7	6	5	4	3	2	1	0
Имя бита	IRP	RP1	RP0	-TO	-PD	Z	DC	C
Состояние после сброса	0	0	0	1	1	x	x	x
Доступность.	R/W	R/W	R/W	R	R	R/W	R/W	R/W
Обозначения: R – читаемый разряд, W – записываемый разряд, x – неопределенное значение								

Назначение битов регистра состояния приведено ниже.

Бит 7: **IRP** – выбор банка (используется для косвенной адресации)

1=Банк 2,3 (100h - 1FFh)

0= Банк 0,1 (00h - FFh)

Биты 6-5: **RP1:RP0** – выбор банка (используется для прямой адресации)

11- Банк 3 (180h - 1FFh)

10- Банк 2 (100h - 17Fh)

01- Банк 1 (80h - FFh)

00- Банк 0 (00h - 7Fh)

Бит 4: **-TO –** выход сторожевого таймера (WDT)

1=после включения питания или команды CLRWDT, SLEEP

0=после срабатывания WDT

Бит 3: **-PD** – флаг включения питания.

1=после включения питания (POR) или команды CLRWDT (см п.3.11)

0=после выполнения команды SLEEP

Бит 2: **Z** – признак нуля

1=если результат арифметической или логической операции = ”0”

0= если результат арифметической или логической операции не ”0”

Бит 1: **DC** – дополнительный перенос/заем после команд ADDWF, ADDLW, SUBLW, SUBWF (для заема - инверсия)

1= если перенос из 4-го младшего разряда результата

0= если нет переноса из 4-го младшего разряда результата

Бит 1: **C** – перенос/заем после команд ADDWF, ADDLW, SUBLW, SUBWF

1=если есть перенос из старшего разряда результата

0=если нет переноса из старшего разряда результата

Примечание: Заем имеет обратное значение. Вычитание выполняется сложением, при этом вычитаемое представляется в дополнительном коде. Команды сдвигов (RRF, RLF) выполняются через перенос.

Регистр STATUS может быть адресован любой командой как любой другой регистр. Если он – адресат команды, которая модифицирует флаги Z, C, DC, то запись в эти биты не производится. Они устанавливаются в соответствии с результатом операции. Кроме того, биты TO и PD не изменяются. Следовательно, результат операции с регистром STATUS может быть отличным от ожидаемого.

3.5. РЕГИСТР OPTION_REG

Регистр OPTION_REG доступен для чтения и для записи. Он содержит биты: конфигурации предделителя (PSC) для TMR0/WDT, внешнего прерывания INT и состояния выходов порта B. Описание битов регистра OPTION_REG показано в таблице 2.

Таблица 2

Регистр **OPTION_REG** (адреса 81h, 181h)

№ бита	7	6	5	4	3	2	1	0
Имя бита	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
Состояние после сброса	1	1	1	1	1	1	1	1
Доступность.	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Обозначения: R – читаемый разряд, W – записываемый разряд.								

Назначение битов регистра OPTION_REG приведено ниже.

Бит 7: **RBPU**: включение подтягивающих резисторов на порте B

0 = подтягивающий резистор включен

1 = подтягивающий резистор выключен

Примечание: Если используется низкий уровень напряжения при программировании по входу PGM и выводы PORTB подтянуты к высокому уровню, то 3-ий бит в регистре TRISB необходимо сбросить, чтобы вход RB3 не был подтянут к высокому уровню. Это необходимо для правильного программирования устройства.

Бит 6: **INTEDG**: выбор активного фронта сигнала прерывания на входе RB0/INT

0 = прерывание по заднему фронту.

1 = прерывания по переднему фронту.

Бит 5: **TOCS**: выбор источника тактового сигнала для TIMER0.

1 = тактовый сигнал с входа RA4/T0CKI.

0 = внутренний источник тактового сигнала (CLKOUT).

Бит 4: **T0SE**: выбор фронта приращения TMR0 при внешнем тактовом сигнале

1 = приращение по заднему фронту сигнала на T0CKI.

0 = приращение по переднему фронту сигнала на T0CKI.

Бит 3: **PSA**: выбор включения предделителя.

1 = предделитель включен перед таймером WDT.

0 = предделитель включен перед таймером TMR0.

Биты 2-0: **PS2 - PS0**; выбор коэффициента предделителя. Коэффициенты деления предделителя показаны в таблице 3.

Таблица 3

Коэффициенты деления в зависимости от значений PS2- PS0

Значение			TMR0 (PSA = 0)	WDT (PSA = 1)
PS2	PS1	PS0		
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

Установка коэффициента деления предделителя 1:1 для TMR0 соответствует переключению предделителя на сторожевой таймер.

3.6. РЕГИСТР INTCON

Регистр INTCON доступен для чтения и для записи. Он содержит биты масок прерываний и флаги прерываний. Флаги прерываний должны сбрасываться программно. Регистр показан в таблице 4.

Таблица 4

Регистр INTCON (адреса 0Bh, 8Bh, 10Bh, 18Bh)

№ бита	7	6	5	4	3	2	1	0
Имя бита	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
Состояние после сброса	0	0	0	0	0	0	0	x
Доступность.	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Обозначения: R – читаемый разряд, W – записываемый разряд, x – неопределенное значение								

Назначение битов регистра INTCON приведено ниже.

Бит 7: **GIE** – Общее (глобальное) управление прерываниями.

1=все немаскируемые прерывания разрешены.

0=все прерывания запрещены.

Примечание: Если происходит прерывание, то бит GIE сбрасывается. По команде выхода из подпрограммы (RETFIE) этот бит устанавливается.

Бит 6: **PEIE** – маска прерываний от периферийных устройств

1=все немаскируемые прерывания периферийных устройств разрешены

0=все прерывания периферийных устройств запрещены

Бит 5: **T0IE** – маска прерывания по переполнению TMR0

1=прерывание TMR0 разрешено

0= прерывание TMR0 запрещено

Бит 4: **INTE** – маска внешнего прерывания по входу RB0/INT

1=прерывание по входу RB0/INT разрешено

0=прерывание по входу RB0/INT запрещено

Бит 3: **RBIE** – маска прерывания по изменению состояния на входах порта RB

1=прерывание по изменению порта RB разрешено

0=прерывание по изменению состояния порта RB запрещено

Бит 2: **T0IF** – флаг прерывания при переполнении TMR0

1=устанавливается, если регистр TMR0 переполнен (очищается программно)

0=если регистр TMR0 не переполнен

Бит 1: **INTF** – флаг внешнего прерывания по входу RB0/INT

1=устанавливается, если происходит прерывание по входу RB0/INT

0=если прерывание по входу RB0/INT не произошло

Бит 0: **RBIF** – флаг прерывания по изменению состояния порта RB

1=устанавливается, если изменилось состояние на одном из входов RB7:RB4 (очищается программно)

0=если состояния на входах RB7:RB4 не изменились.

3.7. РЕГИСТР PIE1

Регистр PIE1 доступен для чтения и для записи. Он содержит маски прерываний периферийных устройств. Для того, что бы разрешить прерывания от периферийных устройств, необходимо установить бит PEIE (бит 6 регистра INTCON). Описание битов регистра PIE1 приведено в таблице 5.

Таблица 5

Регистр **PIE1** (адрес 8Ch)

№ бита	7	6	5	4	3	2	1	0
Имя бита	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
Состояние после сброса	0	0	0	0	0	0	0	X
Доступность.	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Обозначения: R – читаемый разряд, W – записываемый разряд, x – неопределенное значение								

Назначение битов регистра **PIE** приведено ниже.

Бит 7: **PSPIE** – маска прерывания параллельного ведомого порта

1=прерывание при чтении/записи порта разрешено

0=прерывание при чтении/записи порта запрещено

Бит 6: **ADIE** – маска прерываний A/D преобразования

1=прерывание A/D разрешено

0=прерывание A/D запрещено

Бит5: **RCIE** – маска прерываний приемника USART

1=прерывание от приемника USART разрешено

0=прерывание от приемника USART запрещено

Бит 4: **TXIE** - маска прерываний передатчика USART

1=прерывание от передатчика USART разрешено

0=прерывание от передатчика USART запрещено

Бит 3: **SSPIE** – маска прерывания синхронного последовательного порта

1=прерывание SSPIE разрешено

0=прерывание SSPIE запрещено

Бит 2: **CCP1IE** – маска прерывания CCP1

1=прерывание CCP1 разрешено

0=прерывание CCP1 запрещено

Бит 1: **TMR2IE** – маска прерывания равенства TMR2 и регистра PR2

1=прерывание при равенстве TMR2 и регистра PR2 разрешено

0=прерывание при равенстве TMR2 и регистра PR2 запрещено

Бит 0: **TMR1IE** – маска прерывания по переполнению TMR1

1=прерывание по переполнению TMR1 разрешено

0= прерывание по переполнению TMR1 запрещено.

3.8. РЕГИСТР PIE2

Регистр PIE2 доступен для чтения и для записи. Он содержит маски прерываний модуля CCP2, записи EEPROM и при конфликте на шине. Описание битов регистра PIE2 приведено в таблице 6.

Таблица 6

Регистр **PIE2** (адрес 8Dh)

№ бита	7	6	5	4	3	2	1	0
Имя бита	–	–	–	EEIE	BCLIE	–	–	CCP2IE
Состояние после сброса	0	0	0	0	0	0	0	0
Доступность.	U	R/W	U	R/W	R/W	U	U	R/W
Обозначения: R – читаемый разряд; W – записываемый разряд; U – физически бит не существует, при обращении читается как 0.								

Назначение битов регистра **PIE2** приведено ниже.

Бит 4: **EEIE** – маска прерываний для записи EEPROM

1=прерывания для записи EEPROM разрешены

0=прерывания для записи EEPROM запрещены

Бит 3: **BCLIE** – прерывания при конфликте на шине

1= прерывания в случае конфликта на шине разрешены

0=прерывания запрещены

Бит 0: **CCP2IE** – маска прерываний CCP2

1=прерывания CCP2 разрешены

0=прерывания CCP2 запрещены

3.9. РЕГИСТР PIR1

Регистр PIR1 доступен для чтения и для записи. Он содержит флаги прерываний периферийных устройств. Флаги должны быть сброшены программно. Описание битов регистра PIR1 приведено в таблице 7.

Таблица 7

Регистр **PIR1** (адрес 0Ch)

№ бита	7	6	5	4	3	2	1	0
Имя бита	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
Состояние после сброса	0	0	0	0	0	0	0	0
Доступность.	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Обозначения: R – читаемый разряд; W – записываемый разряд.								

Назначение битов регистра PIR1 приведено ниже.

Бит 7: **PSPIF** – флаг прерывания параллельного ведомого порта

1=если выполнена операция чтения/записи параллельного порта

0=если нет операции чтения/записи параллельного порта

Бит 6: **ADIF** – флаг прерываний A/D преобразования

1=если преобразование завершено A/D

0= если преобразование не завершено A/D

Бит 5: **RCIF**- флаг прерываний от приемника USART

1=буфер приемника USART наполнен

0= буфер приемника USART пуст

Бит 4: **TXIF**- флаг прерываний передатчика USART

0=буфер передатчика USART наполнен

1= буфер передатчика USART пуст

Бит 3: **SSPIF**- флаг прерывания синхронного последовательного порта

1=если произошло прерывание SSP, то он должен быть программно очищен перед возвращением из подпрограммы обработки прерываний. Условия установки этого бита, когда работа SSP происходит:

В SPI режиме – если произошла передача/прием данных.

В режиме I^2C ведомого – если произошла передача/прием данных.

I^2C ведущего – если произошла передача/прием данных.

0=если прерываний SSP не было.

Бит 2: **CCP1IF**- флаг прерывания CCP1

Режим захвата

1=если значение TMR1 зафиксировано

0=если фиксация не произошла

Режим сравнения

1=если значение TMR1 равно заданному

0=если нет равенства

Режим PWM

В данном режиме не используется.

Бит 1: **TMR2IF**- флаг прерывания равенства TMR2 и регистра PR2

1=устанавливается при равенстве TMR2 и регистра PR2

0=если нет равенства

Бит 0: **TMR1IF**- флаг прерывания по переполнению TMR1

1=если произошло переполнение TMR1

0= если нет переполнения TMR1

3.10. РЕГИСТР PIR2

Регистр PIR2 доступен для чтения и для записи. Он содержит флаги прерываний модуля CCP2, записи EEPROM и при конфликте на шине. Описание битов регистра PIR2 приведено в таблице 8.

Таблица 8

Регистр **PIR2** (адрес 0Dh)

№ бита	7	6	5	4	3	2	1	0
Имя бита	–	–	–	EEIF	BCLIF	–	–	CCP2IF
Состояние после сброса	0	0	0	0	0	0	0	0
Доступность.	U	R/W	U	R/W	R/W	U	U	R/W
Обозначения: R – читаемый разряд; W – записываемый разряд; U – физически бит не существует, при обращении читается как 0.								

Назначение битов регистра **PIR2** приведено ниже.

Бит 4: **EEIF** –флаг прерываний для записи EEPROM

1=запись EEPROM закончена

0= запись EEPROM не закончена или не начиналась

Бит 3: **BCLIF** – прерывания при конфликте на шине

1= случай конфликта на шине имеет место в SSP, в режиме I^2C-ведущего

0=если нет конфликта на шине.

Бит 0: **CCP2IF** –флаг прерываний CCP2

Режим захвата

1=если значение TMR1 захвачено

0=если захвата не произошло

Режим сравнения.

1=если значение TMR1 равно заданному

0=если нет равенства

Режим PWM

В данном режиме не используется.

3.11. РЕГИСТР PCON

Регистр управления питанием (power control) **PCON** доступен для чтения и для записи. Он содержит флаги позволяющие определить, произошел ли сброс микроконтроллера

при включении питания (POR)

по сигналу на выводе MCLR

от сторожевого таймера WDT

по обнаружению снижения питания (BOR).

Описание битов регистра приведено в таблице 9.

Таблица 9

Регистр **PCON** (адрес 8Eh)

№ бита	7	6	5	4	3	2	1	0
Имя бита	–	–	–	–	–	–	POR	BOR
Состояние после сброса	0	0	0	0	0	0	0	1
Доступность.	U	U	U	U	U	U	R/W	R/W
Обозначения: R – читаемый разряд; W – записываемый разряд; U – физически бит не существует, при обращении читается как 0.								

Назначение битов регистра **PCON** приведено ниже.

Бит 1: POR - флажок «Сброс» при включении питания

1=«Сброс» при включении питания не происходил

0=«Сброс» при включении питания (должен быть установлен программой после «Сброса» при включении питания)

Бит 0: BOR - флаг «Сброс» при снижении питания

1=«Сброс» при снижении питания не происходил

0=«Сброс» при снижении питания (должен быть установлен программой после «Сброса» при снижении питания)

3.12. СИСТЕМА КОМАНД

Каждая команда PIC16C87XX представляет собой 14-битовое слово, которое разделено на следующие части: - 1. код операции, -2. поле для одного операнда, который может участвовать или нет в этой команде. Система команд PIC16C87XX включает в себя байт-ориентированные команды, бит-ориентированные, операции с константами и команды передачи управления.

Для байт-ориентированных команд "f" обозначает собой регистр, с которым производится действие; "d" - бит определяет, куда положить результат. Если "d" =0, то результат будет помещен в регистр W

(аккумулятор), при "d"=1 результат будет помещен в регистр с адресом "f", упомянутом в команде.

Для бит-ориентированных команд "b" обозначает номер бита, участвующего в команде, а "f" -это регистр, в котором этот бит расположен.

Для операций с константами и команд передачи управления, "k" обозначает восьми или одиннадцати битную константу.

Все команды выполняются в течение одного командного цикла. В двух случаях исполнение команды занимает два командных цикла: проверка условия и переход либо изменение программного счетчика как результат выполнения команды GOTO и CALL. Один командный цикл состоит из четырех периодов генератора. Таким образом, для генератора с частотой 4 МГц время исполнения командного цикла будет 1 мкс, а производительность контроллера составит 1 миллион операций в секунду (1 MIPS – 1 million instructions per second). Принятые обозначения в системе команд и сама система команд PICmicro представлены в приложении 2.

4. КРАТКОЕ ОПИСАНИЕ ОТЛАДОЧНЫХ СРЕДСТВ

4.1. ОБЩИЕ СВЕДЕНИЯ

Существует множество инструментальных средств, предназначенных для облегчения и интенсификации труда разработчиков. Основное назначение любого отладчика – помочь разработчику разобраться на программно – аппаратном уровне в процессах происходящих в устройстве, а затем добиться желаемых характеристик функционирования. Кроме этого инструментальные средства могут обладать дополнительными возможностями, которые избавляют разработчика от множества утомительных процедур.

Наиболее универсальными и мощными средствами отладки микропроцессорных систем, на сегодня являются внутрисхемные отладчики, к которым относится и используемый нами модуль MPLAB ICD, компании Microchip Technology Inc.

4.2. ОБЩЕЕ ОПИСАНИЕ MPLAB IDE И MPLAB–ICD

MPLAB IDE (*Integrated Development Environment)* – интегрированная среда разработки, представляет собой программный продукт, работающий под управлением операционной системы *WINDOWS* и предназначенный для написания, отладки и оптимизации программ для *Microchip PIC* контроллеров. *MPLAB IDE* включает в себя:

MPLAB Project Manager– менеджер проекта;

MPLAB Editor – текстовый редактор,

MPLAB-SIM Simulator – симулятор;

MPASM – универсальный Ассемблер, а так же MPLINK (Линковщик) и MPLIB (Библиотекарь).

Для контроллеров семейства PIC17, PIC18, а так же dsPIC, в среде MPLAB IDE есть возможность работать с компиляторами языка Си – более высокого уровня, по сравнению с ассемблером.

Ряд аппаратных средств, работающих под управлением MPLAB IDE, приведен ниже.

MLPAB-ICE Emulator – эмулятор;

программаторы PICSTART Plus и PRO MATE 2;

инструментальные средства третьих лиц – большое количество других компаний делают инструментальные средства разработки, работающие с *MPLAB.*

MPLAB-ICD (*In Circuit Debugging)* - внутрисхемный отладчик, позволяет выполнять внутрисхемную отладку программы в реальной схеме, используя генератор и периферию отлаживаемого контроллера. Для отладки используются только два вывода контроллера, работа идёт по *ICSP*^{TM} интерфейсу.

Используя *ICD*, можно отлаживать аппаратно-зависимые участки кода, которые трудно, а порой невозможно воспроизвести в симуляторе, например: работа с АЦП, измерение временных параметров входного сигнала, организация обратной связи с управляемым объектом, отладка интерфейсов *USART, SPI, I*^{2}*C* и т.п.

4.3. MPLAB–ICD

MPLAB-ICD работает под управлением универсальной программной среды *MPLAB* и обладает следующими возможностями:

отладка в режиме реального времени и пошаговая отладка.

связь с компьютером по *RS-232*.

одна задаваемая точка останова.

просмотр и модификация содержимого управляющих регистров, *RAM* и *EEPROM*.

внутрисхемная отладка и встроенная система программирования *PIC*-контроллеров серии *PIC16F87x*.

работа от источника питания отлаживаемой конструкции в диапазоне от 3,0 до 5,5 В.

диапазон рабочих частот от 32 кГц до 20 МГц.

Внешний вид комплекта *MPLAB ICD* показан на рис. 3. В него входят:

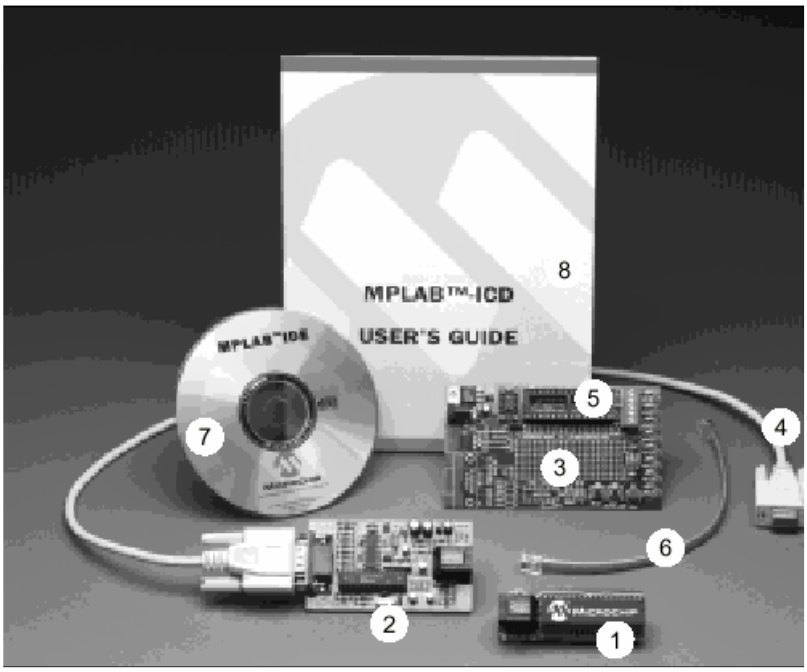


Рис. 3. Внешний вид комплекта MPLAB-ICD.

1. Эмуляционная плата *MPLAB ICD* header с микроконтроллером *PIC16F877-20/P*.

2. Основной модуль *MPLAB ICD* module.

3. Демонстрационная макетная плата *MPLAB ICD* demo board (см приложение 4).

4. Кабель для подключения к компьютеру через RS 232.

5. 40-выводная и 28-выводная панели для подключения микроконтроллера или эмуляционной платы.

6. 9-дюймовый 6-выводной кабель.

7. CD с программным обеспечением и документацией.

8. Инструкции по эксплуатации.

Во время использования внутрисхемного программирования, *MPLAB-ICD* задействует следующие ресурсы микроконтроллера:

1. *MCLR/V_{pp}* используется для программирования.

2. *RB6 и RB7* зарезервированы для программирования и внутрисхемной отладки.

3. Шесть регистров общего назначения зарезервированы для *DEBUG MONITOR*a - *70h, 1EBh - 1EFh.*

4. Первая ячейка памяти программ должна содержать инструкцию *NOP.*

5. Память программ с адреса *0x1F00* по *0x1FFF* зарезервирована для кода отладки.

6. Один уровень стека недоступен.

Внимание! *MPLAB-ICD* не поддерживает низковольтное программирование. При использовании *ICD* функция низковольтного программирования должна быть отключена

4.3.1. Демонстрационная плата MPLAB ICD demo board.

Демонстрационная плата предназначена для демонстрации *PIC16F8XX* и изучения его возможностей. Плата подключается к основному модулю через эмуляционную плату *MPLAB-ICD Header*.

PIC16f8XX может быть вставлен непосредственно в демонстрационную плату в обход эмуляционной головки.

На рис. 4. показана демонстрационная плата.

Демонстрационная плата состоит из следующих частей.

1. 40 и 28 штырьковые разъемы.

2. Восемь *DIP* переключателей для соединения/разъединения каждого из восьми светодиодов с ответствующей линией порта **C**.

3. Восемь красных светодиодов подключенных к порту **C** для отображения 8-битных двоичных величин.

4. Две кнопки. Одна для сброса, вторая для внешнего воздействия на *RB*0 (при нажатии лог. «0»).

5. Потенциометр для подачи аналогового сигнала заданного уровня на RA0/AN0.

6. Область макетирования.

7. Разъем подключения внешних устройств к ножкам микроконтроллера – для расширения макетирования.

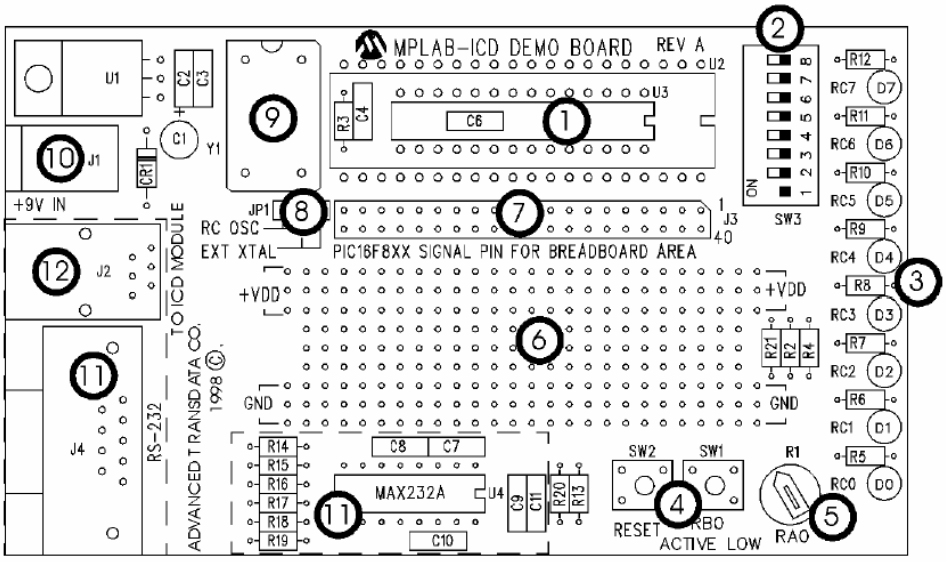


Рис. 4. Демонстрационная плата MPLAB ICD.

8. Перемычка выбора генератора - внутренний **RC** генератор (приблизительно 2 МГц) или внешний кристалл.

9. Разъем для внешнего кристалла.

10. Разъем питания - 9В, 0,75А.

11. Выход интерфейса RS-232.

12. Разъем для подключения основной платы MPLAB ICD.

В приложении 4 находятся фрагменты принципиальной схемы демонстрационной платы.

4.4. РАБОТА В СРЕДЕ MPLAB И РАБОТА С MPLAB-ICD

Работа с MPLAB возможна в двух режимах: в режиме симулятора MPLAB SIM – Simulator и в режиме внутрисхемной отладки MPLAB ICD Debugger.

В первом режиме программа функционирует в режиме симулятора *PIC* микроконтроллера на ЭВМ. Во втором работа идет с внутрисхемным отладчиком (непосредственно с микроконтроллером).

Работа с *MPLAB* идет в рамках проекта, который включает в себя файлы с исходным текстом, конечные файлы (*.*HEX*) и подключаемые библиотеки.

4.4.1. Создание нового проекта.

Выберите из меню Project>New Project.

В появившемся диалоге (рис. 5) необходимо ввести имя проекта в поле «*File Name*» например «*student.pjt*».

ВНИМАНИЕ! Придумывая имя своему проекту, не забывайте, что вы, скорее всего, не один Саша (Маша, Гриша, Даша) на курсе, а работаете Вы все в одной сети, поэтому имя должно быть **ОРИГИНАЛЬНЫМ**.

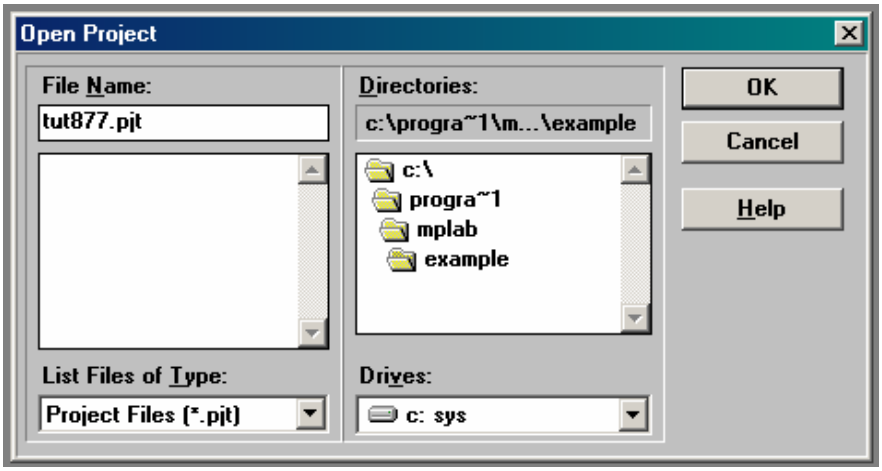


Рис. 5. Диалоговое окно нового проекта.

Нажмите кнопку «*OK*»

В следующем диалоге (рис. 6) в поле «*include Path*» указать путь к фалу «*p16f877.inc*».

Нажмите кнопку «*Change*»

В поле «Processor» выберите PIC16F877

Выберите нужный режим работы *MPLAB IDE* – режим симулятора (*MPLAB Sim – Simulator*) или режим внутрисхемного программирования и отладки (*MPLAB-ICD – Debugger*).

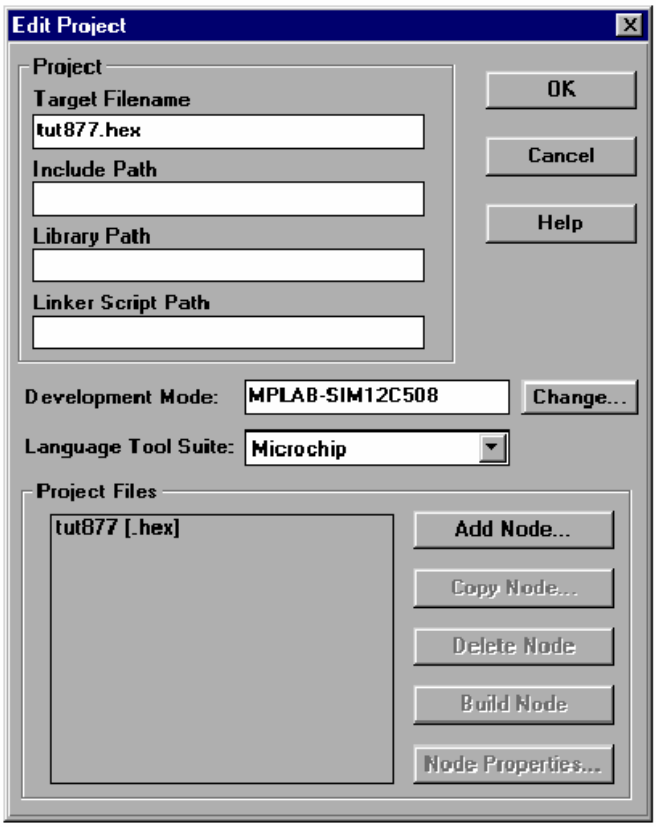


Рис. 6. Диалог редактирования проекта.

В начале работы выберите режим «*Simulator*», затем, когда проект будет набран, скомпилирован и отлажен на симуляторе, переключите *MPLAB* в режим (*MPLAB-ICD Debugger*).

Настройка (*MPLAB-ICD Debugger*) приведены ниже (рис. 7).

Device – PIC16f877
Oscillator – RC
Watchdog Timer – OFF
Power On Timer – ON
Brown out Detect – OFF
Low Voltage Program – Disable
Code Protect – Data EE code protection off

Flash Memory Write – No Memory written to by EECON
Code Protect – Code protection OFF

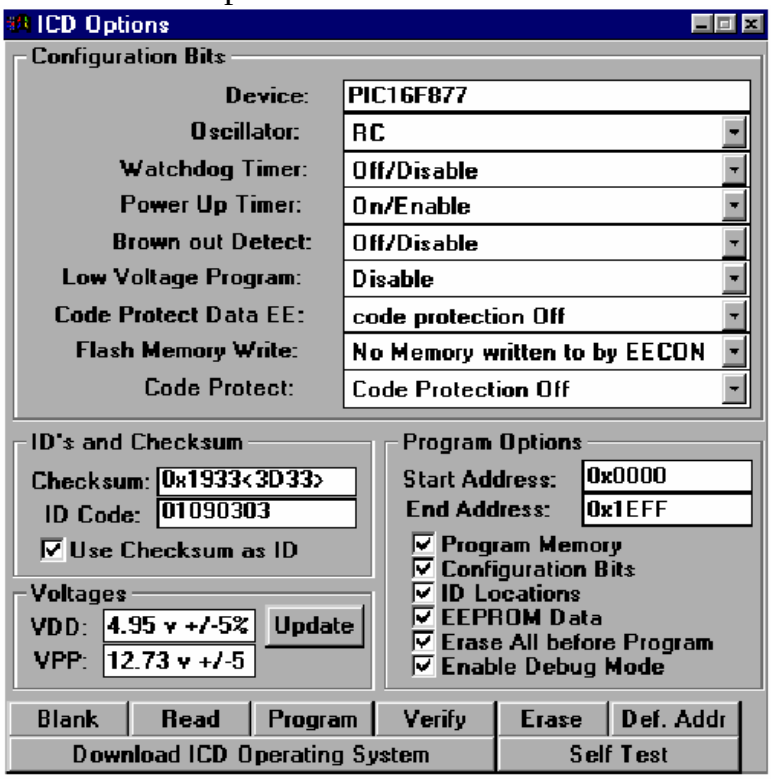


Рис. 7. Окно опций MPLAB-ICD.

Порт связи – *COM2*
Скорость связи – 19200 бод
Загружаемые регистры – all
Диапазон рабочих частот – 2Мгц – 10Мгц
Нажмите «*OK*».
Нажмите «*Add Node*».

В поле «*File Name*» введите имя исходного *ASM* файла. Он должен находится в том же каталоге, что и файл проекта. Рекомендуется называть файл исходных текстов тем же именем, что и проект.

Выберите из меню *File>New.*

Выберите *File>Save As…* в появившемся диалоге в поле «*File Name*» укажите имя файла, совпадающее с именем файла исходных текстов указанного в проекте.

4.4.2.Функции необходимые для работы MPLAB-IDE/ICD

Изменение параметров проекта – Project>Edit Project (CTRL+F3).
Компиляция проекта – Project>Make Project (F10).
Пошаговое выполнение программы – *Debug>Run>Step (F7).*
Остановка выполнения программы – *Debug>Run>Halt (F5).*

Сброс контроллера – Debug>Run>Reset (F6)

Запуск на выполнение – *Debug>Run>Run (F9)*

Для просмотра содержимого памяти контроллера в программном пакете *MPLAB* предусмотрены так называемые «Окна просмотра». Доступ к любому из них осуществляется через пункт меню «*WINDOWS*».

Special function register window – окно регистров специальных функций.

Program memory – Окно, отображающее содержимое памяти программ.

Absolute listing – Листинг программы.

Stack – Окно стека.

Symbol list – Окно, отображающее присвоенные имена.

Project – Окно позволяющее редактировать настройки проекта.

Watch windows – Данный пункт позволяет создать собственное «окно просмотра»,

New watch window – создание нового окна.

Load watch window– загрузка сохраненного окна.

Add to active watch window – Добавить переменную к активному окну.

Edit active watch window – Редактировать активное окно.

Save active watch window – сохранить активное окно.

При создании нового окна в меню необходимо выбрать переменную, которую необходимо отслеживать и нажать на кнопку *ADD*. При необходимости можно редактировать параметры отображения данной переменной.

5. ПРОГРАММА РАБОТЫ

5.1. Изучите краткое описание микроконтроллера и отладочной среды.

5.2. Создайте новый проект (по п.4).

5.3. Найти в рабочем каталоге файл tut877.asm и сохраните его, изменив фирменное имя на свое (sasha.asm, ivanov.asm, klava.asm). Используйте для этого меню: «*File > Open*» и : «*File > Save As*».

5.4. Проверьте все настройки программы (по п. 4). Исходный текст программы приведен ниже.

Программа 1.

Программа представляющая двоичное значение напряжения потенциометра на светодиодах порта **C**. Несомненно, что для написания и отладки программы необходимо детально знать схему включения

контроллера и периферии отлаживаемой системы. Принципиальную схему демонстрационной платы можно найти в приложении 4.

В программе используется модуль АЦП, описание регистров управления которого приведено ниже.

```
     list p=16f877
     include "p16f877.inc"

     org     0x000 ; Начало программы на вектор сброса
     nop
     Start
     banksel PORTC       ; Выбор 0 банка памяти данных
     clrf    PORTC       ; Очистка регистра порта C
     movlw   B'01000000' ; Включение АЦП. Частота Fosc/8
     movwf   ADCON0
     banksel TRISC       ; Выбор 1 банка памяти данных
     movlw  B'10000111'  ;Установка предделителя перед TMR0
     movwf OPTION_REG    ; коэффициент деления 1:256
     clrf    TRISC       ; Настройка порта C на вывод
     movlw B'00001110' ;Настройка АЦП - левое выравнивание
     movwf   ADCON1      ; аналоговый канал RA0
     banksel PORTC       ; Выбор 0 банка памяти данных
Main
     btfss   INTCON,T0IF ; Ждать переполнения TMR0
     goto    Main
     bcf  INTCON,T0IF    ;Сбросить флаг прерывания от TMR0
     bsf     ADCON0,GO   ; Запуск АЦП
Wait
     btfss   PIR1,ADIF   ; Ждать окончания преобразования
     goto    Wait
     movf    ADRESH,W    ;Вывод результата преобразования
     movwf   PORTC       ; на светодиоды порта C
WaitPush
     btfss   PORTB,0     ; Ждать нажатия на кнопку RB0
     goto    WaitPush
     goto    Main        ; Повтор программы
end
```

5.4. Запустите программу в PIC16F877. Изменяйте положение RA3, светодиоды должны отображать в двоичном коде напряжение потенциометра. У Вас этого не происходит. В программе намеренно сделана ошибка, и мы попробуем с ней разобраться. Любая из ниже перечисленных причин не позволит корректно работать программе.

– Значение преобразования АЦП не выводится на светодиоды (в PORTC).

– АЦП не включен или преобразование не запускается.

– В алгоритме программы имеется ошибка.

5.5. Проанализируем первую причину. Установите точку останова в строке программы:

movf ADRESH,W ;Write A/D result to PORTC

Для этого поместите курсор в соответствующую строку, нажмите правую кнопку мыши, что бы появилось меню (рис. 8). Выберите **Break Point(s).**

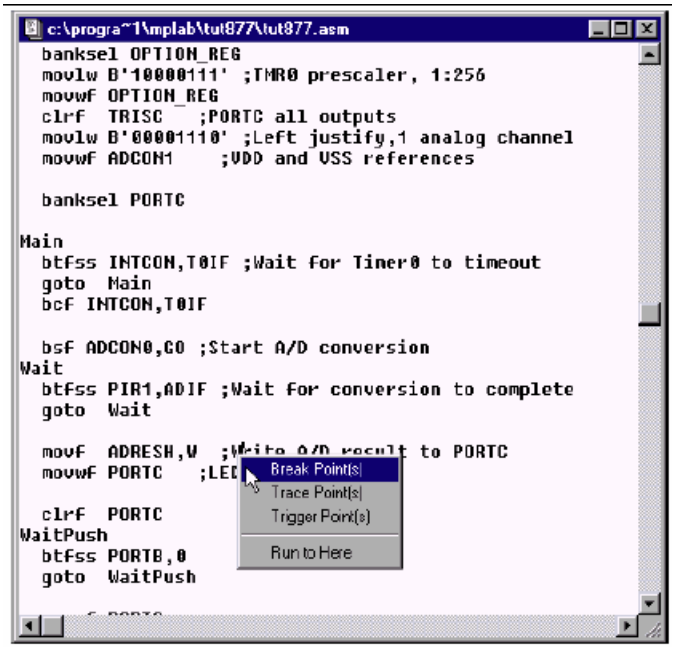


Рис. 8. Установка точки останова.

5.6. Запустите программу в режиме реального времени *(Debug > Run > Run*). Когда программа достигнет точки останова – она приостановится. Наша программа не остановится. Для принудительного останова необходимо выбрать меню: *Debug > Run > Halt.*

5.7. Посмотрите, где остановилась программа. Это должно произойти на одной из двух строк ожидания конца преобразования АЦП. Можно сделать вывод, что проблема в АЦП – флаг завершения преобразования не устанавливается. Инициализация АЦП происходит в начале программы, поэтому необходимо произвести сброс *(Debug > Run > Reset).*

5.8. Создайте окно просмотра (Watch window).

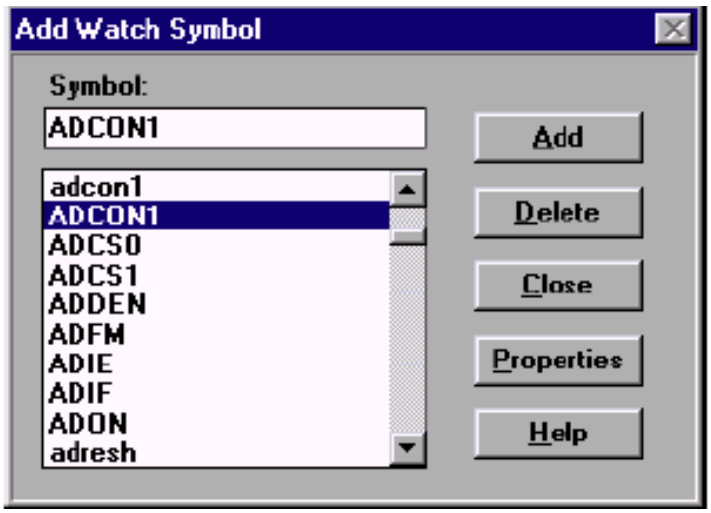


Рис. 9. Добавление регистров для просмотра

Для создания окна просмотра необходимо выбрать меню *Window > Watch Window > New Watch Window* . Откроется диалог Add Watch Symbol (добавление наблюдаемого символа) (рис. 9). Добавьте в него (Add) регистры ADCON0 и ADCON1 (как показано на рис. 10).

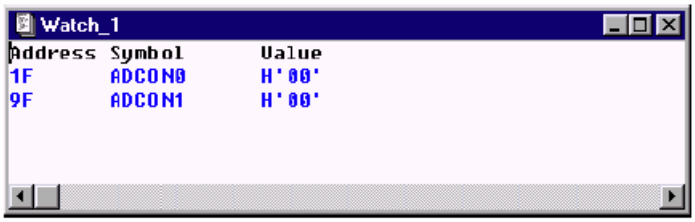


Рис. 10. Окно просмотра.

На рис. 10 показано окно просмотра – на нем должны быть видны выбранные регистры их адрес и содержимое. В диалоге MPLAB-ICD установите параметры загрузки в Minimum & Watch Window. Эта установка позволяет наблюдать изменение содержимого окна в процессе отладки, но программа работает медленнее.

5.9. Установите точку останова на строчку:

```
clrf PORTC          ;Clear PORTC
```

и запустите программу.

5.10. Теперь после останова программы на вашем экране вы должны увидеть картину, показанную на рис. 11.

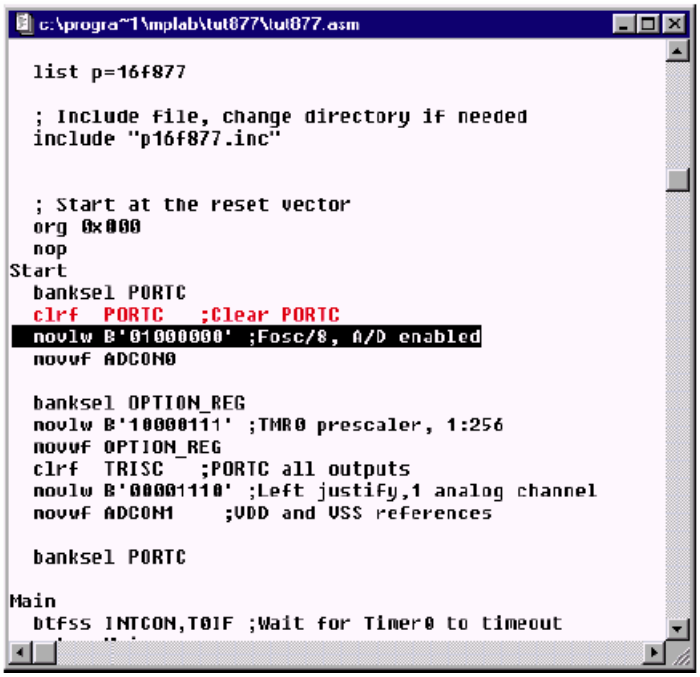


Рис.11. Фрагмент программы, приостановленной в точке останова.

5.11. Теперь, выполняя пошаговую отладку (*Debug* > *Run* > *Step*), сделайте два шага и посмотрите содержимое регистров управления АЦП (рис. 12). Распишите в двоичном коде полученные данные.

5.12. Для модуля АЦП последний (нулевой) бит должен быть 1, а не 0. Исправьте строчку:

movlw B'0100000**0**';Fosc/8, A/D enabled

на

movlw B'0100000**1**';Fosc/8, A/D enabled

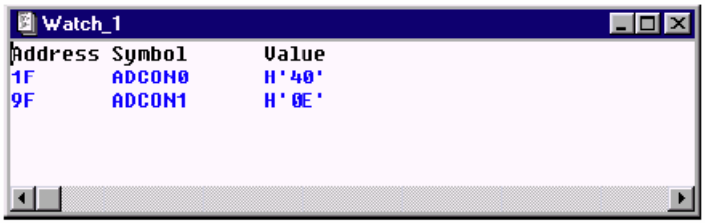


Рис. 12. Обновленное окно просмотра.

5.13. Теперь заново оттранслируйте программу (не забудьте сохраниться), запишите ее в память и запустите на выполнение. Программа должна работать.

Рекомендуется так же рассмотреть возможности окна Stopwatch в меню “Window”, для оценки времени работы отлаживаемой программы.

5.14. Самостоятельно наберите и проверьте работу программ, представленных ниже, сначала в симуляторе, а затем на демонстрационной плате.

Программа 2.

Данная программа выводит последовательно на линии порта **C** двоичный код чисел с 1 до 8 в порядке возрастания и убывания.

```
     list p=16f877
     include "p16f877.inc"

     cblock 0x71
     ;здесь можно присвоить имена регистрам с адреса 0x71
     endc

     banksel TRISC  ; Выбор второго банка регистров
     clrf TRISC; Очистка регистра-защелки порта C
     banksel PORTC  ; Выбор первого банка регистров
     movlw 1        ; Загрузка в аккумулятор «1»
     movwf PORTC    ; Пересылка аккумулятора в порт C
label1
     incf PORTC; Инкремент регистра порта C
     btfss PORTC,3  ; Пропустить переход при достижении 8
     goto label1
label2
     decfsz PORTC   ;Декремент порта C с проверкой на ноль
     goto label2
     goto label1

     end
```

Программа 3.

Программа последовательно инкрементирует регистр *PORTC*, после обнуления регистра счетчика (*counter*).

```
     list p=16f877
     include "p16f877.inc"

     cblock 0x71
     counter
     endc

     movlw 0x10; Загрузка в аккумулятор «10h»
     movwf counter  ; Пересылка содержимого аккумулятора в
;counter
     movlw 1
     movwf PORTC
label1
     decfsz counter ; Декремент counter,
; пропустить след. команду если 0
     goto lebel2
     banksel TRISC
     clrf TRISC; Настройка порта C на вывод
     banksel PORTC
lebel2
```

```
incf PORTC; Инкремент регистра порта C
goto label1

end
```

6. ОБОРУДОВАНИЕ

1. Компьютер IBM PC/AT с установленным MPLAB IDE.
2. Комплект MPLAB ICD.
3. Микроконтроллер PIC16F877.
4. Источник питания 9В, 0.2 А.

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем отличия Гарвардской и Фон-неймановской архитектур?
2. Какие виды памяти используются в PIC16F877? Какой их объем?
3. Поясните способ организации стека в PIC16F87x. Сколько уровней стека доступно при работе с MPLAB–ICD?
4. Перечислите регистры управления АЦП.
5. Сколько инструкций сделает за секунду контроллер PIC16F87x с подключенным кварцевым резонатором 10МГц. Какие программные средства в MPLAB IDE позволяют оценить время исполнения программы.
6. Сторожевой таймер WATCHDOG и его назначение.
7. Возможна ли постановка делителя перед WATCHDOG таймером? А после него?
8. В чем отличие директив Ассемблера от мнемоник команд?
9. Что такое симулятор и чем он отличается от внутрисхемного отладчика?
10. Как увидеть содержимое таймера в процессе выполнения программы?
11. Какие функции выполняет директива cblock используемая в программах 2 и 3?
12. В чем заключаются отличия CISC и RISC архитектур процессора?
13. Перечислите как можно больше способов записи шестнадцатеричных данных в среде MPLAB.
14. Можно ли подключить дополнительно внешнюю память программ к контроллеру PIC16F877 и как? А память данных?

Лабораторная работа №2. Модуль АЦП в микроконтроллере PIC16F877

1. ЦЕЛЬ РАБОТЫ

Изучение основ работы модуля АЦП в микроконтроллере PIC16F877, и составление программ с использованием АЦП. Получение практических навыков по работе с инструментальными средствами отладки микропроцессорных систем.

2. ВВЕДЕНИЕ

Отличительная особенность многих современных 8-разрядных микроконтроллеров — интегрированный на кристалле модуль многоканального аналого-цифрового преобразователя (АЦП). Модуль АЦП предназначен для ввода в МК аналоговых сигналов с датчиков физических величин и преобразования этих сигналов в двоичный код с целью последующей программной обработки. Структурная схема типового модуля АЦП представлена на рис. 13.

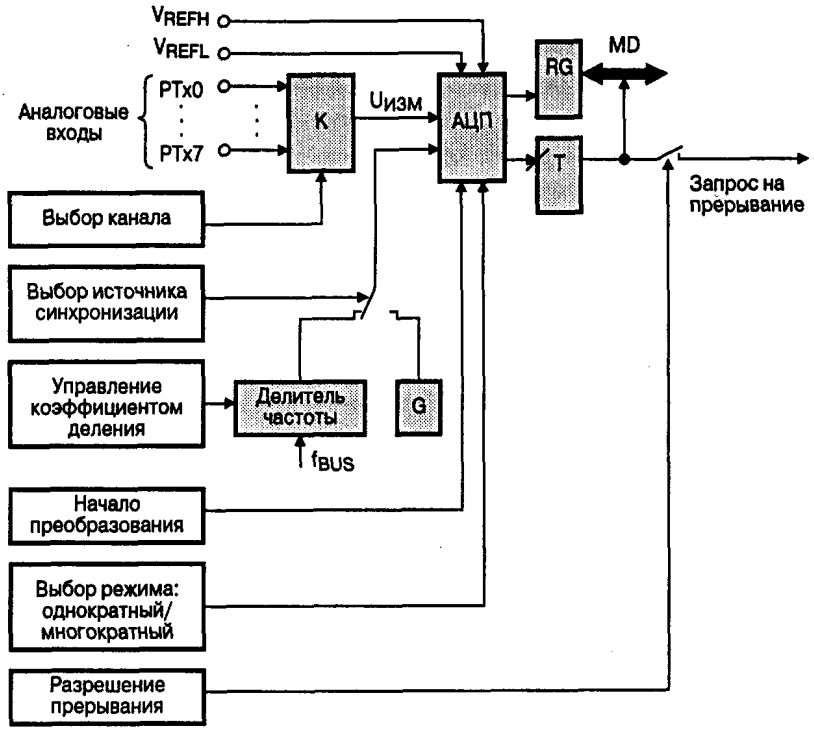


Рис. 13. Структура модуля АЦП.

Многоканальный аналоговый коммутатор служит для подключения одного из источников аналоговых сигналов (PTx0... PTx7)

к входу АЦП. Выбор источника сигнала для измерения осуществляется посредством записи номера канала коммутатора в соответствующие разряды регистра управления АЦП ADCON0. Заметим, что в модулях АЦП 8-разрядных МК предусмотрена только программная установка номера канала, режим последовательного автоматического сканирования каналов с записью результата измерения каждого канала в индивидуальную ячейку памяти не реализуется.

Диапазон измеряемых значений напряжения аналоговых входов определяется опорным напряжением $U_{оп}$. Разрешающая способность АЦП составляет $U_{оп}/ 2^n$, где n — число двоичных разрядов в слове результата. Для 10 разрядного АЦП и опорным напряжением 5В, разрешающая способность составит около 5 мВ. Попробуйте посчитать, какой будет результат преобразования АЦП, если на вход подать 4В.

Максимальное значение опорного напряжения, как правило, равно напряжению питания МК. Два вывода модуля АЦП используются для задания опорного напряжения: V_{REFH} — верхний предел $U_{оп}$, V_{REFL} — нижний предел. Разность потенциалов на входах V_{REF+} и V_{REF-} и составляет $U_{оп}$. Если измеряемое напряжение $U_{изм}$ => V_{REFH} , то результат преобразования будет равен $3FF, код $00 соответствует напряжениям $U_{изм}$<= V_{REFL}. Для достижения максимальной точности измерения следует выбрать максимально допустимое значение $U_{оп}$. В этом случае напряжение смещения нуля входного буфера и нелинейность передаточной характеристики АЦП будут вносить относительно малые погрешности.

В данной лабораторной работе изучается модуль АЦП микроконтроллера PIC16F877.

3. ПРОГРАММНАЯ МОДЕЛЬ PIC16F877

Микроконтроллер построен по Гарвардской архитектуре и имеет разделенную память программ и память данных.

На кристалле интегрировано 8К 14-битных слов FLASH памяти программ.

Память данных разбита на так называемые банки, каждый из которых имеет объем 128 байт. Выбор банков осуществляется переключением битов RP1 и RP0 (биты 6 и 5, регистра STATUS). Младшие ячейки каждого банка зарезервированы для регистров специальных функций, потом расположены универсальные регистры, выполненные как статическая память. Для уменьшения программы и более быстрого доступа некоторые часто используемые регистры

специальных функций расположенные в одном банке, могут быть отображены в другом банке.

Управление периферийными устройствами сводится к установке (сбросу) и опросу соответствующих битов регистров специальных функций. Карта памяти данных приведена в приложении 1. Для работы с регистрами специальных функций используются те же команды, что и для работы с универсальными регистрами. Система команд приведена в приложении 2.

Ниже будут описаны основные, необходимые для управления микроконтроллером, регистры.

Для наглядности состояния порта **C** микроконтроллера, на демонстрационной плате к нему включены светодиоды. Для настройки порта на ввод в регистр-защелку (для порта **C** это **TRISC**) нужно записать «1», и соответственно «0» - на вывод. Регистр TRISC находится в первом банке.

Если требуется осуществить косвенную адресацию, используют регистры косвенной адресации FSR и INDF. Для того чтобы прочитать ячейку памяти данных или записать в нее, нужно занести в регистр FSR адрес этой ячейки, а данные этой ячейки в регистр INDF. Следует отметить, что 7 бит регистра STATUS – IRP и 7 бит регистра FSR отвечают за выбор банка при косвенной адресации.

Управление прерываниями осуществляется записью в регистр INTCON соответствующих битов. Вектор прерываний находится по адресу 0004h памяти программ, и, следовательно, данный адрес необходимо зарезервировать для обработчика прерываний. Возврат из обработчика прерываний осуществляется командой RETFIE.

4. МОДУЛЬ АЦП

Модуль аналого-цифрового (A/D) преобразователя у микроконтроллера PIC16F877 имеет восемь аналоговых вводов.

A/D обеспечивает преобразование аналогового входного сигнала в соответствующий 10-разрядный цифровой код. A/D преобразование осуществляется методом последовательного приближения, на время преобразования уровень входного сигнала удерживается устройством выборки и хранения. Источник опорного напряжения задается программно. Внутренним источником является положительное напряжение питания устройства (V_{DD})

A/D преобразователь может работать, когда микроконтроллер находится в режиме SLEEP. В таком случае АЦП должен тактироваться

от внутреннего RC-генератора, т.к. основной частотозадающий генератор контроллера в SLEEP режиме отключается.

4.1. РЕГИСТРЫ УПРАВЛЕНИЯ АЦП.

АЦП семейства контроллеров PIC16F87x настраивается следующими регистрами: ADCON0 (таблица 10), ADCON1(таблица 11).

Результат же преобразования складывается в регистры ADRESH, ADRESL. Обратите внимание на расположение этих регистров в памяти.

Таблица 10.

Регистр **ADCON0** (адрес 1Fh)

№ бита	7	6	5	4	3	2	1	0
Имя бита	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/-DONE	—	ADON
Состояние после сброса	0	0	0	0	0	0	0	0
Доступность.	R/W	R/W	R/W	R/W	R/W	R/W	U	R/W
Обозначения: R – читаемый разряд; W – записываемый разряд; U – физически бит не существует, при обращении читается как 0.								

Назначение битов регистра ADCON0 приведено ниже.

Биты 7-6: **ADCS1:ADCS0** –выбор частоты преобразования

00= FOSC/2

01= FOSC/8

10= FOSC/32

11= F_{RC} (синхронизация от внутреннего RC генератора)

Бит 5-3: **CHS2:CHS0** – выбор аналогового канала

000 = канал 0, (RA0/AN0)

001 = канал 1, (RA1/AN1)

010 = канал 2, (RA2/AN2)

011 = канал 3, (RA3/AN3)

100 = канал 4, (RA5/AN4)

101 = канал 5, (RE0/AN5)

110 = канал 6, (RE1/AN6)

111 = канал 7, (RE2/AN7)

Бит 2: **GO/-DONE** –состояние A/D преобразования

Если ADON = 1

1= A/D выполняется преобразование (установка бита запускает A/D преобразование)

0= A/D преобразование окончено (автоматически сбрасывается аппаратным путем при окончании A/D преобразования)

Бит 1: зарезервированный бит, читается как “0”

Бит 0: **ADON** –включение модуля A/D

1=модуль включен

0= модуль выключен, и выходы преобразователя закрыты для снижения потребления.

Таблица 11

Регистр **ADCON1** (адрес 9Fh)

№ бита	7	6	5	4	3	2	1	0
Имя бита	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0
Состояние после сброса	0	0	0	0	0	0	0	0
Доступность.	R/W	U	U	U	R/W	R/W	R/W	R/W
Обозначения: R – читаемый разряд; W – записываемый разряд; U – физически бит не существует, при обращении читается как 0.								

Назначение битов регистра ADCON1 приведено ниже.

Бит 7: **ADFM** –формат записи результата A/D преобразования

1=правостороннее выравнивание, 6 старших значащих бита ADRESH читается как “0”

0= левостороннее выравнивание, 6 младших значащих бита ADRESH читается как “0”

Бит 6-4: зарезервированные биты, читаются как “0”

Бит 3-0: **PCFG3:PCFG0** –управление конфигурацией входов A/D

Таблица 12

Управляющие биты настройки каналов **PCFG3:PCFG0**

PCFG3: PCFG0	AN7 RE2	AN6 RE1	AN5 RE0	AN4 RA5	AN3 RA3	AN2	AN1	AN0	V_{REF+}	V_{REF-}
0000	A*	A	A	A	A	A	A	A	V_{DD}	V_{SS}
0001	A	A	A	A	V_{REF+}	A	A	A	RA3	V_{SS}
0010	D*	D	D	A	A	A	A	A	V_{DD}	V_{SS}
0011	D	D	D	A	V_{REF+}	A	A	A	RA3	V_{SS}
0100	D	D	D	D	A	D	A	A	V_{DD}	V_{SS}
0101	D	D	D	D	V_{REF+}	D	A	A	RA3	V_{SS}
011x	D	D	D	D	D	D	D	D	V_{DD}	V_{SS}
1000	A	A	A	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2
1001	D	D	A	A	A	A	A	A	V_{DD}	V_{SS}
1010	D	D	A	A	V_{REF+}	A	A	A	RA3	V_{SS}
1011	D	D	A	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2
1100	D	D	D	A	V_{REF+}	V_{REF-}	A	A	RA3	RA2
1101	D	D	D	D	V_{REF+}	V_{REF-}	A	A	RA3	RA2
1110	D	D	D	D	D	D	D	A	V_{DD}	V_{SS}
1111	D	D	D	D	V_{REF+}	V_{REF-}	D	A	RA3	RA2

A = аналоговый вход, D = цифровой канал ввода/вывода.

Регистр **ADCON0** используется для настройки АЦП, а с помощью регистра **ADCON1** выбирается одна из предложенных комбинаций настройки соответствующих портов для работы как цифровых и

аналоговых каналов и внешних или внутренних источников опорного напряжения АЦП (см. табл.12). Бит **ADFM** в регистре **ADCON1** дает прекрасную возможность выбора, как представить 10-ти разрядный результат преобразования в двух 8-ми разрядных регистрах. Когда преобразование закончилось, результат загружается в регистры **ADRESH** (старшие биты результата) и **ADRESL** (младшие биты результата). Вместе с этим сбрасывается бит **GO/DONE** в регистре **ADCON0<2>** и устанавливается флаг прерывания **ADIF** в регистре **PIR<6>**.

4.2. ВРЕМЕННЫЕ ТРЕБОВАНИЯ К ПОДКЛЮЧЕНИЮ КАНАЛА АЦП.

Для обеспечения необходимой точности преобразования, конденсатор C_{HOLD} должен успевать полностью заряжаться до уровня входного напряжения, т.к. именно с напряжением на емкости в дальнейшем будет работать АЦП. Схема аналогового входа показана на рисунке 14. Сопротивления R_S и R_{SS} непосредственно влияют на время зарядки конденсатора C_{HOLD}. Сопротивление канала ключа выборки R_{SS} зависит от напряжения питания (см. график на том же рисунке). Максимальное рекомендуемое значение внутреннего сопротивления источника аналогового сигнала – 10 кОм. После того как был выбран один из аналоговых каналов, перед преобразованием должно пройти определенное время, обозначаемое T_{ACQ}, которое складывается из времени задержки усилителя (около 2 мкс) и времени заряда конденсатора T_C. При температуре 25^0C и сопротивлении источника R_{SS} равном 10 кОм временная задержка составит около 17 мкс.

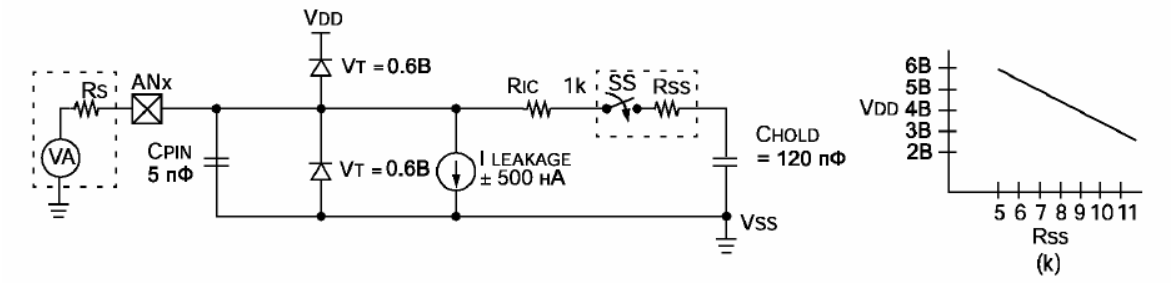


Рис. 14. Схема аналогового входа АЦП.

На схеме (рис. 14) приняты следующие обозначения:
C_{PIN} – паразитная емкость вывода;
V_T – пороговое напряжение;
$I_{LEAKAGE}$ – ток утечки вывода;
R_{IC} – сопротивление соединения;
SS – переключатель защелки;

C_{HOLD} – конденсатор защелки.

Ниже приводится порядок программирования АЦП:

1. Установить конфигурацию модуля АЦП, выбрав аналоговые входы и источник опорного напряжения (ADCON1), канал АЦП (ADCON0) и источник синхронизации АЦП (ADCON0).

2. Включить модуль АЦП (ADCON0<ADON>).

3. Если необходимо разрешить прерывание, установив биты GIE и PEIE и ADIE.

4. Сбросить бит ADIF.

5. Выдержать время, требуемое для устройства выборки и хранения.

6. Начать преобразование, установив бит GO/DONE (ADCON0)

7. Ожидать конца преобразования АЦП:

опрашивая бит GO/DONE (ADCON0);

ожидая прерывания АЦП;

опрашивая бит ADIF (PIR1).

8. Считать регистр результата преобразования АЦП (ADRES) и очистить бит ADIF, если необходимо.

5. МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Для написания программы в MPLAB необходимо сначала создать проект работы. Для этого, загрузив программу MPLAB, в верхнем меню Project>New Project нужно создать проект с оригинальным именем (Sasha.pjt). После чего в меню File>New File создать новый файл с таким же именем (Sasha.asm), сохранив его в меню File>Save As под таким именем.

ВНИМАНИЕ!

Файл Sasha.asm должен находиться в том же каталоге, что и Sasha.pjt.

Придумывая имя своему проекту, не забывайте, что вы, скорее всего, не один Саша (Маша, Гриша, Даша) на курсе, а работаете Вы все в одной сети, поэтому имя должно быть **ОРИГИНАЛЬНЫМ**.

Набрав программу, на ваш взгляд правильную, нужно в меню Project>Edit Project выбрать функцию Add Node и дополнить проект программой ассемблера Sasha.asm (ранее там уже находился Sasha.hex).

В этом же меню (Project>Edit project) для выбора режима работы программы MPLAB нужно выбрать функцию Change…, в которой выбрать тип микроконтроллера (в данном случае PIC16F877) и режим работы программы – MPLAB SIM Simulator.

Нажать кнопку F10 или выбрав в меню Project>Make Project и оттранслировать программу из языка Ассемблера в машинные коды.

Отладьте программу, используя F6,F7 и Ctrl+F9 для сброса программного счетчика, пошагового выполнения программы и анимационного выполнения программы.

Для понимания происходящего в программе удобно пользоваться картой специальных регистров (Window>Special Function Register), окном просмотра произвольных регистров (Window>Watch Window>New Watch Window).

Для генерирования сигналов на входах микроконтроллера используется меню Debug>Simulator Stimulus>Asynchronous Stimulus (например, для симуляции нажатия кнопки на входе PORTB<0>). Выбрав это меню, нужно щелкнуть правой кнопкой мыши на любом из 12 вариантов симуляции. В появившемся меню выбрать Assign Pin и далее «ножку» микроконтроллера, на которой будет происходить нужное вам явление (Pulse – 0->1->0 либо 1->0->1, Low – 0, High – 1, Toggle 0->1 либо 1->0)

Для работы с АЦП и симулирования результата преобразования можно использовать меню Window>Modify, в котором можно занести в регистр, хранящий результат преобразования АЦП, нужное вам значение.

После отладки программы в меню Project>Edit Project>Change изменить режим симуляции на режим MPLAB ICD Debugger и включить питание макетной платы, предварительно убедившись, что она у Вас есть и подключена к компьютеру.

В новом появившемся меню MPLAB ICD кнопка Program запишет Вашу программу в микроконтроллер.

Для работы микроконтроллера в режиме реального времени необходимо правой кнопкой мыши пометить (Break Point(s)) то место в программе, где нужно остановиться. После чего нажать F9.

Если программа в режиме симуляции после сброса (F6) оставляет данные в регистрах, полезным бывает сочетание кнопок Ctrl+Shift+F5, что эмулирует выключение питания. Для этой цели можно также воспользоваться Window>Modify и в появившемся меню в колонке Address задать начальный а в колонке End Address – конечный адрес области в которую можно записать любые данные (например, нули (00)) нажатием кнопки Write.

6. ОБОРУДОВАНИЕ

1. Компьютер IBM PC/AT с установленным продуктом MPLAB IDE.
2. Комплект MPLAB ICD.
3. Источник питания 9В, 0.2 А.

7. ВАРИАНТЫ ЗАДАНИЙ

1. Написать программу, выводящую средние 6 разрядов результата АЦП преобразования на порт **C**.
2. На всех светодиодах «отображать» содержимое младшего бита результата АЦП преобразования.
3. Зажечь светодиоды, при условии, что аналоговый сигнал больше половины опорного.
4. При превышении первого порога аналоговым сигналом зажечь 7-4 светодиоды, а при превышении второго порога – 3-0 светодиоды (пороги задаются преподавателем).
5. Вывести значение аналогового сигнала после нажатия кнопки и удерживать его 1 секунду.
6. Обеспечить мигание всех светодиодов разом, при превышении аналоговым сигналом порога, заданного преподавателем.
7. Занести двумя нажатиями кнопки два значения аналогового сигнала, и после второго нажатия обеспечить вывод этих двух значений, с периодом 1 секунду.
8. На светодиодах организовать светящуюся полосу, длина которой соответствует результату АЦП преобразования.

8. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие регистры, работающие с АЦП, вы знаете? Каково их назначение?
2. Какое количество памяти данных реализовано в PIC16F877?
3. Установка какого бита вызывает начало преобразования АЦП? Какие возможные способы для определения окончания преобразования АЦП вы знаете?
4. Как изменится время T_{ACQ}, если уменьшить внутреннее сопротивление источника аналогового сигнала? Пояснить, используя схему устройства выборки и хранения.
5. Может ли АЦП работать, когда микроконтроллер находится в «спящем режиме»? А при выключенном питании? Каким образом?

6. Сколько аналоговых каналов имеют микроконтроллеры семейства PIC16F87x. Какие регистры отвечают за выбор рабочего аналогового сигнала?

7. Существует ли возможность подключения внешней памяти программ к PIC16F877? А памяти данных?

8. Как получается, что 10-разрядный результат АЦП преобразования, помещается в 8-разрядный регистр?

9. Изменится ли время T_{ACQ} при изменении температуры кристалла и если да, то как?

9. СОДЕРЖАНИЕ ОТЧЕТА

1. Цель работы.
2. Вариант задания.
3. Программы для микроконтроллера PIC16F877 с комментариями.
4. Ответы на контрольные вопросы.
5. Выводы по проделанной работе.

Лабораторная работа №3. Таймеры микроконтроллера PIC16F877

1. ЦЕЛЬ РАБОТЫ

Изучение основ работы таймеров, решение различных задач с помощью таймеров. Получение практических навыков по работе с инструментальными средствами отладки микропроцессорных систем.

2. ВВЕДЕНИЕ

Большинство задач управления, которые возлагаются на микропроцессорную систему, должны выполняться в реальном времени. Можно выделить типовые задачи, которые должен решать микроконтроллер для эффективного управления:

- Отсчет равных интервалов времени заданной длительности, повтор алгоритма управления по истечении каждого такого интервала (формирование меток реального времени).
- Контроль над состоянием линии ввода.
- Измерение длительности сигнала заданного логического уровня на линии ввода.
- Подсчет числа импульсов внешнего сигнала на заданном временном интервале.
- Формирование на портах микроконтроллера сигнала заданного логического уровня с программируемой задержкой по отношению к изменению сигнала на линии ввода.
- Формирование на линии вывода импульсного сигнала с программируемой частотой и программируемым коэффициентом заполнения.

Каждая из перечисленных задач в отдельности может быть выполнена только программными средствами, без применения специальных аппаратных решений. Но у такого подхода имеется существенный недостаток: невозможность выполнения вычислений одновременно с отсчетом временного интервала. Поэтому в состав микроконтроллеров включаются специальные аппаратные средства, которые называют таймерами.

В составе PIC16F877 имеется три таймера, а так же два модуля захвата-сравнения-ШИМ. Отдельно следует так же отметить дополнительный – сторожевой таймер (WDT), который способен обеспечивать устойчивую работу системы.

3. ПРОГРАММНАЯ МОДЕЛЬ PIC16F877

Микроконтроллер PIC16F877 построен по Гарвардской архитектуре и имеет разделенную память программ и память данных.

На кристалле интегрировано 8К (8192) 14-битных слов FLASH памяти программ.

Память данных разбита на так называемые банки, каждый из которых имеет объем 128 байт. Выбор банков осуществляется переключением битов RP1 и RP0 (биты 6 и 5, регистра STATUS). Младшие ячейки каждого банка зарезервированы для регистров специальных функций, потом расположены универсальные регистры, выполненные как статическая память. Для уменьшения программы и более быстрого доступа некоторые часто используемые регистры специальных функций расположенные в одном банке, могут быть отображены в другом банке.

Управление периферийными устройствами сводится к установке (сбросу) и опросу соответствующих битов регистров специальных функций. Карта памяти данных приведена в приложении 1. Для работы с регистрами используются те же команды, что и для работы с ячейками ОЗУ. Система команд приведена в приложении 2.

Ниже будут описаны основные, необходимые для управления микроконтроллером, регистры.

На демонстрационной плате отладочного комплекта MPLAB ICD к порту С через переключатели подключена линейка светодиодов. Для настройки на вывод порта С в регистр-защелку TRISC нужно записать «0», и соответственно «1» - на ввод. Направлением обмена данными порта В управляет регистр TRISB, и т.д. Регистр TRISC находится в первом банке (прил. 1). При включении и сбросе регистры TRIS устанавливаются в «1», т.е. по умолчанию все порты настроены на ввод во избежание конфликта.

Если требуется осуществить косвенную адресацию, используют регистры косвенной адресации FSR и INDF. Для того чтобы прочитать ячейку памяти данных или записать в нее, нужно занести в регистр FSR адрес этой ячейки, а данные в регистр INDF. Следует отметить, что седьмой бит регистра STATUS – IRP и седьмой бит регистра FSR отвечают за выбор банка при косвенной адресации.

Управление прерываниями осуществляется записью в регистр INTCON соответствующих битов. Вектор прерываний находится по адресу 0004h памяти программ, следовательно, данный адрес необходимо зарезервировать для обработчика прерываний. Возврат из обработчика прерываний осуществляется командой RETFIE.

4. ТАЙМЕРЫ

4.1. МОДУЛЬ ТАЙМЕРА **TMR0**

Модуль таймера TMR0 – это простой 8-разрядный счетчик с переполнением (регистр TMR0). Источником синхронизации может быть внутренний генератор системы ($F_{OSC}/4$) или внешний (RA4/T0CKI). Когда источник синхронизации внешний, модуль таймера 0 может быть запрограммирован на счет по переднему и по заднему фронту. К модулю таймера 0 может быть подключен предделитель. Таймер 0 включается очисткой бита T0CS в регистре OPTION_REG<5>

Модуль таймера 0 генерирует сигнал прерывания T0IF (INTCON<1>)при переполнении (FF->00) и если прерывание для него разрешено, т.е. установлен бит T0IE в регистре INTCON<5> и установлен бит разрешения глобальных прерываний GIE (INTCON<7>), то произойдет прерывание, и контроллер начнет работать с адреса 04h. Выход из процедуры прерывания командой RETFIE.

Биты управления таймера TMR0 и сторожевого таймера содержатся в регистре OPTION_REG.

Таблица 13

Регистр **OPTION_REG** (адреса 81h, 181h)

№ бита	7	6	5	4	3	2	1	0
Имя бита	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
Состояние после сброса	1	1	1	1	1	1	1	1
Доступность.	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Обозначения: R – читаемый разряд, W – записываемый разряд.								

Назначение битов регистра OPTION_REG для работы с таймерами приведено ниже.

Бит 5: **TOCS**: выбор источника тактового сигнала для TIMER0.

1 = тактовый сигнал с входа RA4/T0CKI.

0 = внутренний источник тактового сигнала (CLKOUT).

Бит 4: **T0SE**: выбор фронта приращения TMR0 при внешнем тактовом сигнале

1 = приращение по заднему фронту сигнала на T0CKI.

0 = приращение по переднему фронту сигнала на T0CKI.

Бит 3: **PSA**: выбор включения предделителя.

1 = предделитель включен перед сторожевым таймером WDT.

0 = предделитель включен перед таймером TMR0.

Биты 2-0: **PS2 - PS0**; выбор коэффициента предделителя. Коэффициенты деления предделителя показаны в таблице 14.

Таблица 14

Коэффициенты деления в зависимости от значений PS2- PS0

Значение			TMR0 (PSA = 0)	WDT (PSA = 1)
PS2	PS1	PS0		
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

Установка коэффициента деления предделителя 1:1 для TMR0 соответствует переключению предделителя на сторожевой таймер.

4.2. МОДУЛЬ ТАЙМЕРА **TMR1**

Модуль таймера 1 – это 16-разрядный таймер/счетчик, состоящий из 2-х 8-разрядных регистров TMR1H и TMR1L, которые доступны для чтения и для записи. Прерывание (если оно разрешено) генерируется при переполнении (FFFF->0000), при этом устанавливается флажок TMR1IF(PIR1<0>). Это прерывание можно разрешить или замаскировать, используя бит маски прерывания TMR1IE (PIE1<0>).

За настройку модуля таймера TMR1 отвечает регистр T1CON (таблица 15).

Таблица 15

Регистр **T1CON:** регистр управления таймера1 (адрес 10h)

№ бита	7	6	5	4	3	2	1	0
Имя бита	–	–	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
Состояние после сброса	0	0	0	0	0	0	0	0
Доступность.	U	U	R/W	R/W	R/W	R/W	R/W	R/W
Обозначения: R – читаемый разряд; W – записываемый разряд; U – физически бит не существует, при обращении читается как 0.								

Назначение битов регистра T1CON приведено ниже.

Биты 5-4: **T1CKPS1:T1CKPS0** – коэффициент деления предделителя таймера1

11=1:8-значение предделителя

10=1:4-значение предделителя

01=1:2-значение предделителя

00=1:1-значение предделителя

Бит 3: **T1OSCEN** –работа таймера1 от кварцевого резонатора

1=работа от кварцевого резонатора разрешена

0= работа от кварцевого резонатора запрещена (инвертор кварцевого резонатора отключен)

Бит 2: $\overline{\text{T1SYNC}}$ –синхронизация внешней тактовой частоты

TMR1CS =1

1=асинхронный режим

0=синхронный режим

TMR1CS =0

Этот разряд игнорируется

Бит 1: **TMR1CS**- источник синхронизации таймера1

1=внешняя синхронизация (по переднему фронту на контакте RC0/T1OSO/T1CKI)

0=внутренняя синхронизация (FOSC/4)

Бит 0: **TMR1ON** –включение таймера1

1= таймер1 включен

0= таймер1 отключен

4.3. МОДУЛЬ ТАЙМЕРА **TMR2**

Модуль таймера 2 – это 8-разрядный счетчик с предделителем и постделителем. Тактирование таймера 2 осуществляется внутренней частотой ($F_{OSC}/4$) через предделитель, который может иметь коэффициент деления 1:1, 1:4 или 1:16. (T2CON<1:0>).

Модуль таймера 2 имеет 8-разрядный регистр периода PR2. Таймер 2 инкрементируется от 00h до значения, записанного в регистр PR2, после чего сбрасывается в 00h, генерируя сигнал прерывания, если оно разрешено

ВНИМАНИЕ! Прерывание генерируется с учетом коэффициента постделителя, т.е. если коэффициент пересчета постделителя 1:2, то сигнал о прерывании будет сгенерирован только после двух переполнений таймера 2.

В таблице 16 приведен регистр управления таймером 2.

Таблица 16

Регистр **T2CON**: регистр управления таймера2 (адрес 12h)

№ бита	7	6	5	4	3	2	1	0
Имя бита	–	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
Состояние после сброса	0	0	0	0	0	0	0	0
Доступность.	U	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Обозначения: R – читаемый разряд; W – записываемый разряд; U – физически бит не существует, при обращении читается как 0.								

Назначение битов регистра T2CON приведено ниже.

Биты 6-3: **TOUTPS3: TOUTPS0** – коэффициент деления постделителя

0000=1:1
0001=1:2
0010=1:3
……..
1111=1:16

Бит 2: **TMR2ON** – управление таймером2

1= таймер2 включен
0= таймер2 выключен

Биты1-0: **T2CKPS1:T2CKPS0** – коэффициент деления предделителя

00=1:1
01=1:4
1x=1:16

В лабораторной работе задания рассчитаны на работу с прерываниями от таймеров и с прерываниями от порта B (нажатие кнопки, PORTB<0>)

4.4. СТОРОЖЕВОЙ ТАЙМЕР **WDT**

Встроенный сторожевой таймер **WDT** работает от отдельного встроенного RC генератора и не требует внешних компонентов. Это позволяет работать таймеру при выключенном тактовом генераторе в спящем режиме **SLEEP** микроконтроллера. В нормальном режиме работы при переполнении сторожевого таймера происходит сброс микроконтроллера. Если контроллер находится в спящем режиме, переполнение таймера «разбудит» его с продолжением нормальной работы. Сторожевой таймер включен, если бит **WDTE** в слове конфигурации установлен.

Время переполнения зависит от температуры кристалла, напряжения питания и разброса технологических параметров микроконтроллера (см. раздел «электрические характеристики» в [1]). Если требуется большее время переполнения WDT, нужно программно подключить предделитель в регистре **OPTION_REG** с коэффициентом деления до 1:128.

5. МЕТОДИЧЕСКИЕ УКАЗАНИЯ

5.1. Для написания программы в MPLAB необходимо сначала создать проект работы. Для этого, загрузив программу MPLAB, в верхнем меню Project>New Project нужно создать проект с оригинальным именем (Sasha.pjt). После чего в меню File>New File

создать новый файл с таким же именем (Sasha.asm), сохранив его в меню File>Save As под таким именем.

ВНИМАНИЕ!

Файл Sasha.asm должен находиться в том же каталоге, что и Sasha.pjt.

Придумывая имя своему проекту не забывайте, что вы, скорее всего, не один Саша (Маша, Гриша, Даша) на курсе, а работаете Вы все в одной сети, поэтому имя должно быть **ОРИГИНАЛЬНЫМ**.

5.2. Набрав программу, на ваш взгляд правильную, нужно в меню Project>Edit Project выбрать функцию Add Node и дополнить проект программой ассемблера Sasha.asm (ранее там уже находился Sasha.hex)

5.3. В этом же меню (Project>Edit project) для выбора режима работы программы MPLAB нужно выбрать функцию Change…, в которой выбрать тип микроконтроллера (в данном случае PIC16F877) и режим работы программы – MPLAB SIM Simulator.

5.4. Нажать кнопку F10 или выбрав в меню Project>Make Project и оттранслировать программу из языка Ассемблера в машинные коды.

Отладьте программу, используя F6, F7 и Ctrl+F9 для сброса программного счетчика, пошагового выполнения программы и анимационного выполнения программы.

Для понимания происходящего в программе удобно пользоваться картой специальных регистров (Window>Special Function Register), окном просмотра произвольных регистров (Window>Watch Window>New Watch Window).

Для генерирования сигналов на входах микроконтроллера используется меню Debug>Simulator Stimulus>Asynchronous Stimulus (например, для симуляции нажатия кнопки на входе PORTB<0>). Выбрав это меню нужно щелкнуть правой кнопкой мыши на любом из 12 вариантов симуляции и в появившемся меню выбрать Assign Pin и далее «ножку» микроконтроллера, на которой будет происходить нужное вам явление (Pulse – 0->1->0 либо 1->0->1, Low – 0, High – 1, Toggle 0->1 либо 1-.>0)

Для работы с АЦП и симулирования результата преобразования можно использовать меню Window>Modify, в котором можно занести в регистр, хранящий результат преобразования АЦП, нужное вам значение.

После отладки программы в меню Project>Edit Project>Change изменить режим симуляции на режим MPLAB ICD Debugger и включить питание макетной платы, предварительно убедившись, что она у Вас есть и подключена к компьютеру.

В новом появившемся меню MPLAB ICD кнопка Program запишет Вашу программу в микроконтроллер.

Для работы микроконтроллера в режиме реального времени необходимо правой кнопкой мыши пометить (Break Point(s)) то место в программе, где нужно остановиться. После чего нажать F9.

Если программа в режиме симуляции после сброса (F6) оставляет данные в регистрах, полезным бывает сочетание кнопок Ctrl+Shist+F5, что эмулирует выключение питания. Для этой цели можно также воспользоваться Window>Modify и в появившемся меню в колонке Address задать начальный а в колонке End Address – конечный адрес области в которую можно записать любые данные (например, нули (00)) нажатием кнопки Write.

6. ОБОРУДОВАНИЕ

1. Компьютер IBM PC/AT с установленным MPLAB IDE.
2. Комплект MPLAB ICD.
3. Источник питания 9В, 0.2 А

7. ВАРИАНТЫ ЗАДАНИЙ

1. По нажатию кнопки зажигать светодиоды после переполнения таймера TMR0, а гасить их при отпускании.
2. Зажигать и гасить светодиоды с периодом 1 сек, используя таймер TMR1.
3. Обеспечить регулировку яркости свечения светодиодов, используя потенциометр и модуль АЦП.
4. Зажигать и гасить светодиоды с периодом 1 сек, используя таймер TMR2.
5. Нажатием кнопки запустить счет секунд, а отпусканием остановить этот счет. При этом выводить на светодиодах количество прошедших секунд.
6. Прерыванием, поступившим от таймера TMR0 запустить таймер TMR1, а прерыванием, поступившим от таймера TMR1 запустить таймер TMR2. При переполнении этого таймера зажечь все светодиоды.
7. Организовать бегущую тень (ноль) на светодиодах порта С с частотой 0,4 Гц.
8. На светодиодах порта сделать двоичный секундомер, который запускается, останавливается и сбрасывается кнопкой.
9. Организовать бегущую тень (ноль) на светодиодах порта С, с частотой задаваемой положением ручки потенциометра.

10. Обеспечить плавное включение и выключение любых пяти светодиодов при нажатии и отпускании кнопки, соответственно.

8. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое сторожевой таймер? Как можно его использовать?
2. Какое количество памяти программ установлено в PIC16F877?
3. Если предделитель таймера 2 настроен на значение 1:4, а постделитель – 1:2, когда произойдет прерывание от таймера 2?
4. Перечислите таймеры, входящие в состав PIC16F877 и назовите режимы их работы.
5. Какой регистр управляет таймером TMR2? Каким битом запускается таймер TMR0 ?
6. Какие отличия режима асинхронного счетчика от синхронного счетчика? Укажите достоинства и недостатки.
7. Какой максимальной длительности периоды можно формировать при помощи TMR0? Поясните ответ.
8. Изменится ли время переполнения таймера TMR1, если будет изменяться тактовая частота контроллера. Если да, то как можно стабилизировать это время?
9. Какой максимальной длительности периоды можно формировать при помощи TMR1? Поясните ответ.
10. Существуют ли ситуации, когда для корректной работы контроллера невозможно обойтись без таймера? Поясните примером.
11. Какой максимальной длительности периоды можно формировать при помощи TMR2? Поясните ответ.
12. Предложите алгоритм программы и схему устройства для измерения временных интервалов между внешними импульсами.

9. СОДЕРЖАНИЕ ОТЧЕТА

1. Цель работы.
2. Вариант задания.
3. Программы для микроконтроллера PIC16F877 с комментариями.
4. Ответы на контрольные вопросы.
5. Выводы о проделанной работе.

Список используемых источников

1. **PIC16F87X** 28/40-pin 8-Bit CMOS FLASH Microcontroller/ DS30292B Microchip Technology Inc 1999.

2. MPLAB ® ICD USER'S GUIDE / DS51184D Microchip Technology Inc., 2000.

3. В.А. Ульрих Микроконтроллеры PIC16C7X/Справочник по КМОП-микросхемам с АЦП. – М. Наука и техника, 2000, 254 с.

4. Ремизевич Т. В. Микроконтроллеры для встраиваемых приложений: от общих подходов – к семействам HC05 и HC08 фирмы Motorola./под ред. Кирюхина И. С. – М.: ДОДЭКА 2000, – 272с.

5. Современные микроконтроллеры: Архитектура, средства проектирования, примеры применения, ресурсы сети Интернет. Под ред. Коршуна И. В. – М.: Издательство Аким, 1998 – 272с.

Internet - ресурсы

6. http://www.microship.com – официальный сайт фирмы Microship.

7. http://www.microchip.ru – русскоязычный сайт фирмы Microship.

8. http://www.gamma.spb.ru – сайт фирмы «Гамма Санкт-Петербург».

9. http://www.gaw.ru – русскоязычный сайт по микропроцессорам.

Приложения

Приложение 1

Карта памяти данных контроллера PIC16F877

Банк 0		Банк 1		Банк 2		Банк 3	
Регистр	**Адрес**	**Регистр**	**Адрес**	**Регистр**	**Адрес**	**Регистр**	**Адрес**
INDF	00h	INDF	80h	INDF	100h	INDF	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	-	105h	-	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRIBSB	186h
PORTC	07h	TRISC	87h	-	107h	-	187h
PORTD	08h	TRISD	88h	-	108h	-	188h
PORTE	09h	TIRSE	89h	-	109h	-	189h
PCLATH	0Ah	CLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh		18Eh
TMR1H	0Fh	-	8Fh	EEADRH	10Fh	-	18Fh
T1CON	10h	-	90h		110h		190h
TMR2	11h	SSPCON2	91h				
T2CON	12h	PR2	92h				
SSBUF	13h	SSPADD	93h				
SSPCON	14h	SSPSTAT	94h				
CCPR1L	15h	-	95h				
CCPR1H	16h	-	96h				
CCP1CON	17h	-	97h	**Регистры общего назначения** (16 байт)		**Регистры общего назначения** (16 байт)	
RCSTA	18h	TXSTA	98h				
TXREG	19h	SPBRG	99h				
RCREG	1Ah	-	9Ah				
CCPR2L	1Bh	-	9Bh				
CCPR2H	1Ch	-	9Ch				
CCP2CON	1Dh	-	9Dh				
ADRESH	1Eh	ADRESL	9Eh				
ADCON0	1Fh	ADCON1	9Fh				
	20h		0A0h		120h		1A0h
Регистры общего назначения (96 байт)		**Регистры общего назначения** (80 байт)		**Регистры общего назначения** (80 байт)		**Регистры общего назначения** (80 байт)	
			0F0h		170h		1F0h
	7Fh	**Регистры, отображающие адреса 70h-7Fh банка 0**		**Регистры, отображающие адреса 70h-7Fh банка 0**		**Регистры, отображающие адреса 70h-7Fh банка 0**	

Приложение 2

Система команд PIC16F877

№	Мнемокод	Описание	Изм. флаги	Циклы	Прим.
1.	ADDWF f,d	Сложение W с f	C, DC, Z	1	1,2
2.	ANDWF f,d	Логическое И W и f		1	1,2
3.	CLRF f	Сброс регистра f	Z	1	2
4.	CLRW	Сброс регистра W	Z	1	
5.	COMF f,d	Инверсия регистра f	Z	1	1,2
6.	DECF f,d	Декремент регистра f	Z	1	1,2
7.	DECFSZ f,d	Декремент f и пропуск следующей команды, если результат декремента равен 0	—	1(2)	1,2,3
8.	INCF f,d	Инкремент регистра f	Z	1	1,2
9.	INCFSZ f,d	Инкремент f и пропуск следующей команды, если результат декремента равен 0	—	1(2)	1,2,3
10.	IORWF f,d	Логическое ИЛИ W и f	Z	1	1,2
11.	MOVF f,d	Пересылка регистра f	Z	1	1,2
12.	MOVWF f	Пересылка W в f		1	
13.	NOP	Холостая команда	—	1	
14.	RLF f,d	Сдвиг f влево через перенос	C	1	1,2
15.	RRF f,d	Сдвиг f вправо через перенос	C	1	1,2
16.	SUBWF f,d	Вычитание W из f	C, DC, Z	1	1,2
17.	SWAPF f,d	Обмен местами тетрад в f	—	1	1,2
18.	XORWF f,d	Исключающее ИЛИ W и f	Z	1	1,2
19.	BCF f,b	Сброс бита b в регистре f	—	1	1,2
20.	BSF f,b	Установка бита b в регистре f	—	1	1,2
21.	BTFSC f,b	Пропустить следующую команду, если бит b, в регистре f равен нулю	—	1(2)	3
22.	BTFSS f,b	Пропустить следующую команду, если бит b, в регистре f равен единице	—	1(2)	3
23.	ADDLW k	Сложение константы с W.	C, DC,Z	1	
24.	ANDLW k	Логическое И константы и W	Z	1	
25.	CALL k	Вызов подпрограммы	—	2	
26.	CLRWDT	Сброс сторожевого таймера (WDT)	TO,PD	1	
27.	GOTO k	Переход по адресу k	—	2	
28.	IORLW k	Логическое ИЛИ константы и W	Z	1	
29.	MOVLW k	Пересылка константы в W	—	1	
30.	RETFIE	Возврат из прерывания.	—	2	
31.	RETLW k	Возврат из подпрограммы с загрузкой константы в W	—	2	
32.	RETURN	Возврат из подпрограммы.	—	2	
33.	SLEEP	Переход в режим SLEEP	TO,PD	1	
34.	SUBLW k	Вычитание W из константы.	C, DC,Z	1	
35.	XORLW k	Исключающее ИЛИ константы и W	Z	1	

Описание полей кода операции

Символ	Описание
f	Адрес регистра (0x00-0x7F) – файл
W	Рабочий регистр
b	Номер бита в 8-ми разрядном регистре
k	Константа
D	Регистр назначения: d=0 - результат в регистре W d=1 - результат в регистре f По умолчанию d=1

Примечания:

1. Когда модифицируется регистр порта ввода/вывода, (например MOVF PORTB,1), значение считывается непосредственно с ножек микросхемы. Например, если в регистре порта «1», а контакты конфигурированы как входы и внешнее устройство установит низкий уровень, то в регистр данных будут записаны «0».

2. Если команда выполняется над регистром TMR0 (когда d=1, результат записывается в регистр таймера 0), то предделитель, будет обнулен.

3. Если счетчик программ (PC) изменяется или результат проверки условия истинен, то команда выполняется за два цикла. Во втором цикле выполняется команда NOP.

Приложение 3

Краткие сведения об ассемблере MPASM

Формат представления данных.

В MPASM возможно представление данных не только в численном виде, но и в форме выражений. Ниже приведены некоторые элементы, которые (кроме целочисленных констант и символов) могут быть в выражениях:

* умножение, / деление, + прибавить, - вычесть, << сдвиг влево,

>> сдвиг вправо, = равно, <> не равно.

Числа задаются в следующем формате. (Для примера взято число $162_{(10)} = A2_{(16)} = 242_{(8)} = 10100010_{(2)}$).

Двоичные (Binary) числа: 0b<цифры>, b'<цифры>'.

Пример: 0b10100010 или b'10100010' – эти записи эквивалентны.

Восьмеричные (Octal) числа: <цифры>o, o'<цифры>' (признак восьмеричного числа - буква o, а не цифра ноль).

Пример: o'242' или 242o – эти записи эквивалентны.

Десятичные (Decimal) числа: .<цифры> либо d'<цифры>'

Пример: .162, или d'162' – эти записи эквивалентны.

Шестнадцатеричные (Hexadecimal) числа: 0x<цифры>, h'<цифры>' или <цифры>h (в последней форме записи число должно начинаться с 0..9)

Пример: 0xA2, или h'A2', или 0A2h – все эти записи эквивалентны.

ASCII коды: a'<цифры>', или '<цифры>'.

Пример: a'B', или 'B' – в обоих случаях Ассемблер сгенерирует ASCII код буквы B.

Основные директивы Ассемблера.

Кроме команд процессора MPASM, как и другие языки Ассемблера позволяет использование специальных управляющих слов – директив. Ниже дано описание и формат представления некоторых **директив** MPASM.

`<label> equ <expr>` эта директива определяет константу;

Пример:

```
four equ 4; присваивает имени four значение 4.
```

`<label> set <expr>` эта директива определяет переменную (подобна `equ`, но может быть переопределена другой директивой).

Пример:

```
width  set 0x12
```

```
length set 0x14
area   set length * width
```

[<label>] **org** <address> расположить программу с адреса address;

Пример:

```
int_1 org 0x20 ;Метка int_1 организована с адреса 20h
      nop    ;Начало подпрограммы
```

include <<filename>> или include "<include_file>" - подключает другой исходный файл, (включение может быть вложенным)

Пример:

```
include "c:\sys\sysdefs.inc" ;подключить ;системные определения
include <regs.h>; подключить описания регистров
```

list [<list_option>, ..., <list_option>] директива позволяющая управлять листингом. Запускается с ключами, мы рассмотрим только ключ **p** – тип процессора

Пример:

```
list p=16f877
```

banksel <const> – по этой директиве Ассемблер генерирует команды выбора банка, в которой находится предварительно определенная константа - const.

Пример:

```
banksel  TRISA
movwf    TRISA
```

end - конец ассемблерной программы, все, что после этой директивы будет игнорировано.

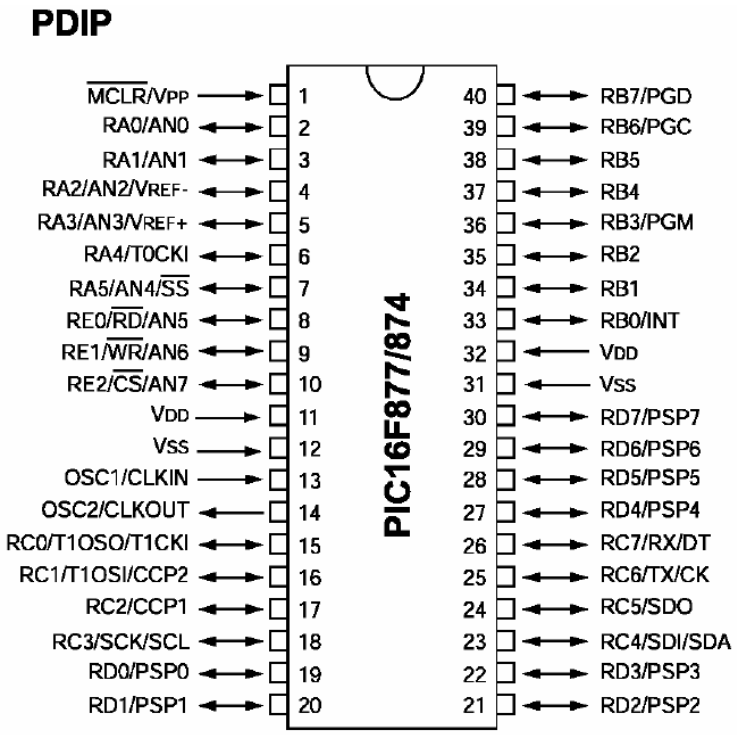
Пример:

```
list p=16f877
 :      ; выполняемый код
 :      ;
end     ; Конец команд
```

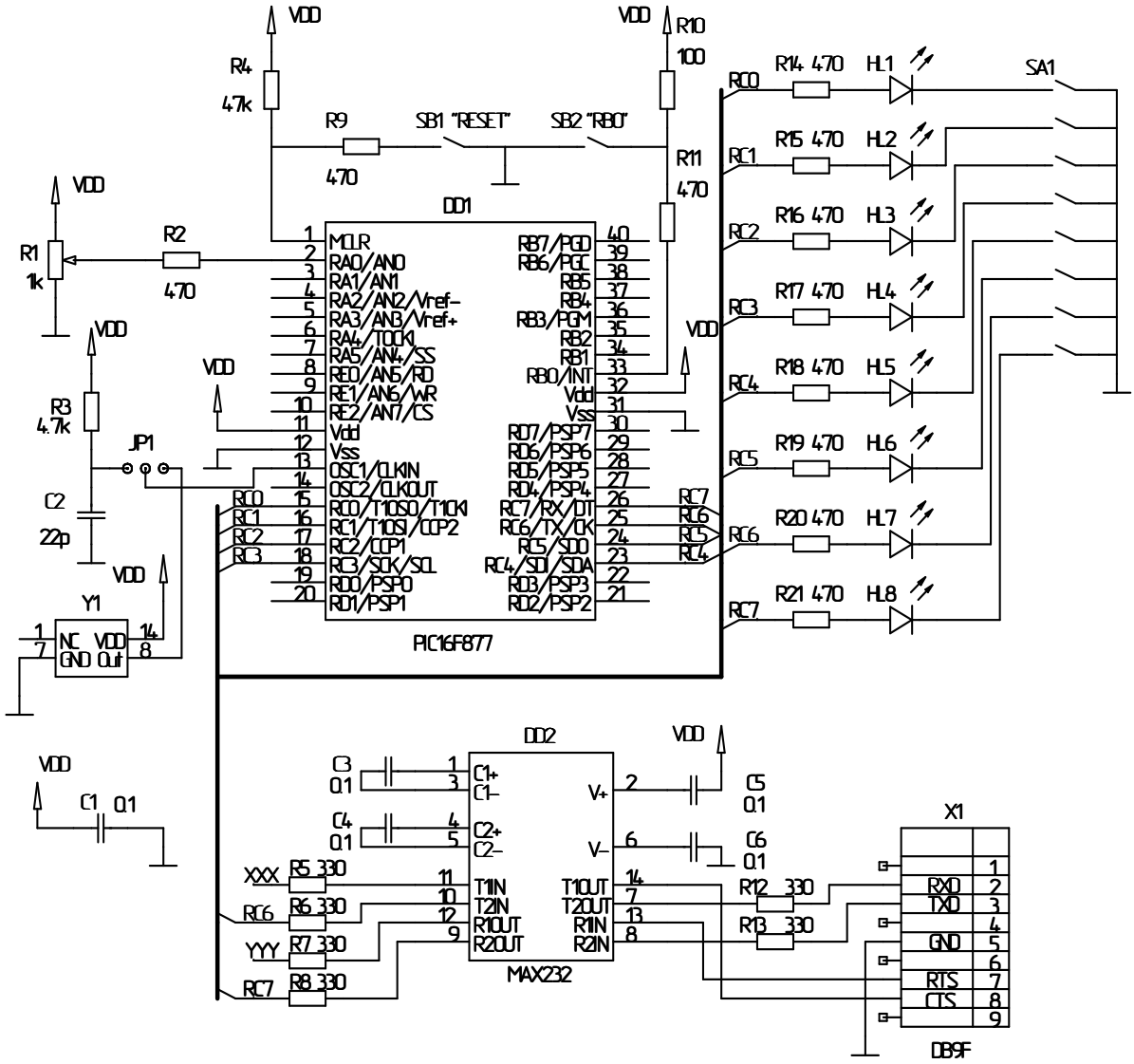
Мы рассмотрели только самый минимум директив Ассемблера. MPASM предоставляет разработчику очень обширный набор управляющих команд. Более подробно с ними Вам поможет познакомиться MPLAB ® ICD USER'S GUIDE, используйте так же интерактивную помощь программы.

Приложение 4

Назначение выводов PIC16F877



Фрагменты принципиальной схемы демонстрационной платы



Содержание

Галина Степановна Воробьева
Д.Н. Добровольский
А.В. Крыцкий
Денис Владимирович Пайгин
Дмитрий Александрович Пестунов

PIC-контроллеры

Подписано к печати 23.03.2004.
Формат 60x84/16. Бумага офсетная. Печать RISO. Усл. печ. л. 3.49 Уч.-изд. л. 3,16. 100 экз. Заказ . Цена свободная. Издательство ТПУ. 634050, Томск, пр. Ленина, 30.