Министерство образования и науки Российской Федерации
Национальный исследовательский университет «МИЭТ»

Алексеев А.А., Владимиров Л.Л., Гуреев П.В., Левицкий Д.О., Тогидный О.Б., Шумилин С.С.

Практикум для лабораторных работ по курсу «Программирование микроконтроллеров»

Утверждено редакционно-издательским советом университета

Под редакцией Д.О.Левицкого

Москва 2015

УДК 004.31(076)

Рецензент руководитель направления по внедрению и развитию образовательных программ «ПКК Миландр» *М.В. Кузнецов*.

Алексеев А.А., Владимиров Л.Л., Гуреев П.В., Левицкий Д.О., Тогидный О.Б., Шумилин С.С.

Практикум для лабораторных работ по курсу «Программирование микроконтроллеров». – М.: МИЭТ, 2015. – 48 с.: ил.

Практикум содержит описания лабораторных работ по дисциплине «Программирование микроконтроллеров» на основе 8-разрядного микроконтроллера К1886ВЕ61У, разработанного и производимого компанией «ПКК Миландр» (г. Москва, г. Зеленоград). Рассмотрены следующие темы: основные понятия процесса разработки, внутрисхемной отладки программ и программирования внутренней памяти микроконтроллеров К1886ВЕ61У; синтез простейших прошивок этих микроконтроллеров с использованием языка Assembler; ввод значений на индикаторы отладочной платы Eval17; использование встроенного таймера в микроконтроллере К1886ВЕ61У. Учтено, что эта дисциплина изучается после общих курсов «Организация ЭВМ и микропроцессорных систем», «Операционные системы».

Практикум не является исчерпывающим по данной дисциплине, а есть лишь опорная точка для студента и преподавателя на момент написания. И студенту, и преподавателю нужно творчески подходить к дисциплине, искать и находить новые источники знаний. Любая лабораторная работа с течением времени может меняться, модифицируясь как в части теории, так и практики, путем внедрения новых работ и удаления устаревших.

Для студентов бакалавриата и магистратуры изучающих программирование и применение микроконтроллеров.

Лабораторная работа № 1

Основные понятия процесса разработки, внутрисхемной отладки программ и программирования внутренней памяти микроконтроллеров К1886ВЕ61У

Цель работы: получение начальных сведений о работе с интегрированной средой разработки, внутрисхемной отладке программ и программировании внутренней памяти микроконтроллеров К1886ВЕ61У; ознакомление с интерфейсом графической оболочки DEVCPP.EXE компании «ПКК Миландр»; ознакомление со структурой директорий среды разработки; настройка параметров среды разработки и запуск тестовых примеров на языке Assembler.

Оборудование: персональный компьютер (ПК); отладочная плата Eval17 на основе микроконтроллеров К1886ВЕ61У компании «ПКК Миландр».

Программное обеспечение: исполняемый модуль DEVCPP.EXE, разработанный компанией «ПКК Миландр»; исполняемый модуль для компиляции программ на языке Assembler MPASMWIN.EXE из пакета MPLAB IDE фирмы MICROCHIP; драйвер giveio.sys, разработанный компанией «ПКК Миландр», для работы с LPT-портом под ОС Windows XP и программа установки драйвера LOADDRV.EXE (при работе с USB-программатором установка драйвера не требуется).

Программное и аппаратное обеспечение предоставлено компанией «ПКК Миландр»: http://milandr.ru/index.php?page=programmnoe-obespech

Продолжительность работы – 4 ч.

Теоретические сведения

Общие положения

Отладочный комплект для микроконтроллера К1886ВЕ61У предназначен для демонстрации функционирования данных

микроконтроллеров и их основных периферийных модулей, для начального обучения программированию микроконтроллеров с помощью прилагаемой демонстрационной программы; а также для отладки собственных проектов с применением установленных на плате блоков и возможностью макетирования дополнительной схемы на монтажном поле платы. Выводы микроконтроллера, используемые в собственных проектах, отсоединяются с помощью легко удаляемых перемычек. Программирование памяти микроконтроллера К1886ВЕ61У осуществляется с помощью внутрисхемного программатора.

Для демонстрации функционирования плата Eval17 подключается к COM порту ПК или к интерфейсу RS-232 дополнительного внешнего устройства, например к аналогичной демонстрационно-отладочной плате Eval17. Подключение производится с помощью прилагаемого нуль-модемного кабеля. Питание платы осуществляется от адаптера стабилизированного напряжения 5 В (для варианта платы без блока LIN интерфейса) или от адаптера 12 В (для варианта с блоком LIN интерфейса).

Для демонстрации функционирования используется прилагаемое программное обеспечение: демонстрационная программа, прошиваемая в память микроконтроллера К1886ВЕ61У, и демонстрационная программа для ПК.

Состав отладочного комплекта для микроконтроллера К1886ВЕ61У

Набор для отладки микроконтроллера К1886ВЕ61У имеет следующую комплектацию:

- отладочная плата Eval17;
- микроконтроллер К1886ВЕ61У;
- внутрисхемный программатор;
- кабель RS-232/RS-232;
- блок питания (5 В) для отладочной платы;
- руководство по эксплуатации микроконтроллеров;
- техническое описание демонстрационно-отладочных плат;
- программное обеспечение для работы с демонстрационно-отладочными платами.

Описание отладочной платы Eval17

На демонстрационно-отладочной плате Eval17 (рис.1) установлены следующие блоки и компоненты:

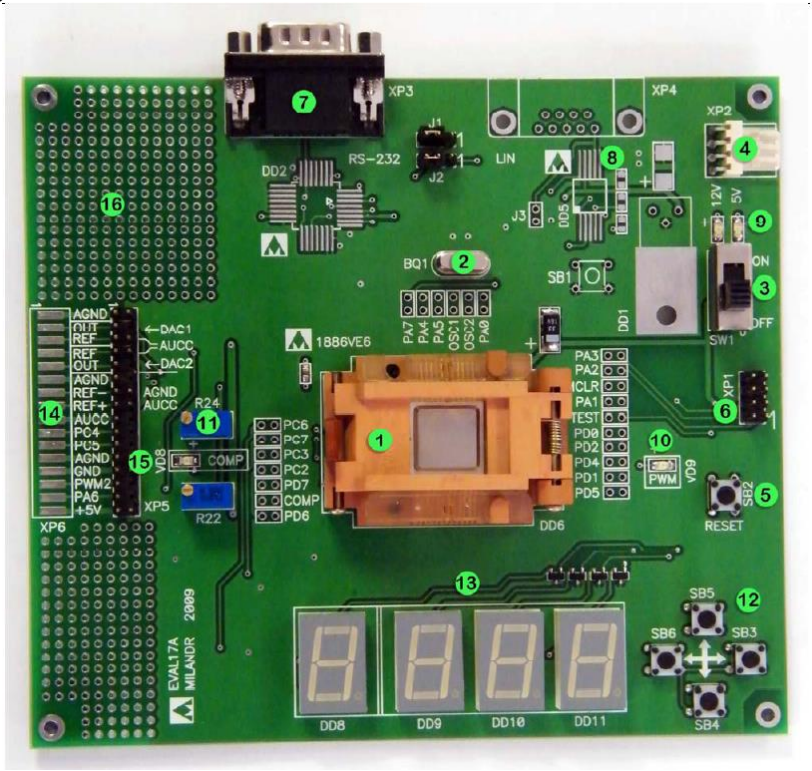


Рис.1. Демонстрационно-отладочная плата Eval17

1 – контактирующее устройство для установки микроконтроллера К1886ВЕ61У в спутнике. Набор легко удаляемых перемычек, обеспечивающих возможность отключения от схемы выводов микроконтроллера (за исключением выводов питания);

2 – кварцевый резонатор (16 МГц) для тактирования микроконтроллера. BQ1 подключен к выводам OSC1 и OSC2 микроконтроллера и определяет его тактовую частоту;

3 – переключатель SW1, обеспечивающий включение платы и уход в режим программирования. Плата Eval17 питается от адаптера с

напряжением 5 В. Адаптер подключается к разъему XP2. Для включения платы надо перевести переключатель SW1 в положение «ON». Необходимо обратить внимание на наличие раздельных аналоговых и цифровых линий питания и «земли». Это требуется для снижения влияния цифровых помех на аналоговые сигналы для АЦП и ЦАП;

4 – разъем XP2 для подключения внешнего источника питания: 5 В.

5 – схема сброса микроконтроллера. Сброс микроконтроллера осуществляется клавишей SB2 «RESET»;

6 – схема для подключения внутрисхемного программатора XP1. При программировании памяти микроконтроллера переключатель SW1 должен находиться в положении «OFF». Для программирования применяется внутрисхемный программатор для микроконтроллеров серии 1886, формирующий сигналы MCLR и TEST напряжением 12 В. Схема подключения преобразует уровни этих сигналов до значений логических уровней. Программатор формирует напряжение питания только для программируемого микроконтроллера, поэтому на плате в режиме программирования предусмотрено разделение линий питания всей платы и программируемого микроконтроллера;

7 – блок интерфейса RS-232, реализованный на микросхеме 5559ИН4У, производства компании «ПКК Миландр». Для работы модуля USART микроконтроллера К1886ВЕ61У с интерфейсом RS-232 необходимо установить перемычки J1 и J2 в положение 2-3;

8 – блок LIN интерфейса, также реализованный на микросхеме 5559ИН15, производства компании «ПКК Миландр». Блок содержит конфигурационную перемычку J3 для включения питания линии LIN интерфейса (для режима ведущего) и клавишу SB1 для локального включения по LIN интерфейсу. Питание блока LIN интерфейса осуществляется от адаптера с напряжением 12 В. Для питания остальной части схемы от этого адаптера блок содержит микросхему DD1 для формирования напряжения 5 В. Для работы модуля USART микроконтроллера К1886ВЕ61У с LIN интерфейсом необходимо установить перемычки J1 и J2 в положение 1-2. (В стандартном варианте комплектации схема блока LIN не монтируется.);

9 – схема индикации включения питающих напряжений 5 В и 12 В. Индикация напряжения питания 5 В осуществляется при переключении SW1 в положение «ON». Индикация напряжения питания 12 В осуществляется при включении адаптера питания на 12 В;

10 – индикация состояния ШИМ-выхода PWM1 с помощью светодиода VD9, показывающая состояние выхода ШИМ PWM1 для визуальной оценки скважности сигнала. Максимальная яркость свечения соответствует скважности импульса, равной 1;

11 – многооборотные подстроечные резисторы R22 и R24, задающие входное напряжение для блоков АЦП (PC2/ADC2 и PC3/ADC3). Значение напряжения на выводах резисторов регулируются винтом в диапазоне от AGND до AUcc;

12 – клавиатура (клавиши SB3 – SB6). Клавиатура состоит из четырех клавиш и служит для управления демонстрационными программами. Выбор сканируемой клавиши определяется портами PD0, PD1, PD4 или PD5, состояние клавиши считывается с порта PC7;

13 – схема индикации из четырех 7+1 сегментных индикаторов DD8 – DD11 (светодиодные матрицы с общим катодом). Разряды индикатора (катоды) выбираются портами PD0, PD1, PD4 и PD5. Сегменты (аноды) через токоограничивающие резисторы подключены к сдвиговому регистру DD7, загружаемому с помощью портов PC6 и PC7;

14 – набор универсальных разъемов XP5 и XP6 для подключения измерительных приборов и подачи внешних аналоговых сигналов на периферийные модули микроконтроллера. На разъемы выведены сигналы:

- выход OUT и опорное напряжение REF для модулей ЦАП (DAC1 и DAC2),
- опорные напряжения REF- и REF+ для модуля АЦП,
- два входа модуля АЦП (PC4/ADC4 и PC5/ADC5),
- выход ШИМ (PD3/PWM2),
- цифровой вывод (PA6),
- аналоговые и цифровые питание и «земля» (AUcc, Ucc, AGND, GND);

15 – индикация состояния выхода аналогового компаратора (светодиод VD8). Свечение светодиода соответствует логической единице;

16 – монтажное поле для макетирования и отладки собственных проектов.

Схематичное устройство платы

На рис.2 показано схематичное устройство платы.

Регистр порта A, D, C. Порты A, D, C – это 8-разрядные

двунаправленные порты ввода-вывода. Направление данных управляется регистром направления DDRA, DDRD, DDRC соответственно. Значение «1» в регистрах DDRA, DDRD, DDRC конфигурирует соответствующие выводы порта как вход. Значение «0» – как выход. Выводы портов A, D и C мультиплексированы с аналоговыми входами АЦП.

Таймер 0 1 2 – таймеры-счетчики разного назначения.

АЦП – аналого-цифровой преобразователь. Преобразует аналоговый сигнал в цифровой. Например, при подаче на преобразователь напряжения, на выходе получаем бинарное число, эквивалентное величине подаваемого напряжения.

ЦАП – цифроаналоговый преобразователь. Преобразует цифровой сигнал в аналоговый. Например, на преобразователь посылается сигнал в виде двоичного числа, на выходе устанавливается напряжение, величина которого равна заданному числу.

Контроллер прерываний – микросхема или встроенный блок процессора. Отвечает за возможность последовательной обработки запросов на прерывание от разных устройств.

Процессорное RISC-ядро – выполняет операции над данными (арифметическими, логическими и т.д.), а также управляет другими устройствами на плате путем выполнения пользовательской программы.

ПЗУ – постоянное запоминающее устройство. Такое устройство является энергонезависимым, т.е. данные в нем хранятся вне зависимости от наличия питания.

ОЗУ – оперативное запоминающее устройство. Позволяет сохранять данные только при включенном питании. Память в ОЗУ хранится в виде массива данных.

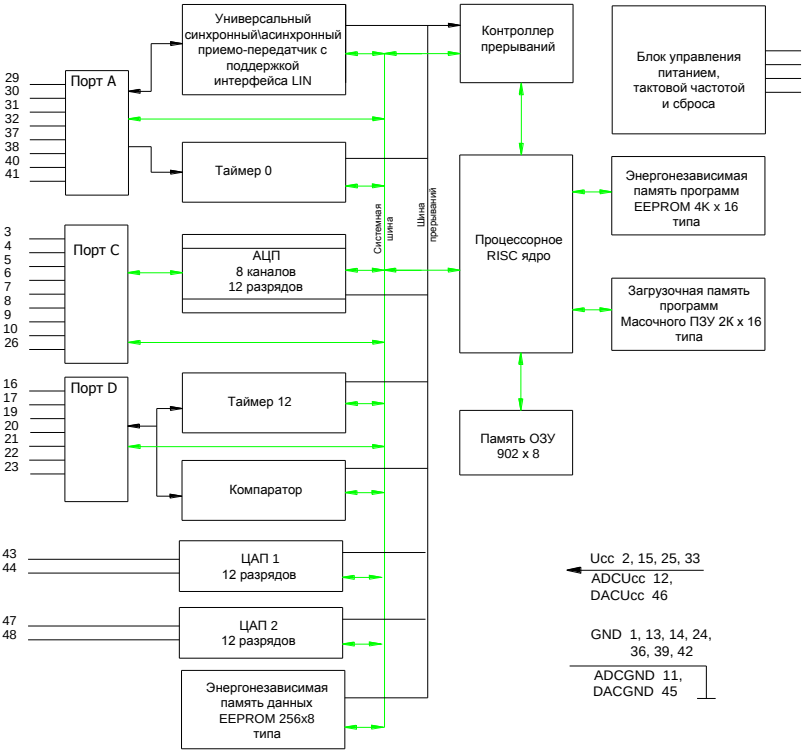


Рис.2. Схематичное устройство платы

Настройка блоков периферии

Под термином «блок периферии» будем иметь в виду блок, реализующий определенные функции. Общее количество блоков и режимов соответственно равно восьми. В одном режиме может отображаться одно значение, состоящее из трех 16-ричных цифр.

Ввод новых значений данных и просмотр результатов оцифровки сигналов осуществляется с помощью установленных индикаторов и клавиатуры демоплаты.

На индикаторе отображаются:

– номер режима (индикатор DD8);

– значение данных для данного канала в 16-ричном коде (индикаторы DD9, DD10 и DD11).

На рис.3 представлены четыре кнопки (SB5, SB3, SB5, SB6) для ввода значений на цифровое табло. SB5 увеличивает значение на индикаторе, SB4 уменьшает значение, SB6 и SB3 предназначены для выбора индикатора.

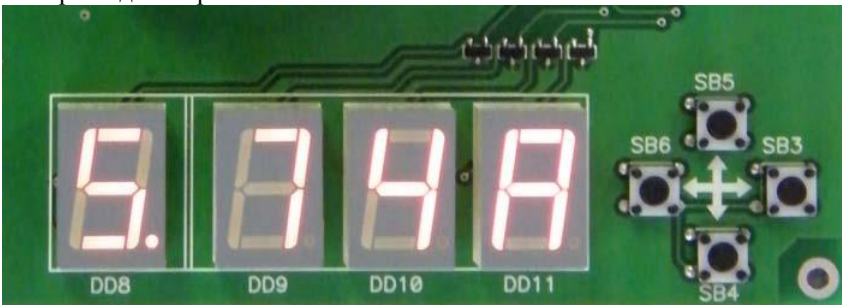


Рис.3. Световое табло и клавиатура

Соответствие номеров канала и модулей микроконтроллера приведено в табл.1.

Таблица 1

Соответствие режимов и модулей

Номер режима / блока	Модуль	Изменение значения
1	АЦП2	Только индикация
2	АЦП3	Только индикация
3	АЦП4	Только индикация
4	АЦП5	Только индикация
5	PWM1	Разрешен ввод
6	PWM2	Разрешен ввод
7	DAC1	Разрешен ввод
8	DAC2	Разрешен ввод

Многооборотные подстроечные резисторы R22 и R24

Резисторы R22 и R24, показанные на рис.4, задают входное напряжение для блоков 1, 2, 3 и 4. Значение напряжения на выводе резисторов регулируется винтом в диапазоне от AGND до AUcc.

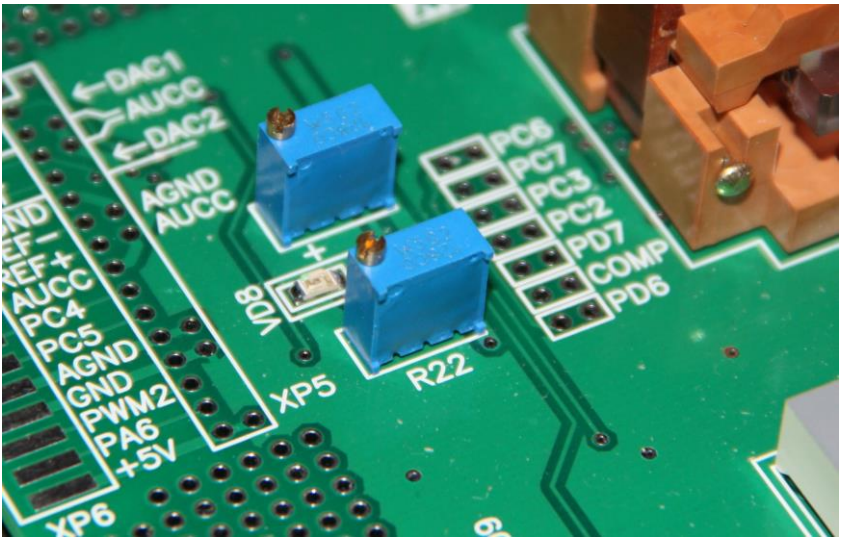


Рис.4. Подстроечные резисторы

Структура проекта для процесса разработки

Перед началом работы со средствами разработки, внутрисхемной отладки программ и программирования внутренней памяти микроконтроллеров К1886ВЕ61У надо создать папку проекта и расположить в ней следующие файлы, необходимые для успешной компиляции:

- MPASMWIN.EXE;
- 1886VE6M.INC.

В процессе компиляции автоматически будут созданы файлы:

- Project1.dev;
- main.asm;
- main.cod;
- main.err;
- Makefile.bat;
- main.lst;
- main.bin;
- main.hex.

Примечание 1. Наименование «main» является лишь именем

исходного файла типа «.asm» и может быть иным.

Примечание 2. main.bin и main.hex – файлы, в которых хранятся бинарный и 16-ричный коды соответственно, инструкции, загружаемые в память микроконтроллера.

Лабораторное задание

1. Загрузить тестовую программу на микроконтроллер.
2. Выполнить этап трансляции описания, получив файлы .bin, .hex.
3. Убедиться, что на индикаторе, размещенном на плате, высветились значения каналов
4. Проверить все каналы.

Порядок выполнения работы

1. Создание проекта и его структуры.

Запустите приложение DEVCPP.EXE **от имени администратора** (ПКМ – Запуск от имени администратора).

Создайте новый проект: **File** – **New** – **Project**. В качестве компилятора выберите **MPASM**. Укажите имя проекта и место его хранения.

Примечание. Длина адреса не должна превышать 62 символов, адрес должен состоять только из латинских букв.

Скопируйте в папку, где хранится проект, компилятор MPASMWIN.EXE и заголовочный файл 1886VE6M.INC.

Откройте **Project** – **Project Options** – **Compilers** и в первом списке выберите **MPASM**. Выберите генерацию инструкций для микроконтроллера **1886BE6**. Закройте окно **Project Options** нажатием OK.

Откройте **Tools** – **Compiler Options**. Во второй строке выберите адрес хранения компилятора MPASMWIN.EXE. Закройте окно **Compiler Options** нажатием OK.

2. Подготовка файла на языке Assembler.

Для подготовки файла на языке Assembler скопируйте в окно проекта текст из тестовой программы, находящейся по адресу:

D:\LW\IDE1886\Examples\Eval17_VE6\Project1.asm

3. Компиляция.

Проект может быть скомпилирован через меню **Execute – Compile** или с помощью сочетания клавиш Ctrl+F9 либо через значок **Compile** на панели быстрого доступа. В директории проекта должны появиться файлы с расширениями .bin и .hex.

При возникновении ошибки «все права защищены» следует проверить директории хранения компиляторов.

4. Подключение программатора к плате.

Внимание! Переключатель SW1 должен находиться в положении «OFF» до того, как USB-программатор будет подключен, в противном случае возможен выход микросхемы из строя.

Подключите плату к ПК с помощью USB-программатора. При этом красный (розовый) провод на шине USB-программатора должен соединиться с выводом №1 разъема XP1, как показано на рис.5,а.

Обратите внимание! На рис.5,б,в показаны примеры некорректного подключения разъема программатора: на рис.5,б красный провод «висит» слева от разъема, а на рис.5,в красный провод «висит» справа от разъема.

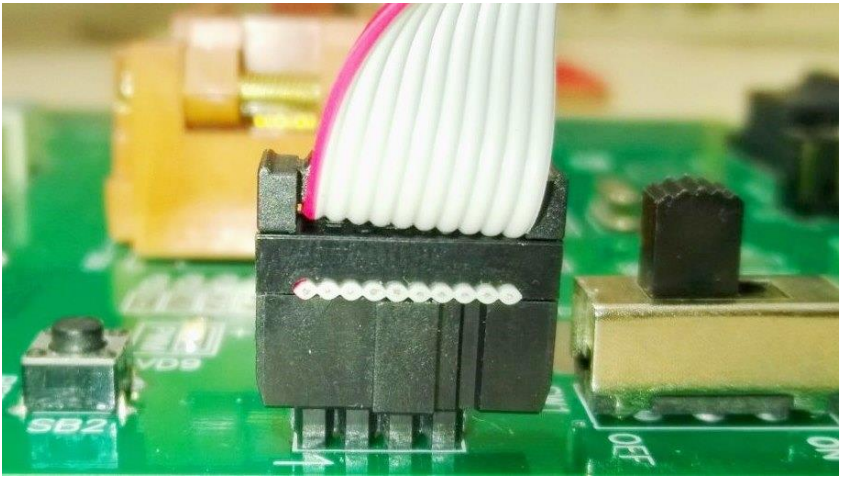
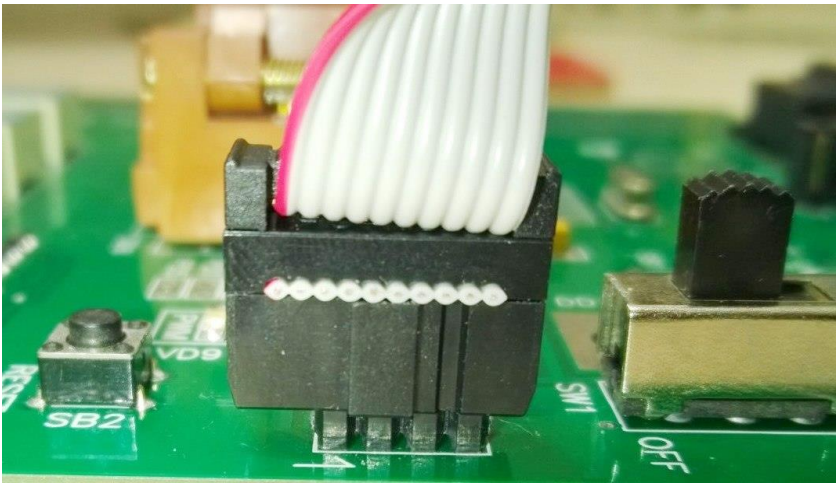
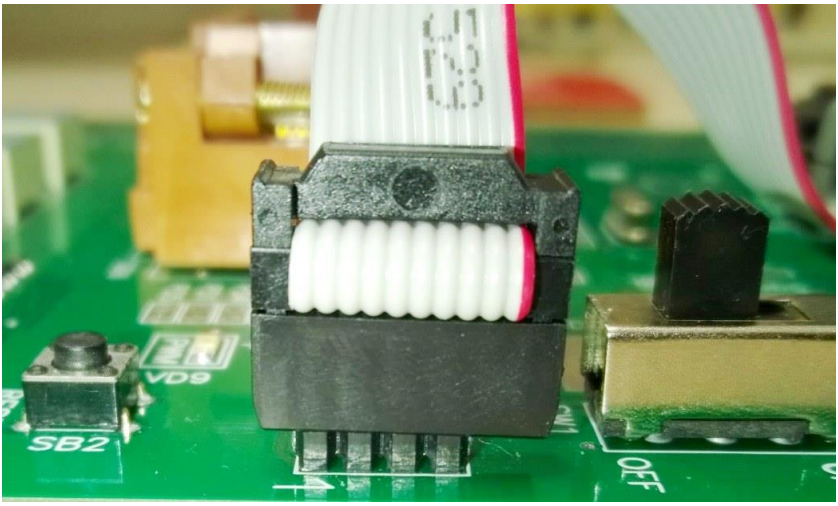


Рис.5. Корректное (а) и некорректное (б,в) подключение USB-программатора к плате.



б)



в)

5. Загрузка программы в микроконтроллер.

Перейдите с основного окна проекта на вкладку **Programming** и далее – на вкладку **Program Memory**, показанную на рис.6.

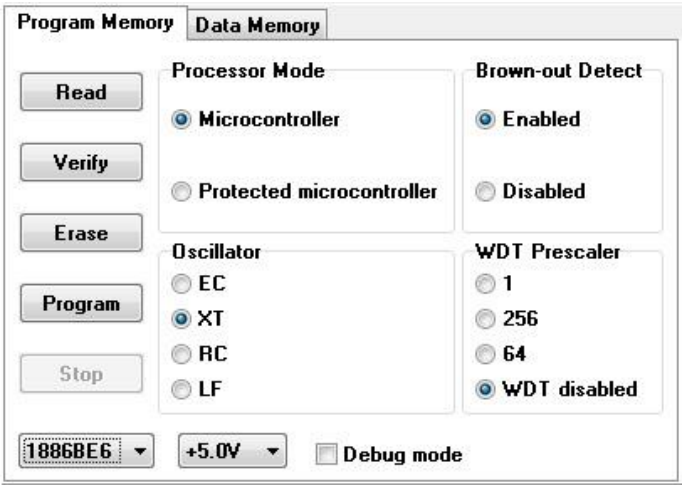


*Рис.6. Вид вкладки **Program Memory***

В окне **Program Memory**:

- в разделе **Processor Mode** выберите **Microcontroller**;
- в разделе **Oscillator** выберите **XT**;
- в разделе **Brown-out Detect** выберите **Enabled**;
- в разделе **WDT Prescaler** выберите **WDT disabled**.
- установите напряжение **+5.0V**;
- в списке микроконтроллеров выберите **1886BE6**.

Далее необходимо загрузить программу в память микроконтроллера.

Для успешной загрузки любой программы на микроконтроллер К1886ВЕ61У сначала необходимо стереть прежние инструкции, записанные в его память, нажав клавишу **Erase**, затем загрузить новые инструкции нажатием клавиши **Program**.

После того как новая программа будет успешно загружена в память микроконтроллера, следует отключить USB-программатор.

Подключите питание 5 В на вход XP2.

Переключатель 3 на микросхеме поставьте в положение «ON».

6. Получение результата.

На индикаторе, размещенном на плате, должны высветиться значения каналов, а именно АЦП2, АЦП3, АЦП4, АЦП5 и четыре свободных канала. Подобрав нужные значения подстроечных

резисторов R22 и R24, получите на выходе АЦП2 и АЦП3 значения 700 и 900 соответственно.

С помощью клавиатуры введите на вход ШИМ1 (PWM1) значение 500, на вход ШИМ2(PWM2) – 600, на вход ЦАП1 (DAC1) – 700, на вход ЦАП2(DAC2) – 800.

Покажите работу преподавателю.

Требования к отчету

Отчет должен содержать:

1) формулировку цели работы;
2) конспект практической части;
3) анализ полученных результатов.

Контрольные вопросы

1. С какой целью производится стирание памяти микроконтроллера?
2. С какой целью производится верификация памяти микроконтроллера?
3. Что общего и какие различия у файлов с разрешением .bin и .hex?
4. Сохраняются ли в памяти микроконтроллера значения, введенные на каналы при перезапуске, при перезагрузке, при перепрограммировании?

Лабораторная работа № 2

Синтез простейших инструкций с помощью IDE MPASM для микропроцессорного ядра К1886ВЕ61У с использованием языка Assembler

Цель работы: ознакомление с системой команд микроконтроллера К1886ВЕ61У и использование команд языка Assembler для выполнения арифметических и логических операций.

***Оборудование*:** ПК; отладочная плата Eval17 на основе микроконтроллеров К1886ВЕ61У компании «ПКК Миландр».

Программное обеспечение: исполняемый модуль DEVCPP.EXE компании «ПКК Миландр»; исполняемый модуль для компиляции программ на языке Assembler MPASMWIN.EXE из пакета MPLAB IDE фирмы MICROCHIP; драйвер giveio.sys компании «ПКК Миландр» для работы с LPT-портом под ОС Windows XP и программа установки драйвера LOADDRV.EXE.

Программное и аппаратное обеспечение предоставлено компанией «ПКК Миландр»: http://milandr.ru/index.php?page=programmnoe-obespech

Продолжительность работы – 4 ч.

Теоретические сведения

Для отображения режимов работы демопрограммы используются 4-разрядные 8-сегментные (7+1) светодиодные индикаторы D8 – D11 (светодиодные матрицы с общим катодом). Разряды индикаторов (катоды) выбираются портами PD0, PD1, PD4 и PD5. Сегменты (аноды) через токоограничивающие резисторы подключены к сдвиговому регистру DD7, загружаемому с помощью портов PC6 и PC7.

Функции. Условия вызова функций

Как такового понятия «функция» в языке программирования Assembler для микроконтроллеров К1886ВЕ61У нет. Но можно условно

выделить функции в самой программе, осуществляя переходы в определенную область памяти.

Для вызова функции используется команда «CALL имя_функции» и «GOTO имя_функции». Отличие этих команд заключается в том, что после вызова функции командой CALL возможен аппаратный возврат указателя адреса, а после выполнения функции через команду GOTO возврат невозможен. Функция может храниться в любом месте программного кода. Для выхода из функции используется команда RETURN.

Пример:

```
CALL FUNC     ; компилятор переходит на строку FUNC.
.
.
.
FUNC:         ; указатель адреса функции.
.             ; команды, составляющие тело функции.
.
.
RETURN        ; выход из функции.
```

Вызов функции удобен когда есть условие ее вызова.

Пример:

```
MOVLW 0x20   ; записать в WREG значение 0x20.
MOVWF A      ; скопировать значение из WREG в переменную A
MOVLW 0xFA   ; записать в WREG значение 0xFA.
CPFSGT A     ; если A>WREG,
             ;   то следующая команда пропускается.
CALL FUNC    ; если же A<=WREG,
             ;   то вызвать некоторую функцию FUNC
```

Особые регистры

WREG – 8-битный регистр, используемый практически во всех командах языка Assembler как буфер обмена. Адрес регистра 0xA.

ALUSTA – регистр, содержащий биты статуса арифметического и логического блоков и биты управления режимом для режима косвенной адресации.

Как в случае со всеми другими регистрами, в регистр ALUSTA может быть загружен результат выполнения любой команды.

При сложении двух 16-ричных чисел иногда появляется новый разряд:

0x0F + 0x0F = 0x1E.

Появление единицы в разряде ведет к изменению 0-го бита (он же бит переноса) регистра ALUSTA с 0 на 1.

Пример: сложим числа 0x20 и 0xFA и проверим, чему равен 0-ой бит ALUSTA.

```
MOVLW 0x20    ; записать в WREG значение 0x20.
MOVWF A       ; скопировать значение из WREG в A
MOVLW 0xFA    ; записать в WREG значение 0xFA
ADDWF A,1     ; прибавить к a число 0xFA, результат - в A.

BTFSC ALUSTA,0  ; если 0-й бит регистра ALUSTA равен 0,
                ; то следующая команда пропускается.
CALL  FUNC      ; если же 0-й бит равен 1,
                ; то вызвать функцию FUNC.
```

PRODH:PRODL – регистры, используемые для сохранения результата умножения двух чисел, каждое из которых состоит из двух 8-ричных цифр.

Пример: при умножении числа 0xAF на число 0xDE результат будет равен 0x97C2. В регистр PRODH помещается значение 97, а в регистр PRODL – C2.

Примеры программ

Пример 1. Требуется написать программу на языке программирования Assembler, которая выводит на индикатор DD9 число 3, а на индикатор DD10 – число 6.

Начать следует с подключения заголовочного файла и выделения памяти под используемые регистры. Адрес переменной DATAREG рекомендуется задать таким же, как в этом примере. Адреса других переменных рекомендуется использовать в диапазоне от 00H до 1FH для успешной реализации команды MOVFP:

```
INCLUDE <1886VE6M.INC>   ; подключаем заголовочный файл.
;*****************************************************
; объявляем требуемые регистры
DATAREG      EQU 0x1C
;*****************************************************
  ORG 0x0    ; задаем начальный адрес программы.
  BSF GLINTD ; запрещаем работу прерываний.
  GOTO PREP  ; переходим к области подготовки,
```

```
                ; игнорируя область обработки прерываний.
```

Порты D и C будем использовать для отображения символов на индикаторе. В рамках лабораторной работы порт D отвечает за активацию всех элементов на необходимом индикаторе, порт C предотвращает активацию лишних.

```
  ORG   0x30        ; пропускаем область памяти, предназ-
                    ; наченную для обработки прерываний.
PREP:
  MOVLB 6           ; выбираем 6-й банк, так как регистр
                    ; ADCON хранится именно там.
  CLRF  ADCON0      ; отключаем АЦП.
  SETF  ADCON1      ; переключаем порт C в цифровой режим.

  MOVLB 1           ; выбираем 1-й банк.
  SETF  PORTD       ; PD = 1111 1111.
  CLRF  PORTC       ; PC = 0000 0000.
  MOVLW 0x0
  MOVWF DDRD        ; переключаем порт D на выход.
  MOVWF DDRC        ; переключаем порт C на выход.
```

Далее прописывается функция, которая будет выводить значения на табло. Начало функции записывается как «имя_функции:» (без кавычек). В конце нужно зациклить функцию с помощью команды «GOTO имя_функции», чтобы отображение было сразу на всех требуемых индикаторах. Фактически индикаторы будут постоянно загораться и гаснуть, однако частота циклов столь велика, что процесс воспринимается человеческим глазом как постоянное свечение. Если не зациклить функцию, порту D будет присвоено последнее указанное значение.

Каждому индикатору соответствует свое значение порта D согласно табл.1.

Таблица 1

Соответствие портов и индикаторов

Индикатор	Значение порта D
DD8	0xFE
DD9	0xFD
DD10	0xEF
DD11	0xDF

Чтобы зажечь необходимые элементы на индикаторе, следует в регистр DATAREG отправить 8-битное значение в 16-ричном коде. Каждому элементу, и точке в том числе, соответствует определенный бит регистра DATAREG.

На рис. 7 показан порядок нумерации элементов индикатора.

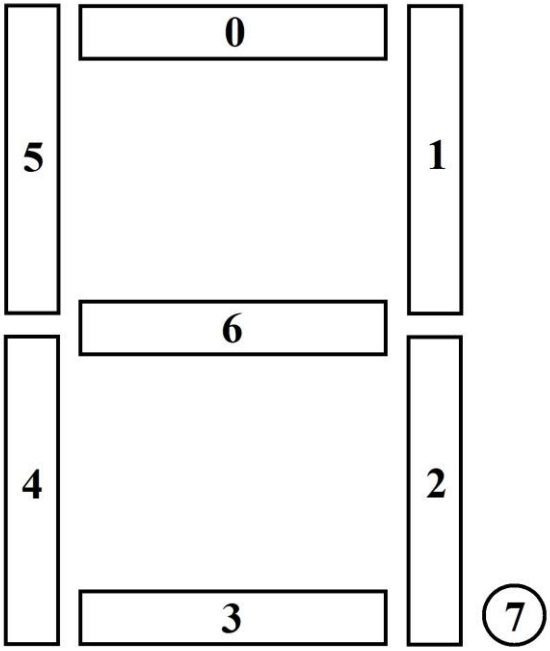


Рис.7. Схема нумерации элементов индикатора

Например, чтобы получить единицу на индикаторе, необходимо активировать только элементы 1 и 2, т.е. в двоичном коде требуемое значение будет равно 0000 0110, соответственно в 16-ричном – 006H или 0x006:

```
START:
  MOVLW 0x4F    ; передаем значение, соответствующее
                ; индикации символа "3" в регистр WREG.
  MOVWF DATAREG ; передаем значение из регистра WREG
                ; в регистр DATAREG.
  CALL TRANSFER ; вызываем функцию TRANSFER.
```

```
  MOVLW 0xFD    ; передаем в регистр WREG значение порта
                ; D для требуемого индикатора.
  MOVWF PORTD   ; передаем значение регистра WREG в порт D

;=== повторяем операцию для другого индикатора
  MOVLW 0x7D     ; индикация символа "6"
  MOVWF DATAREG
  CALL  TRANSFER
  MOVLW 0xEF
  MOVWF PORTD
GOTO START
```

Рассмотрим функцию TRANSFER. Из регистра DATAREG в функцию TRANSFER попадает некоторое значение:

```
;=== передача данных в сдвиговый регистр
TRANSFER:
       BCF DDRC,7
       SETF PORTD,F

       BSF PORTC,7
       BTFSS DATAREG,7
       BCF PORTC,7
       BSF PORTC,6
       BCF PORTC,6

       BSF PORTC,7
       BTFSS DATAREG,6
       BCF PORTC,7
       BSF PORTC,6
       BCF PORTC,6

       BSF PORTC,7
       BTFSS DATAREG,5
       BCF PORTC,7
       BSF PORTC,6
       BCF PORTC,6

       BSF PORTC,7
       BTFSS DATAREG,4
       BCF PORTC,7
       BSF PORTC,6
       BCF PORTC,6

       BSF PORTC,7
```

```
        BTFSS DATAREG,3
        BCF PORTC,7
        BSF PORTC,6
        BCF PORTC,6

        BSF PORTC,7
        BTFSS DATAREG,2
        BCF PORTC,7
        BSF PORTC,6
        BCF PORTC,6

        BSF PORTC,7
        BTFSS DATAREG,1
        BCF PORTC,7
        BSF PORTC,6
        BCF PORTC,6

        BSF PORTC,7
        BTFSS DATAREG,0
        BCF PORTC,7
        BSF PORTC,6
        BCF PORTC,6

        BSF DDRC,7
RETURN
;*********************************************************
END
```

Функция TRANSFER реализована таким образом, что порт D отвечает за активацию всех элементов на необходимом индикаторе, а порт C предотвращает активацию лишних.

Для возврата к адресу, где вызывалась функция, в конце функции используется команда RETURN. Если этого не сделать, компилятор продолжит обработку инструкций, находящихся непосредственно после функции.

Последней использованной командой должна быть END – команда, обозначающая завершение программы.

Пример 2. Требуется написать программу на языке программирования Assembler, которая выводит двухразрядное число сразу на два индикатора – DD10 и DD11.

Микроконтроллер К1886ВЕ61У позволяет передавать 8-битные значения. Восьми бит достаточно, чтобы уместить любое двухразрядное 16-ричное число: наибольшее двухразрядное 16-ричное число «FF» в

двоичном виде выглядит как 1111 1111.

Напишем код программы на базе кода и примера 1.

Сначала выделим память для нового регистра:

```
SHIFT  EQU 0x1D
```

Далее перепишем функцию START:

```
START:
  MOVLW 0x1A  ; число 0x1A передаем в регистр WREG.
  MOVWF SHIFT ; содержимое регистра WREG помещаем в
              ; регистр SHIFT, в регистре WREG
              ; по-прежнему число 0x1A.
  ANDLW 0x0F  ; наложение маски 0x0F на число
              ; в регистре WREG
  CALL SYMGEN ; вызов функции SYMGEN.
```

В качестве числа задано 1A. С помощью команды ANDLW 0x0F на это число накладывается «маска» (табл.2).

Таблица 2

Наложение маски на число

Маска	0	0	0	0	1	1	1	1
Число	0	0	0	1	1	0	1	0

Для наложения маски требуется выполнить операцию конъюнкции между соответствующими битами. В результате получается 4-битное двоичное число 0000 1010, соответствующее 16-ричному 0x0A. Вызов функции SYMGEN необходим, чтобы закодировать 8-битное число A на индикатор. Функция прибавляет значение, хранимое в WREG, к PCL, перемещая тем самым указатель адреса инструкции на необходимую позицию, и записывает в регистр WREG то значение, которое требуется для отображения соответствующего символа. После этого осуществляется считывание из стека адреса инструкции, следующей за инструкцией «CALL SYMGEN». Функцию SYMGEN рекомендуется объявить следом за функцией TRANSFER:

```
SYMGEN:
       ADDWF PCL,1
       RETLW 0x3F   ; "0"
       RETLW 0x06   ; "1"
```

```
        RETLW 0x5B    ; "2"
        RETLW 0x4F    ; "3"
        RETLW 0x66    ; "4"
        RETLW 0x6D    ; "5"
        RETLW 0x7D    ; "6"
        RETLW 0x07    ; "7"
        RETLW 0x7F    ; "8"
        RETLW 0x6F    ; "9"
        RETLW 0x77    ; "A"
        RETLW 0x7C    ; "b"
        RETLW 0x39    ; "C"
        RETLW 0x5E    ; "d"
        RETLW 0x79    ; "E"
        RETLW 0x71    ; "F"
RETURN
```

Возвращаемся к написанию функции START. Реализуем индикацию полученного символа на соответствующем индикаторе, как в примере 1:

```
    MOVWF DATAREG
    CALL  TRANSFER
    MOVLW 0xDF
    MOVWF PORTD
```

Далее с помощью команды RRCF осуществляем цикличный сдвиг вправо на 1 бит 4 раза. Маска остается прежней – 0x0F:

```
  RRCF SHIFT,1  ; цикличный битовый сдвиг вправо,
                ; результат сохраняется в регистр SHIFT.
  RRCF SHIFT,1
  RRCF SHIFT,1
  RRCF SHIFT,0  ; цикличный битовый сдвиг вправо,
                ; результат сохраняется в регистр WREG.
  ANDLW 0x0F
```

Наложение маски на число после битового перехода показано в табл.3.

Таблица 3

Наложение маски на число после битового перехода

Маска	0	0	0	0	1	1	1	1

Число	1	0	1	0	0	0	0	1

Остается получить код соответствующего символа с помощью функции SYMGEN и отобразить полученное значение на второй индикатор. Для сохранения динамического отображения значений необходимо зациклить основную функцию START:

```
CALL  SYMGEN
MOVWF DATAREG
CALL  TRANSFER
MOVLW 0xEF
MOVWF PORTD
GOTO  START
```

Лабораторное задание

1. Написать текст программы для вывода результата операции на световое табло микроконтроллера.
2. Выполнить этап трансляции описания, получив файлы .bin, .hex.
3. Задать значения чисел в коде программы.
4. Загрузить программу в микроконтроллер.
5. Отобразить на индикаторе результат, соответствующий варианту (табл.4). Все исходные числа могут состоять из одной или двух 16-ричных цифр.

Таблица 4

Варианты заданий

Вариант	Требуемая операция
1	A = A1 + A2, где A1, A2 – двухразрядные. Учесть, что результат может быть числом, состоящим из трех цифр.
2	A = A1 – A2, где A1, A2 – двухразрядные. Учесть результат в виде отрицательного числа.
3	Среди чисел A1, A2, A3 найти минимальное и вывести его на индикатор.
4	Среди чисел A1, A2, A3 найти максимальное и вывести его на индикатор.

5	Даны четыре 16-ричные цифры. Вывести на индикатор цифры, значение которых больше 8.
6	Если A = 2, вывести на индикатор слово LOOP. В ином случае – слово SOUP.
7	Если A > B, вывести на индикатор слово FULL. В ином случае – слово FALL".
8	A = A1 * A2, где A1, A2 – двухразрядные. Учесть, что результат может быть числом, состоящим из четырех цифр.

Порядок выполнения работы

1. Создать проект и его структуру.
2. Составить алгоритм программы на листе.
3. Написать соответствующий заданию код программы, указать заголовочный файл.
4. Скомпилировать проект.
5. Подключить плату к компьютеру.
6. Загрузить программу в микроконтроллер.
7. Оценить результат.

Примечание. Программа должна состоять из четырех частей:

1) объявление заголовочного файла и выделение памяти для используемых регистров;

2) объявление начальных данных в адресном пространстве микроконтроллера. Программа должна выдавать разные результаты при разных указанных начальных данных;

3) алгоритм действия над данными. Алгоритм удобно предварительно расписать на листе бумаги, а затем реализовать в коде;

4) вывод результата на индикатор. Выводимый результат может быть либо суммой, разностью или произведением, либо результатом выполнения операции сравнения.

Части могут пересекаться между собой в коде программы.

Требования к отчету

Отчет должен содержать:

1) формулировку цели работы;

2) исходные данные и задания;
3) анализ полученных результатов;
4) алгоритм программы.

Контрольные вопросы

1. Сколько разрядов выделено в памяти микроконтроллера для регистра?
2. Из какого количества символов может состоять максимальное число, которое можно вывести на индикатор?
3. Для чего служит регистр WREG?
4. Для чего служит регистр ALUSTA?
5. Как на языке Assembler организовать вызов подпрограммы?

Лабораторная работа № 3

Ввод значений на индикаторы отладочной платы Eval17

Цель работы: ознакомление с системой команд микроконтроллера К1886ВЕ61У, необходимых для ввода информации на световое табло.

Оборудование: ПК; отладочная плата Eval17 на основе микроконтроллеров К1886ВЕ61У компании «ПКК Миландр».

Программное обеспечение: исполняемый модуль DEVCPP.EXE компании «ПКК Миландр»; исполняемый модуль для компиляции программ на языке Assembler MPASMWIN.EXE из пакета MPLAB IDE фирмы MICROCHIP; драйвер giveio.sys компании «ПКК Миландр» для работы с LPT-портом под ОС Windows XP и программа установки драйвера LOADDRV.EXE.

Программное и аппаратное обеспечение предоставлено компанией «ПКК Миландр»: http://milandr.ru/index.php?page=programmnoe-obespech

Продолжительность работы – 4 ч.

Теоретические сведения

Схема подключения кнопок

Для управления демонстрационными программами используется клавиатура, состоящая из четырех клавиш: SB3–SB6. Выбор сканируемой клавиши определяется портами PD0, PD1, PD4 или PD5, состояние клавиши считывается с порта PC7.

На рис.1 приведена схема подключения кнопок к портам C и D.

Если подать логический «0» на порт PD0, то при нажатой кнопке SB3 на порт C придет логический «0», диод откроется и напряжение на порту C упадет. В остальных случаях на порт C придет логическая «1» с питания. Аналогично и с остальными кнопками.

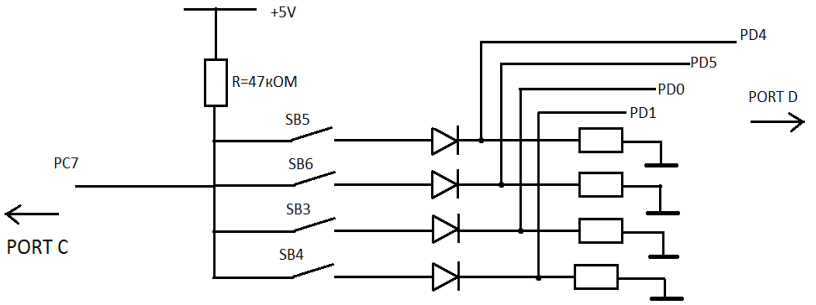


Рис.1. Схема подключения кнопок к портам C и D

Опрос кнопок

На плате Eval17 можно опрашивать кнопки только по одной.

Прежде чем опрашивать кнопки, необходимо установить порт D в состояние 1111 1111. Для опроса кнопки необходимо соответствующий бит порта D установить в 0. После этого проверить, в каком состоянии находится порт PC7, т.е. узнать, чему равен 7-й бит порта C – 0 или 1.

Приведем пример опроса кнопки SB5:

```
POLL:
  SETF PORTD     ; устанавливаем порт D в 1111 1111.
  BCF PORTD,4    ; 4-й бит порта D отвечает за кнопку
                 ; SB5, обнуляем его.
  BTFSS PORTC,7  ; проверяем, пришел ли "0" на порт C.
                 ; если нет – игнорируем следующую команду
  CALL SB_5      ; если да, значит кнопка SB5 нажата –
                 ; переходим к выполнению функции SB_5.
  BSF PORTD,4    ; восстанавливаем порт D
GOTO POLL
```

Таким образом, если 7-й бит порта PC7 равен 0, необходимо прервать программу и вызвать инструкции (функцию SB_5) для выполнения операций, соответствующих положению нажатой кнопки. Если PC7 = 1 – продолжить процесс опроса клавиш.

Важно помнить о состоянии порта D во время опроса кнопок. Всякий раз, когда завершается опрос одной кнопки, необходимо восстанавливать порт D в прежнее состояние – 1111 1111.

Табл.1 составлена по схеме подключения кнопок к портам C и D.

Таблица 1

Соответствие битов порта D, кнопок и индикаторов

Кнопка	Номер бита порта D	Код порта D для индикации	Индикатор
SB3	0	FE	DD8
SB4	1	FD	DD9
SB5	4	EF	DD10
SB6	5	DF	DD11

Следует помнить, что с любой кнопкой можно задавать значения на любой индикатор.

Примеры программ

Пример 1. Требуется написать программу на языке программирования Assembler, которая выводит на индикатор DD9 значение 1 при нажатии кнопки SB5 и значение 0 при нажатии кнопки SB4:

```
INCLUDE <1886VE6M.INC>
;*********************************************************
; объявляем требуемые регистры
DATAREG       EQU    0x1F
DD9           EQU    0xA1   ; значение на индикаторе DD9.
;*********************************************************
  ORG  0x0
  BSF  GLINTD
  GOTO PREP
  ORG  30H
;*********************************************************
PREP:
  MOVLB 6         ; переходим в банк 6, где хранятся
                  ; регистры ADCON0 и ADCON1.
  CLRF ADCON0      ; отключаем АЦП.
  SETF ADCON1
  MOVLB 1         ; переходим в банк1, где хранится
                  ; регистр PORTC и PORTD.
  CLRF PORTC      ; очищаем порты.
  CLRF PORTD
  SETF DDRC       ; переводим порт C и D на вход.
  SETF DDRD
  CLRF DD9        ; чистим регистры перед использованием
```

```
  GOTO POLL      ; игнорируем область прерываний, они
                 ; будут происходить при нажатии кнопок
;*******************************************************
; область обработки прерываний
SB_5:          ; действия, если была нажата кнопка SB_5.
  MOVLW 0x1
  MOVWF DD9
  RETURN
SB_4:          ; действия, если была нажата кнопка SB_4.
  MOVLW 0x00
  MOVWF DD9
  RETURN
;*******************************************************
; область опроса клавиатуры
POLL:
POLL1:             ; опрос кнопки SB5.
  BCF PORTD,4      ; подаем "0" для кнопки SB5.
  BTFSS PORTC,7    ; проверяем, пришел ли "0" на порт C.
                   ; если нет, то переходим к опросу
                   ; следующей кнопки.
  CALL SB_5        ; если да, значит кнопка SB5 нажата -
                   ; переходим к выполнению функции SB_5.
  CALL START       ; процесс индикации текущего значения.
POLL2:             ; опрос кнопки SB4.
  BCF PORTD,1
  BTFSS PORTC,7
  CALL SB_4
  CALL START
  GOTO POLL        ; зацикливание.
;*******************************************************
START:           ; индикация текущего значения регистра DD9
  MOVLB 1        ; объявляем порты D и C на выход.
  SETF PORTD     ; PD = 1111 1111.
  MOVLW 0x0
  MOVWF DDRD
  CLRF PORTC     ; PC = 0000 0000.
  MOVWF DDRC
  MOVFP DD9,WREG  ; помещаем значение регистра DD9 в WREG
  CALL SYMGEN
  MOVWF DATAREG
  CALL TRANSFER
  MOVLW 0xFD
  MOVWF PORTD      ; индикация регистра DD9 на табло.
```

```
; обнуление индикации и вывод пустых значений на
; индикатор DD9 через сдвиговый регистр.
; Операция необходима для того, чтобы число 8
; не перекрывало значения на индикаторе.
    MOVLW 0x0
    MOVWF DATAREG
    CALL TRANSFER
    BCF PORTD,1
    SETF PORTD
    BSF PORTD,1
RETURN
;*********************************************************
;передача данных в сдвиговый регистр
TRANSFER:
       BCF DDRC,7
       SETF PORTD,F

       BSF PORTC,7
       BTFSS DATAREG,7
       BCF PORTC,7
       BSF PORTC,6
       BCF PORTC,6

       BSF PORTC,7
       BTFSS DATAREG,6
       BCF PORTC,7
       BSF PORTC,6
       BCF PORTC,6

       BSF PORTC,7
       BTFSS DATAREG,5
       BCF PORTC,7
       BSF PORTC,6
       BCF PORTC,6

       BSF PORTC,7
       BTFSS DATAREG,4
       BCF PORTC,7
       BSF PORTC,6
       BCF PORTC,6

       BSF PORTC,7
       BTFSS DATAREG,3
       BCF PORTC,7
```

```
        BSF PORTC,6
        BCF PORTC,6

        BSF PORTC,7
        BTFSS DATAREG,2
        BCF PORTC,7
        BSF PORTC,6
        BCF PORTC,6

        BSF PORTC,7
        BTFSS DATAREG,1
        BCF PORTC,7
        BSF PORTC,6
        BCF PORTC,6

        BSF PORTC,7
        BTFSS DATAREG,0
        BCF PORTC,7
        BSF PORTC,6
        BCF PORTC,6

        BSF DDRC,7
RETURN
;*****************************************************
; область преобразования значений для индикации
SYMGEN:
        ADDWF PCL,1
        RETLW 0x3F    ; "0"
        RETLW 0x06    ; "1"
        RETLW 0x5B    ; "2"
        RETLW 0x4F    ; "3"
        RETLW 0x66    ; "4"
        RETLW 0x6D    ; "5"
        RETLW 0x7D    ; "6"
        RETLW 0x07    ; "7"
        RETLW 0x7F    ; "8"
        RETLW 0x6F    ; "9"
        RETLW 0x77    ; "A"
        RETLW 0x7C    ; "b"
        RETLW 0x39    ; "C"
        RETLW 0x5E    ; "d"
        RETLW 0x79    ; "E"
        RETLW 0x71    ; "F"
RETURN
```

```
;*******************************************************
END
```

Лабораторное задание

1. Написать код программы, позволяющий вводить новые значения с клавиатуры и высвечивать эти значения на индикаторе платы согласно варианту (табл.2).

2. Выполнить этап трансляции описания, получив файлы .bin, .hex.

3. Загрузить программу в микроконтроллер.

4. Вывести на световое табло значения, согласно варианту.

Примечание. Все исходные числа могут состоять из одной или двух 16-ричных цифр.

Таблица 2

Варианты заданий

Вариант	Требуемая операция
1	Задействовать четыре кнопки: 1-ая выводит на одном индикаторе число, не равное нулю, 2-ая выводит на этом же индикаторе число 0; 3-я выводит на другом индикаторе число, не равное первому числу и нулю, 4-ая – на этом же индикаторе число 0.
2	Задействовать четыре кнопки, выводящие разные числа, не равные нулю, на один и тот же индикатор.
3	Задействовать две кнопки. При нажатии одной, на табло должно выводиться слово COOL, при нажатии другой – слово LIFE.
4	Задействовать две кнопки. При нажатии одной на табло должно выводиться слово On, при нажатии другой – слово OFF.
5	Задействовать четыре кнопки. При нажатии любой из кнопок – SB3, SB4, SB5, SB6 – на любом индикаторе высветить номер последней нажатой кнопки.

Требования к отчету

Отчет должен содержать:

1) формулировку цели работы;
2) исходные данные и задания;
3) анализ полученных результатов;
4) алгоритм программы.

Контрольные вопросы

1. Какие функции выполняет порт D в микроконтроллере К1886ВЕ61У?

2. Объяснить работу схемы подключения кнопок.

3. Объяснить принцип опроса кнопок в коде программы.

Лабораторная работа № 4

Использование встроенного таймера в микроконтроллере К1886ВЕ61У

Цель работы: ознакомление со встроенным таймер-счетчиком; получение навыков о встроенных флагах прерывания, а также о флаге запроса прерывания.

Оборудование: ПК; отладочная плата Eval17 на основе микроконтроллеров К1886ВЕ61У компании «ПКК Миландр».

Программное обеспечение: исполняемый модуль DEVCPP.EXE компании «ПКК Миландр»; исполняемый модуль для компиляции программ на языке Assembler MPASMWIN.EXE из пакета MPLAB IDE фирмы MICROCHIP; драйвер giveio.sys компании «ПКК Миландр» для работы с LPT-портом под ОС Windows XP и программа установки драйвера LOADDRV.EXE.

Программное и аппаратное обеспечение предоставлено компанией «ПКК Миландр»: http://milandr.ru/index.php?page=programmnoe-obespech

Продолжительность работы – 4 ч.

Теоретические сведения

Работа с прерываниями

В лабораторной работе №3 рассмотрено прерывание с использованием команд CALL и RETURN. Команда RETURN позволяет вернуться в место, где была вызвана подпрограмма командой CALL. В данной работе будем дополнительно использовать команду RETFIE – возврат из немаскированного прерывания.

Немаскированное прерывание – прерывание, в котором бит разрешения запроса прерывания (в данной лабораторной работе T0IE) равен единице.

Перед обработкой прерываний требуется разрешить все прерывания, для этого необходимо установить бит глобального запрета прерываний «GLIND» в 1. После этого следует перейти к области опроса пре-

рываний. В этой области нужно задать начальные данные регистров для вывода значений на световое табло, установить значения для регистров данного таймера и битов управления, а затем проверить бит запроса прерывания.

Область обработки прерываний для таймера 0 можно определять только по адресу 0x10. Остальные тонкости работы с прерываниями описаны в последующих разделах.

Устройство таймера 0. Предделитель частоты. Биты и регистры управления

В данной лабораторной работе используется блок «таймер 0» микроконтроллера К1886ВЕ61У. «Таймер 0» задает некоторую частоту – внутреннюю или внешнюю. Внешней частоты нет, поэтому используется только внутренняя. Эта частота представляет собой поток импульсов, каждый импульс фиксируется в связанных регистрах таймера TMR0L:TMR0. Минимальное значение этих регистров равно 0x0000, максимальное – 0xFFFF. Как только эти регистры будут переполнены, флаг запроса прерывания T0IF поменяет значение с 0 на 1. Важно задать бит T0IE – бит разрешения прерываний для данного таймера. Если T0IE = 1 – прерывания будут происходить, если T0IE = 0 – прерывания не будут обработаны программой.

Имеется проблема, заключающаяся в том, что частота слишком велика. Ее можно разделить с помощью так называемого предделителя частоты. Предделитель частоты задается 4-мя битами регистра T0STA, а именно T0PS3, T0PS2, T0PS1, T0PS0. Описание этих битов представлено в табл.1.

Таблица 1

Описание битов предделителя частоты

Функциональное имя бита	Описание
T0CS	Бит выбора источника тактирования для «таймера 0». Этот бит выбирает источник синхронизации для «таймера 0»: 1 – внутренняя тактовая частота с генератора циклов, 0 – внешнее тактирование с вывода PA1/T0CLK.

T0PS[3..0]	Биты выбора предделителя для таймера 0. Эти биты позволяют выбрать величину деления частоты: 0000 – 1:1 0001 – 1:2 0010 – 1:4 0011 – 1:8 0100 – 1:16 0101 – 1:32 0110 – 1:64 0111 – 1:128 1xxx – 1:256
T0IE	Бит разрешения прерываний: 1 – прерывания разрешены, 0 – прерывания запрещены
T0IF	Бит запроса прерываний: 1 – регистры таймера заполнены, 0 – регистры таймера еще не заполнены

Нюансы регистров

Оптимальная величина деления частоты равна 1:64, тогда флаг запроса T0IF будет устанавливаться раз в секунду.

Запись в любой из регистров TMR0L и TMR0H блокирует изменение соответствующей части счетчика «таймера 0» в последующем после записи цикле микроконтроллера, при этом запись не оказывает влияния на другую часть счетчика. Поэтому рекомендуется последовательно производить запись сначала регистра TMR0L, а затем TMR0H. Запись в любой из регистров TMR0L или TMR0H сбрасывает в исходное состояние предделитель, поэтому следует устанавливать предделитель частоты всякий раз, когда выполняется проверка бита запроса прерываний.

Бит T0IF необходимо обновлять каждый раз, когда регистры таймера переполнены. Регистры таймера также необходимо обновлять. И то, и другое делается в области обработки прерываний. В данной лабораторной работе начальное значение регистров управления примем за 0000.

Примеры программ

Пример 1. Требуется написать программу-таймер на языке программирования Assembler, результат вывести в десятичных цифрах, учесть один индикатор под секунды, старт 9:

```
INCLUDE <1886VE6M.INC>
;*******************************************************
```

```
; объявляем требуемые регистры
DATAREG  EQU 0x2C
TIME     EQU 0x1A
TIME2    EQU 0x1B
TIME3    EQU 0x1C
;*********************************************************
  ORG 0x0 ; задаем начальный адрес расположения программы
  BCF GLINTD ; разрешаем работу прерываний.
  MOVLB 6
  CLRF  ADCON0
  SETF  ADCON1
  MOVLB 1
  CLRF  PORTC
  CLRF  PORTD
  MOVLW 0x1
  MOVWF DDRD
  MOVWF DDRC
  GOTO POLL  ; переходим к области опроса прерываний.
;*********************************************************
; область обработки прерываний
  ORG 0x10
INTERRUPT:
  MOVLW  0x0
  CPFSEQ TIME
  DECF TIME,1
  BCF T0IF
  CLRF TMR0L  ; задаем младший регистр таймера.
  CLRF TMR0H  ; задаем старший регистр таймера.
RETFIE
;*********************************************************
; область опроса прерываний
  ORG  0x30
POLL:
  MOVLW 0x9    ; задаем начальное значение регистра,
               ; значение которого будет на табло.
  MOVWF TIME
  CLRF  TMR0L  ; задаем младший регистр таймера.
  CLRF  TMR0H  ; задаем старший регистр таймера.
  BSF   T0CS   ; выбираем внутреннюю частоту генерации
               ; циклов
  BSF   T0IE   ; разрешаем запрос прерываний.
;*********************************************************
; проверка на необходимость прерывания
DIVIDER:
```

```
  BCF   T0PS0  ; устанавливаем предделитель частоты 1:64.
  BSF   T0PS1
  BSF   T0PS2
  BCF   T0PS3
  BTFSC T0IF   ; проверяем бит запроса прерываний.
  GOTO INTERRUPT
;*********************************************************
; область индикации
START:
  MOVLB 1
  SETF  PORTD
  CLRF  PORTC
  MOVLW 0x0
  MOVWF DDRD
  MOVWF DDRC
  MOVFP TIME,WREG
  CALL  SYMGEN
  MOVWF DATAREG
  CALL  TRANSFER
  MOVLW 0xDF
  MOVWF PORTD
GOTO DIVIDER
;*********************************************************
;передача данных в сдвиговый регистр
TRANSFER:
       BCF DDRC,7
       SETF PORTD,F

       BSF PORTC,7
       BTFSS DATAREG,7
       BCF PORTC,7
       BSF PORTC,6
       BCF PORTC,6

       BSF PORTC,7
       BTFSS DATAREG,6
       BCF PORTC,7
       BSF PORTC,6
       BCF PORTC,6

       BSF PORTC,7
       BTFSS DATAREG,5
       BCF PORTC,7
       BSF PORTC,6
```

```
        BCF PORTC,6

        BSF PORTC,7
        BTFSS DATAREG,4
        BCF PORTC,7
        BSF PORTC,6
        BCF PORTC,6

        BSF PORTC,7
        BTFSS DATAREG,3
        BCF PORTC,7
        BSF PORTC,6
        BCF PORTC,6

        BSF PORTC,7
        BTFSS DATAREG,2
        BCF PORTC,7
        BSF PORTC,6
        BCF PORTC,6

        BSF PORTC,7
        BTFSS DATAREG,1
        BCF PORTC,7
        BSF PORTC,6
        BCF PORTC,6

        BSF PORTC,7
        BTFSS DATAREG,0
        BCF PORTC,7
        BSF PORTC,6
        BCF PORTC,6

        BSF DDRC,7
RETURN
;*********************************************************
; область преобразования значений для индикации
SYMGEN:
        ADDWF PCL,1
        RETLW 0x3F    ; "0"
        RETLW 0x06    ; "1"
        RETLW 0x5B    ; "2"
        RETLW 0x4F    ; "3"
        RETLW 0x66    ; "4"
        RETLW 0x6D    ; "5"
```

```
        RETLW 0x7D    ; "6"
        RETLW 0x07    ; "7"
        RETLW 0x7F    ; "8"
        RETLW 0x6F    ; "9"
RETURN
;*******************************************************
END
```

Лабораторное задание

1. Написать текст программы, которая реализует таймер/секундомер согласно варианту (табл.2).
2. Выполнить этап трансляции описания, получив файлы .bin, .hex.
3. Загрузить программу в микроконтроллер К1886ВЕ61У.
4. Показать на световом табло работающий таймер/секундомер согласно варианту (см.табл.2).

Таблица 2

Варианты заданий

Вариант	Требуемая операция
1	Создать секундомер. Выделить под секунды два индикатора. Числа взять только десятичные. Запрограммировать кнопку SB5 как старт. Старт 00.
2	Создать таймер. Выделить под секунды два индикатора. Числа взять только десятичные. Запрограммировать кнопку SB6 как старт. Старт 49.
3	Создать секундомер. Выделить под секунды два индикатора. Числа взять только десятичные. Запрограммировать кнопку SB4 как стоп. Старт 00.
4	Создать таймер. Выделить под секунды два индикатора. Числа взять только десятичные. Запрограммировать кнопку SB6 как стоп. Старт 28.
5	Создать таймер. Выделить под секунды два индикатора. Числа взять только десятичные. Запрограммировать кнопку SB5 как рестарт. Старт 17.
6	Создать секундомер. Выделить под секунды два индикатора. Числа взять только десятичные. Запрограммировать кнопку SB6 как рестарт.

Порядок выполнения работы

Для написания кода программы необходимо выполнить следующие действия:

1. Объявить заголовочный файл и несколько переменных.
2. Задать начальный адрес программы, разрешить работу прерываний. Отключить АЦП, настроить порты C и D на вход.
3. По адресу 0x60 описать область опроса прерываний и функцию проверки на необходимость прерываний (функция DIVIDER в примере). Сделать вызов функции индикации из функции проверки на необходимость прерываний.
4. Перейти по адресу 0x90. Описать функцию индикации и вспомогательные подпрограммы, необходимые для индикации. В функции индикации переключить порты C и D на выход.
5. Добавить в функции проверки прерываний вызов функции опроса кнопки (см. лабораторную работу №3).
6. Задать адрес 0x10 и описать функцию обработки прерываний для таймера.
7. Выделить адрес для описания функций опроса кнопки. Свободную память программы можно посмотреть на вкладке **Programming**. Описать функцию для нажатой кнопки.
8. В функции индикации описать индикацию пустого значения и чистку порта D (см. лабораторную работу №3).
9. Завершить код командой END.

Требования к отчету

Отчет должен содержать:

1) формулировку цели работы;
2) исходные данные и задания;
3) анализ полученных результатов;
4) алгоритм программы.

Контрольные вопросы

1. Чем отличается команда RETURN от RETFIE?
2. Что представляет собой прерывание по флагу TOIE?

3. Какова внутренняя частота генерации циклов?

4. Почему возникает погрешность в «таймере 0»? Как можно избавиться от нее?

Литература.

1. Демонстрационно-отладочная плата Eval17. Техническое описание. URL:
http://milandr.ru/uploads/Products/product_86/komplekt_1886BE6.pdf
(дата обращения 01.01.2015).

2. Интегрированная среда разработки, внутрисхемной отладки программ и программирования внутренней памяти микроконтроллеров. IDE1886 версия 8.5. Руководство пользователя. URL:
http://milandr.ru/uploads/Products/product_135/IDE1886.pdf
(дата обращения 01.01.2015).

3. Высокопроизводительный универсальный 8-битный микроконтроллер с 12-разрядным АЦП, ЦАП и компаратором 1886ВЕ6(61)У, К1886ВЕ6(61)У, 1886ВЕ6(61)У1, К1886ВЕ6(61)У1, К1886ВЕ61Н4. Спецификация. URL:
http://milandr.ru/uploads/Products/product_39/spec_1886BE6.pdf
(дата обращения 01.01.2015).

Содержание

Учебное издание

Алексеев *Алексей Алексеевич*
Владимиров *Леонид Леонидович*
Гуреев *Павел Владимирович*
Левицкий *Дмитрий Орестович*
Тогидный *Олег Борисович*
Шумилин *Сергей Сергеевич*

Практикум для лабораторных работ по курсу «Программирование микроконтроллеров»

Редактор *А.В. Тихонова*. Технический редактор *Л.Г. Лосякова*. Корректор *Л.Г. Лосякова*. Верстка авторов.

Подписано в печать с оригинал-макета . .2015. Формат 60×84 1/16. Печать офсетная. Бумага офсетная. Гарнитура Times New Roman. Усл. печ. л.2,78. Уч.-изд. л.2.4. Тираж 100 экз. Заказ 11.

Отпечатано в типографии ИПК МИЭТ.
124498, Москва, Зеленоград, площадь Шокина, дом 1, МИЭТ.