Лабораторная работа 1.

Программирование портов ввода-вывода микроконтроллеров AVR

Цель работы:

- изучение системы команд микроконтроллеров AVR и приемов программирования на языке AVR Ассемблер,

- получение навыков отладки программ в среде отладки AVR Studio 4 и VMLab,

- работа со стартовым набором (платой) STK500.

Введение

Микроконтроллеры AVR фирмы Atmel обладают широкими возможностями по вводу и выводу данных. Микроконтроллеры моделей ATx8515 для этих целей имеют четыре параллельных 8-разрядных порта Px (x=A, B, C, D) и один 3-разрядный порт PE (в модели ATmega8515). Все линии портов могут программироваться на ввод или вывод данных независимо друг от друга и имеют возможность подключения ко всем входам внутренних подтягивающих резисторов сопротивлением 35…120 кОм.

В состав каждого порта Px входят три регистра с именами DDRx, PORTx и PINx. В микроконтроллере AT90S8515 регистр PINx не имеет аппаратной реализации. Это имя используется для чтения линий интерфейса. На рис. 1 приведена общая структурная схема 8-разрядных портов Px и структурная схема одного разряда порта PD.y (y=0, 1, ...7) микроконтроллера AT90S8515.

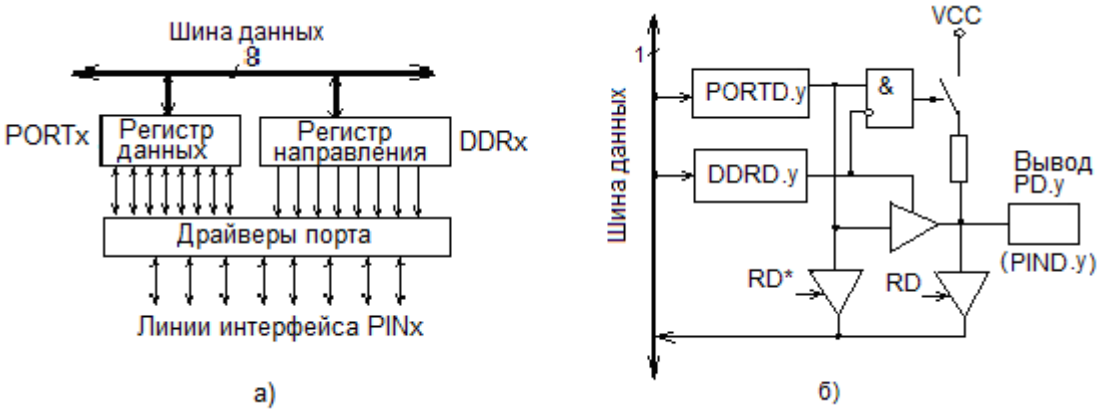


Рис.1. Структура порта Px (а) и схема одного разряда порта PD (б)

Состояние разряда DDRx.y определяет направление передачи бита данных через вывод порта Px.y. При DDRx.y=0 вывод порта Px.y является входом, при DDRx.y=1 – выходом.

В режиме входа состояние разряда PORTx.y определяет состояние вывода Px.y. При PORTx.y=1 вывод порта через внутренний резистор подключается к шине питания Vcc. При PORTx.y=0 резистор отключен, вывод Px.y находится в высокоимпедансном состоянии (Z-состояние).

В режиме выхода состояние разряда PORTx.y определяет значение сигнала на выводе Px.y. При PORTx.y=0 на выводе устанавливается напряжение низкого уровня, при PORTx.y=1 - напряжение высокого уровня.

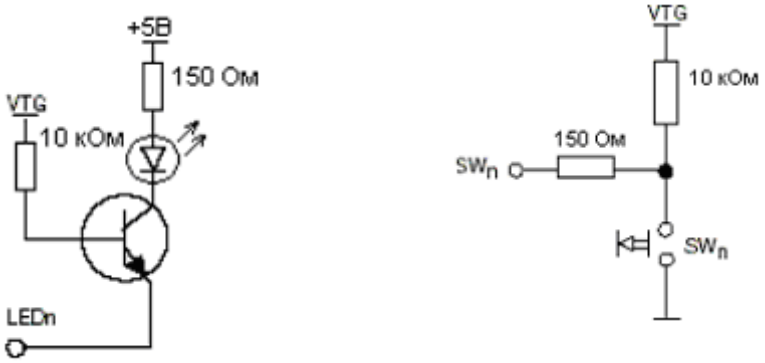
При пуске и перезапуске микроконтроллера все разряды регистров DDRx и PORTx сбрасываются в нулевое состояние, вследствие чего выводы портов работают в режиме входа и находятся в Z – состоянии.

При совместном использовании всех разрядов порта для ввода байта данных используются команды с мнемоникой IN Rd, PINx, для вывода - OUT PORTx, Rr (d, r = 0-31).

Значение выходного сигнала на отдельном выводе порта можно задать с помощью команд установки «0» (CBI PORTx,y) и установки «1» (SBI PORTx,y). Значение входного сигнала на отдельном выводе порта можно проверить, используя команды условного перехода SBIC PINx,y или SBIS PINx,y которые предусматривают пропуск следующей команды по нулевому или по единичному значению PINx.y.

Взаимодействие микроконтроллера с кнопками и светодиодами

Стартовый набор STK500 содержит 8 желтых светодиодов и 8 кнопок без фиксации общего назначения. Схема одного разряда индикатора изображена на рис. 2,а. При поступлении на вывод LEDn сигнала с низким уровнем напряжения (логический «0») светодиод светится, сигнала с высоким уровнем напряжения (логическая «1») – светодиод гаснет.



а) б)

Рис.2. Схема включения светодиода а) и кнопки б)

Схема подключения кнопки изображена на рис.2,б. При нажатии на кнопку - на выводе SWn низкий уровень напряжения (GND), а при отпускании - высокий (VTG). Рабочий диапазон напряжения VTG =1.8…6.0 В.

Выводы светодиодов LEDx и кнопок SWx (x=0÷7) соединены с соответствующими контактами разъемов SWITCHS и LEDS. Следует иметь в виду, что контакты разъемов 8 и 9 использованы для общей шины GND и питания VTG. Поэтому необходимо соблюдать соединение шлейфами одноименных выводов разъемов индикаторов и кнопок с портами микроконтроллеров (шлейф не должен перекручиваться). Для ориентира в шлейфе красным цветом выделена одна из линий, которая должна соединять одноименные выводы разъемов (например, выводы с номером 0).

Особенности работы светодиодов и кнопок необходимо учитывать при программировании портовых операций микроконтроллеров, связанных с обращением к светодиодам и кнопкам.

Ниже приведена программа, которая при нажатии кнопки START выполняет поочередное переключение светодиодов LEDS, при нажатии кнопки STOP переключение останавливается и возобновляется при повторном нажатии кнопки START. В программе 1.1 линии порта PB использованы для индикации и, следовательно, проинициализированы на вывод, а линии 0 и 1 порта PD, соединяемые с кнопками, - на ввод. Затем ожидается нажатие кнопки START, после чего начинается последовательное переключение светодиодов с задержкой и проверка состояния кнопки STOP.

При программировании микроконтроллера AT90S8515 используется файл определений 8515def.inc, ATmega8515 - m8515def.inc.

Программа 1.1

```
;*********************************************************************
;Программа 1.1 для микроконтроллеров ATx8515:
;переключение светодиодов (СД) при нажатии на кнопку START (SW0),
;после нажатия кнопки STOP (SW1) переключение прекращается и
;возобновляется с места остановки при повторном нажатии на кнопку START
;*********************************************************************
;.include "8515def.inc"          ;файл определений для AT90S8515
.include "m8515def.inc"          ;файл определений для ATmega8515
.def temp = r16                  ;временный регистр
.def reg_led = r20               ;регистр состояния светодиодов
.equ START = 0                   ;0-ой вывод порта
```

```
.equ STOP = 1                   ;1-ый вывод порта
.org $000
          rjmp init
;***Инициализация***
INIT:     ldi reg_led,0xFE      ;сброс reg_led.0 для включения LED0
          sec                   ;C=1
          set                   ;T=1 – флаг направления
          ser temp              ;инициализация выводов
          out DDRB,temp         ; порта PB на вывод
          out PORTB,temp        ;погасить СД
          clr temp              ;инициализация
          out DDRD,temp         ; порта PD на ввод
          ldi temp,0x03         ;включение 'подтягивающих'
          out PORTD,temp        ; резисторов порта PD (0-й, 1-й разряды)
WAITSTART:                      ;ожидание
          sbic PIND,START       ; нажатия
          rjmp WAITSTART        ;  кнопки START
LOOP:     out PORTB,reg_led     ;вывод на индикаторы
;***Задержка (два вложенных цикла)***
          ldi r17,2
d1:       ldi r18,2
d2:       dec r18
          brne d2
          dec r17
          brne d1

          sbic PIND,STOP        ;если нажата кнопка STOP,
          rjmp MM               ; то переход
          rjmp WAITSTART        ; для проверки кнопки START
MM:       brts LEFT             ;переход, если флаг T установлен
          sbrs reg_led,0        ;пропуск следующей команды,
                                ; если 0-й разряд reg_led установлен
          set                   ;T=1 - переключение флага направления
          ror reg_led           ;сдвиг reg_led вправо на 1 разряд
          rjmp LOOP             ;переход на проверку нажатия STOP
```

```
LEFT:     sbrs reg_led,7        ;пропуск следующей команды,
                                ; если 7-й разряд reg_led установлен
          clt                   ;T=0 – переключение флага направления
          rol reg_led           ;сдвиг reg_led влево на 1 разряд
          rjmp LOOP
```

Практическая часть

Задание 1. Проверить работу вышеприведенной программы в шаговом режиме работы с помощью симулятора AVR Studio 4. Симуляция замыкания и размыкания кнопок START и STOP осуществляется путем установки «0» (белый цвет) и «1» (черный цвет) в маленьких квадратиках порта линий интерфейса PIND. Перед началом прогона программы установите для обоих кнопок PIND0, PIND1 состояние логической «1» (кнопки отжаты).

Убедившись в правильной работе программы, измените параметры циклов задержки, чтобы длительность задержки составила 0,5с. Проверьте время задержки с помощью симулятора. Для этого установите контрольные точки (*Debug/ Toggle Breakpoint*) перед началом выполнения программного блока задержки и после выхода из него. Запустив программу в режиме прогона (*Debug/Run*) с остановом в контрольных точках, оцените время задержки, контролируя либо показания счетчика циклов *Cycle Counter* в окне *Workspace* AVR Studio 4 (вкладка I/O, секция Processor), либо показания *Stop Watch*.

Выполнив трансляцию программы, загрузите hex-файл в STK500. При программировании STK500 следите, чтобы тип целевого микроконтроллера, установленного на используемой плате, совпадал с типом микроконтроллера в поле Program. В процессе программирования в окне STK500 внизу появляются сообщения о ходе загрузки программы.

Убедившись в правильности загрузки по выводимым сообщениям, проверьте работу программы на макете. Для этого, выключив питание STK500, с помощью проводов соединить выводы разъема порта PD с выводами кнопочного разъема SW. С помощью 10-проводного шлейфа соединить выводы разъема порта PB с выводами разъема светодиодов. Включить питание и проверить работу загруженной программы, поочередно нажимая кнопки START и STOP.

Задание 2. Проверить работу программы в среде VMLab. Для этого, запустив программу VMLab, создать проект, используя инструкции и рекомендации, приведенные в приложении.

В окне проекта, кроме типовых директив, определяющих частоту работы микроконтроллера, имя файла с исходным текстом программы и др., добавить директивы для описания связей

виртуальных кнопок контрольной панели k0, k1 с входами порта PD0, PD1, светодиодов d1…d8 с выходами порта PB и резисторов, используя для этого шаблоны компонентов. Выполнив компиляцию проекта, запустить программу командой *Run*. Нажимая кнопки k0 (START) и k1 (STOP), проконтролировать работу программы по состояниям светодиодов контрольной панели.

Внести в проект директивы, обеспечивающие контроль работы с помощью логического анализатора (*Scope*), указав все выходы порта PB. Настроить *Scope* так, чтобы увидеть полный цикл осциллограммы, уменьшив для этого время задержки.

Задание 3. По заданию преподавателя изменить программу для переключения светодиодов в заданной последовательности. При написании программы на языке ассемблера можно воспользоваться системой команд микроконтроллеров AVR, приведенной в /1,2/.

Таблица 1. Таблица вариантов к лабораторной работе №1.

№	Последовательность переключения светодиода на линейке (включенного светодиода - ВКЛ, выключенного – ВЫКЛ)	Порт индикации	Время переключения, мс
1	Непрерывно, перемещая один ВКЛ светодиод, начиная со старшего 7 разряда, вправо до 0 и обратно	PA	200
2	Непрерывно, перемещая один ВЫКЛ светодиод, начиная с младшего 0 разряда, влево до 7 и обратно	PB	300
3	Непрерывно, перемещая один ВЫКЛ светодиод по четным разрядам, начиная с 6 разряда, вправо до 0 и обратно	PC	250
4	Непрерывно, перемещая один ВКЛ светодиод по нечетным разрядам, начиная с 7 разряда, вправо до 1 и обратно	PA	200
5	Непрерывно, перемещая один ВКЛ светодиод по четным разрядам, начиная с 0 разряда, влево до 6 и обратно	PB	300
6	Непрерывно перемещая один ВЫКЛ светодиод по нечетным разрядам, начиная с 1 разряда, влево до 7 и обратно	PC	250
7	Непрерывно, увеличивая количество ВКЛ светодиодов (начальное состояние – все ВЫКЛ), начиная с 0 разряда, затем уменьшая в обратном направлении	PA	200
8	Непрерывно, уменьшая количество ВКЛ светодиодов (начальное состояние – все ВКЛ), начиная с 7 разряда, затем увеличивая в обратном направлении	PB	300

9	Непрерывно, увеличивая количество ВЫКЛ светодиодов (начальное состояние – все ВКЛ), начиная с 0 разряда, затем уменьшая в обратном направлении	PC	250
10	Непрерывно, уменьшая количество ВЫКЛ светодиодов (начальное состояние – все ВЫКЛ), начиная с 7 разряда, затем увеличивая в обратном направлении	PA	200
11	Непрерывно, перемещая один ВКЛ светодиод, начиная с 7 разряда вправо до 0, затем снова с 7 до 0 и т.д.	PB	300
12	Непрерывно, перемещая один ВЫКЛ светодиод, начиная с 0 разряда влево до 7, затем снова с 0 до 7 и т.д.	PC	250
13	Непрерывно, перемещая два ВКЛ светодиода, начиная с 6 разряда, вправо до 1 и обратно	PA	200
14	Непрерывно, перемещая два ВЫКЛ светодиода, начиная с 2 разряда, вправо до 5 и обратно	PB	300
15	Непрерывно, перемещая два ВКЛ светодиода, начиная с 2 разряда, влево до 6 и обратно	PC	250
16	Непрерывно, перемещая два ВЫКЛ светодиода, начиная с 6 разряда, вправо до 2 и обратно	PA	200
17	Непрерывно, перемещая один ВКЛ светодиод, начиная с 7 разряда вправо до 0, затем один ВЫКЛ светодиод снова с 7 до 0 и т.д.	PB	300
18	Непрерывно, перемещая один ВЫКЛ светодиод, начиная с 0 разряда влево до 7, затем один ВКЛ светодиод снова с 0 до 7 и т.д.	PC	250
19	Непрерывно, перемещая два ВКЛ светодиода, начиная с 7 разряда, вправо до 0, и в обратном направлении два ВЫКЛ светодиода	PA	200
20	Непрерывно, перемещая два ВЫКЛ светодиода, начиная с 0 разряда, влево до 7, и в обратном направлении два ВКЛ светодиода	PB	300
21	Непрерывно, перемещая один ВКЛ светодиод, начиная с 0 разряда, влево до 7, и в обратном направлении два ВЫКЛ светодиода	PC	250
22	Непрерывно, перемещая два ВЫКЛ светодиода, начиная с 7 разряда, вправо до 0, и в обратном направлении один ВКЛ светодиод	PA	200
23	Непрерывно, перемещая три ВКЛ светодиода, начиная с 1 разряда, влево до 6, и обратно	PB	300
24	Непрерывно, перемещая три ВЫКЛ светодиода, начиная с 6 разряда, вправо до 1, и обратно	PC	250
25	Непрерывно, перемещая один ВКЛ светодиод по четным разрядам, начиная с 6 разряда, вправо до 0 и обратно один ВЫКЛ светодиод по нечетным	PB	300

	разрядам, начиная с 1		
26	Непрерывно перемещая один ВЫКЛ светодиод по нечетным разрядам, начиная с 7 разряда, вправо до 1 и обратно один ВКЛ светодиод по четным разрядам, начиная с 0 разряда	PC	200

Отладив программу с помощью симулятора AVR Studio, загрузить ее в микроконтроллер и проверить ее работу на плате STK500.

Задание 4. Проверить работу подготовленной программы в VMLab. Запротоколировать работу программы в виде «скриншота» осциллограммы.

Оформление отчета

Отчет должен содержать:

а) базовую программу со схемой алгоритма и временем задержки 0.5 с, а также запротоколированную оценку времени задержки, полученную в результате симуляции;

б) осциллограмму в виде скриншота из VMLab, иллюстрирующую работу порта управления светодиодами;

в) индивидуальное задание на последовательность включения/выключения светодиодов и соответствующий заданию текст разработанной программы;

г) осциллограмму работы выходного порта (из VMLab) согласно индивидуальному заданию.

Требования при защите: уметь отвечать на вопросы по программе и объяснять наблюдаемые осциллограммы.

Контрольные вопросы

1. Какие из регистров порта в среде AVR Studio 4 являются аналогами физических, какой является виртуальным?

2. Что означает понятие ‘подтягивающий резистор’? Каким образом можно им управлять?

3. Как задается функция ввода или вывода для разряда порта?

4. Каким сигналом (логического «0» или логической «1») можно включить светодиод на плате STK500?

5. Какой сигнал (логический «0» или логическая «1») эмулирует в AVR Studio 4 замыкание кнопки? Как должен выглядеть бит порта PINx.y в AVR Studio 4 при замыкании (размыкании) кнопки?

6. Как можно изменить программную задержку, например, для уменьшения погрешности?

7. Рассчитать максимальную временную задержку, которую можно получить с помощью одного программного цикла при частоте работы микроконтроллера 1МГц, 4МГц или 8 МГц, используя в качестве счетчика циклов 8-разрядный регистр.

Как изменится процедура и величина задержки при использовании в качестве счетчика циклов регистровой пары? Как изменится задержка при организации двух вложенных циклах?

Рекомендуемая литература

1. Инструментальные средства AVR Studio 4 и STK500. Электронный файл.
2. В.Я. Хартов Микроконтроллеры AVR. Практикум для начинающих. Издательство МГТУ им. Н.Э. Баумана. 2012 г.
3. Приложение. Интегрированная отладочная среда VMLab.

Приложение

Интегрированная отладочная среда VMLab

При отсутствии стартового набора STK500 или более нового STK600 можно воспользоваться средой проектирования Visual Micro Lab (VMLab). Эта среда предназначена для отладки проектов, содержащих микроконтроллер и несложную периферию. Разработку и отладку проектов можно осуществлять на языках Си и ассемблера. В сети Internet на сайте *http://www.amctools.com* представлены различные версии программы, среди них свободно распространяемая версия, удобная для симуляции учебных проектов. Программа поддерживает архитектуру микроконтроллеров AVR и ST6 и имеет встроенный компилятор для компиляции программ на языке ассемблера.

При разработке программ на языке Си необходимо установить внешний пакет программ для компиляции программ, например CodeVisionAVR или WinAVR. Последний обеспечивает поддержку пакета утилит компиляции GCC (GNU compiler collection/GNU C compiler). Основным требованием к компилятору является возможность генерации выходных файлов в форматах .coff и .hex.

Создание проекта на ассемблере и подготовка проектного файла

Достоинством VMLab является возможность построения виртуальной окружающей микроконтроллер среды, состоящей из простых логических и электронных элементов. Файлом,

задающим функционирование виртуальной модели, является файл проекта *name* с расширением .prj. В нём описываются тип и параметры микроконтроллера, частота работы, его внешние связи, используемые отладочные средства, файлы с исходным кодом программы, контрольные точки (узлы) схемы. В проект добавляют виртуальные компоненты схемы и описывают их согласно электрической схеме проекта.

Файл проекта содержит ряд директив, в которых перечислены свойства проекта. Все директивы можно разделить на три группы: общие, директивы ассемблера и директивы компилятора Си. Некоторые из них описаны ниже.

.MICRO «Attiny15» − определяет тип используемого микроконтроллера;

.CLOCK 8meg − указывает частоту работы ядра микроконтроллера в начале симуляции. В дальнейшем во время симуляции частота может быть изменена через панель управления;

.PROGRAM ″test.asm″ − определяет основной исходный файл проекта на языке ассемблера. Данная директива несовместима с директивой .SOURCE;

.TARGET ″test.hex″ − задает имя выходного бинарного файла, который используется симулятором при отладке. Директива используется при компиляции проекта из нескольких файлов с помощью обобщенной цепочки компиляции или при использовании цепочки компиляции GCC. При компиляции одного файла на ассемблере по умолчанию генерируется выходной файл с тем же именем;

.TRACE − указывает на необходимость отслеживать состояние выводов и узлов, перечисленных в директиве .PLOT;

.POWER VDD=5 VSS=0 − указывает уровни напряжений на шинах питания и земли. Напряжения могут быть положительными и отрицательными;

.PLOT V(node_1) V(PA0) V(PB0) − указывает, сигналы каких узлов следует контролировать при симуляции. Возможно использование нескольких директив;

.TOOLCHAIN «ASM» − указывает на то, как будет компилироваться исходный код программы. Возможны три варианта: ASM – исходный код на языке ассемблера, GCC – исходный код на языке Си будет компилирован с помощью предварительно установленного пакета компиляции GCC, GENERIC – цепочка компиляции определяется пользователем;

.SOURCE «file1.c» «file2.c» «file3.c» − используется при отладке программы на языке Си для указания исходных файлов. Директива несовместима с директивой .PROGRAM.

Файл проекта имеет простой текстовой формат, который редактируется вручную без использования шаблонов. Синтаксис и параметры директив описаны в файле помощи (*Help*).

Для эмуляции устройств, входящих в микроконтроллерную систему, используют описания компонентов системы путем добавления в файл проекта шаблонов электронных компонентов и заполнения их именами узлов, соединяющих эти компоненты с микроконтроллером и между собой. Вызов шаблонов осуществляется из главного меню программы (*Components*).

При создании виртуальной модели в файле проекта можно указывать:

– простые внешние компоненты, такие как резисторы, конденсаторы, светодиоды, кнопки и переключатели;

– логические элементы И-НЕ/ИЛИ-НЕ, компараторы, генераторы сигналов;

– аналого-цифровые и цифро-аналоговые преобразователи;

– терминалы.

Используемые компоненты описываются в файле проекта в формате:

<тип> <имя> <узлы соединения> <параметры>.

Пример 1. Для описания схемы соединения вывода PB0 микроконтроллера со светодиодом на контрольной панели d1 через ограничительное сопротивление R1 и вывода PD0 с кнопкой k0 в проектный файл необходимо внести описания:

```
d1 vdd a1                    ; светодиод d1 контрольной панели
R1  a1 pb0 560               ; сопротивление 560 Ом
k0 pd0 gnd monostable (10m)  ; кнопка k0 контрольной панели с временем замыкания 10 мс
```

Пример 2. Шаблон описания виртуального терминала RS-232 имеет вид:

```
X[inst_name]  TTY(baud_rate [n_bits] [parity] [odd_parity] [n_stop_bits] [rx_display_as])
                  node_tx node_rx,
```

где TTY– список параметров последовательного канала: скорость (указывается всегда), количество передаваемых бит, отсутствие(0) или наличие (1) бита четности, число стоповых бит (1 или 2), способ представления передаваемых символов в окне приемника терминала (ASCII (1 или 2), десятичный (3), шестнадцатеричный (4));

node_tx – узел, соединяемый с выходом передатчика терминала;

node_rx – узел, соединяемый с входом приемника терминала.

Например, при подключении терминала к линиям передатчика PD1 и приемника PD0 микроконтроллера и передаче данных со скоростью 9600 бит/с описание терминала можно представить в виде:

```
Xmyterm TTY (9600  8  0 0  1  4)  PD0  PD1
```

Во избежание ошибок при описании компонентов моделируемой схемы рекомендуется предварительно нарисовать электрическую схему и указать на ней имена узлов всех соединений.

После запуска программы VMLab создается новый проект (*Project/New Project*). После нажатия кнопки *Enter Name/ Browse/ Create directory* в открывающемся окне можно создать директорию (папку) для проекта. Открыв ее, указывают имя проекта *name*.

Из выпадающего списка *Select Micro* выбирают тип микроконтроллера. Добавляют в проект подготовленный файл программы с расширением .asm. При запросе на копирование файла в папку проекта выполняют копирование. Закрывают окно нажатием на кнопку *OK*. В окне *Code* программы с помощью директивы указывают путь к файлу определений микроконтроллера, например *.include "c:\vmlab\include\8515def.inc*". Синтаксически правильную программу на языке ассемблера можно подготовить непосредственно в среде VMLab, воспользовавшись встроенным компилятором. В окне проекта *name.prj* вводят директивы со свойствами проекта и добавляют описания компонентов схемы.

Подготовив программу, выполняют ее компиляцию. Программой VMLab предусмотрены собственные два режима компиляции, выполняемых командами из меню *Project* (*Build* и *Rebuild*). Далее выполняется отладка в среде VMLab.

Отладка проектов в среде VMLab

После подготовки проектного файла и компиляции наступает самый главный этап реализации проекта – его отладка.

Инструменты отладки вызываются из меню *View*. Традиционными для всех симуляторов являются окна, представляющие внутренние ресурсы микроконтроллера: регистры (*Registers/Flags*), память данных (*Data Memory*), энергонезависимая память данных (*EEPROM*), память программ (*Program Memory*), порты ввода/вывода (*I/0 ports*) и периферийные устройства микроконтроллера (*Peripherals*).

Одним из важных инструментов является окно контрольной панели *Control Panel*. Данное окно предоставляет пользователю интерфейс взаимодействия с отлаживаемой программой – кнопки, светодиоды, окно терминала и т.д. На рис. 3 показан вид контрольной панели с двумя расположенными внизу окнами последовательного канала.

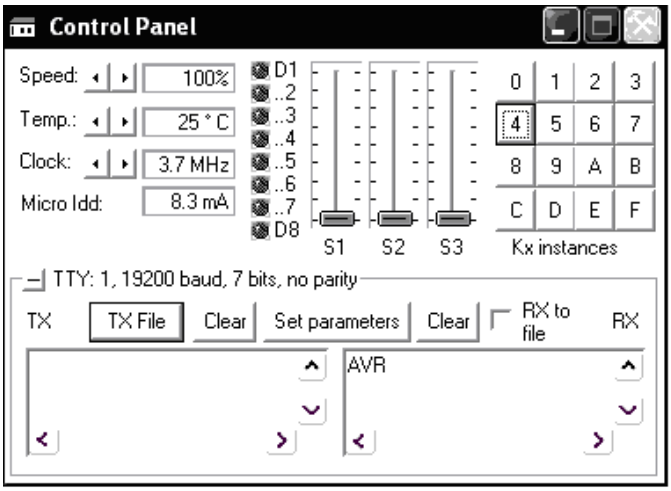


Рис. 3. Окно контрольной панели

Не менее полезным инструментом отладки является виртуальный осциллограф *Scope* (он же логический анализатор), открываемый командой *View/Scope*. Выбор контрольных точек для вывода временных диаграмм определяется директивой .PLOT проектного файла. В окне *Scope* можно изменять масштаб развертки по времени (вдоль горизонтальной оси) и амплитуде (по вертикальной).

Отладка проекта ведется по исходному коду в окне *Code*, в котором желтая полоска, указывает на частоту обращения к каждой отдельно взятой строке программы.

Окно *Code* является центральным при отладке проекта. В нем отображаются все редактируемые файлы проекта, а из контекстного меню можно командой *Go to Program Memory* открыть окно программной памяти и производить прогон программы до заданной точки (команда *Run to cursor*).

Процесс отладки запускается после компиляции проекта из меню командой *Run*. Для управления состоянием программы используется отдельная панель управления с кнопками, нажатие которых позволяет запускать и останавливать программу, а также выполнять ее в различных режимах:

- *Go/Continue* – позволяет запускать программу в режиме реального времени. В данном режиме программа работает только под управлением внешних воздействий и останавливается на точках прерывания;
- *Step over* – позволяет выполнить данную команду за один шаг. Если это обычная команда, то после ее выполнения отладка останавливается. Если это инструкция вызова, то отладчик выполнит процедуру до конца и остановится на команде, следующей за командой вызова;

- *Step into* – режим пошаговой отладки. Отладчик выполняет одну команду и останавливается;
- *Stop* – выполнение данной команды приводит к приостановке выполнения программы для последующей пошаговой отладки или для более детального просмотра параметров микропроцессора;
- *Restart (light)* – данная команда аналогична выполнению перезагрузки микроконтроллера. Текущее время симуляции устанавливается в 0, программный счетчик PC сбрасывается, но вся периферийная часть остается в состоянии, как и до сброса, содержимое памяти не изменяется;
- *Restart (deep)* – данная команда аналогична полному сбросу микроконтроллера. Следует заметить, что при рестарте микроконтроллера все ячейки в окне памяти получают значение, обозначаемое символами "??". Это позволяет определить ячейки, в которые производилась запись данных, и ячейки неинициализированной области, откуда производилось чтение. Измененные со времени прошлого обращения программы значения ячеек подсвечиваются желтым цветом.

Чтобы остановить симуляцию программы в нужном месте программы, в VMLab используется общеизвестный механизм прерываний в заданных точках останова (*Break Point*). В окне *Code* слева от исполняемых строчек текста программы везде, где можно поставить точку останова, появляются зеленые квадратики. Если щелкнуть по такому квадратику, он станет красным восьмиугольным знаком останова. При запуске программы путем нажатия в панели инструментов на кнопку светофора произойдет останов программы на строке с точкой останова. Счетчик времени работы *Time*, расположенный внизу окна программы, останавливается. При этом строка в окне *Code* подсветится, а в окне *Messages* появится сообщение о состоянии программного счетчика PC, времени, прошедшем с начала программы, и причине останова.

Точки останова можно устанавливать и удалять в любом месте до начала симуляции и во время симуляции. Следует, однако, иметь в виду, что при изменении симулируемых файлов точки останова не сохраняются. Потребуется их повторная установка.

Одним из наиболее удобных инструментов отладки является окно переменных *Watches*. Для его настройки имеется специальный мастер (*Debug/Watch Manager*), с помощью которого можно добавлять или удалять переменные и, что тоже удобно, ставить условия останова программы при чтении (R) или записи (W) переменных.

Для наблюдения по ходу работы программы за значениями используемых в программе переменных нужно открыть окно, выполнив команду *Window/Watch*. Щелкнув в окне правой кнопкой мыши, можно открыть окно настройки переменных *Watch Manager*. В нем помечают переменные, за которыми устанавливается наблюдение, и способ отображения их содержимого. Пока программа не запущена, вместо значений переменных стоят вопросительные знаки. После пуска программы при остановах наблюдаются текущие значения переменных.

Для выполнения отладки запускают программу VMLab и открывают созданный проект (*Project/Open Project*). Для этого перейдя в папку проекта, выбирают файл проекта для VMLab (*name.prj*) и открывают его кнопкой *Открыть*. В меню *View* подключают необходимые инструменты для отладки проекта: контрольную панель (*Control Panel*), виртуальный запоминающий осциллограф *(Scope*), порты ввода/вывода (*I/O*), память данных и др. Открыв через меню *Window/Code* окно *Code* (обычно оно открывается сразу при открытии проекта), можно увидеть текст симулируемой программы. В меню *Project* выполняют *Re-Build all ...* и приступают к отладке программы, нажимая кнопку светофора на панели инструментов и наблюдая за окнами *Scope, Control Panel* и *Code*. При обнаружении ошибок в работе программы останавливают симуляцию, нажав кнопку *Stop*. Выполняют полный рестарт командами *Run/Restart(deep)*. Вносят изменения в код программы. После повторной компиляции и сборки проекта возобновляют отладку программы.