Хартов В.Я.

Методические указания
Для проведения лабораторных работ по курсу
Микропроцессорные системы

Программирование и отладка программ на языке Си

При программировании микроконтроллеров всё большую популярность приобретает язык Си. При его использовании сокращается время на разработку, что особенно заметно при написании больших программ, обеспечивается их переносимость на другие платформы. Недостатком языка Си по сравнению с языком Ассемблера является больший объем кода и, как следствие, более низкая скорость работы. Однако, благодаря тому, что в архитектуре AVR изначально заложено эффективное декодирование и исполнение инструкций, генерируемых компиляторами, после компиляции Си - программ получается высокопроизводительный код.

Существует ряд компиляторов Си, поддерживающих архитектуру AVR: CodeVisionAVR, ImageCraft C, AVR GCC и др., которые включают в себя широкий набор библиотек для работы с периферийными устройствами и поддерживают форматы объектных файлов, используемых при отладке в AVR Studio. Кроме того, многие компиляторы поддерживают возможность программирования микроконтроллеров после компиляции исходного текста программы.

Ниже описан процесс разработки и отладки программы на языке Си для учебного проекта. В качестве среды программирования использована программа CodeVisionAVR версии 1.23.7a, для отладки - AVR Studio 4.

Среда CodeVisionAVR

Программа CodeVisionAVR фирмы HP InfoTech – это интегрированная среда разработки, содержащая компилятор языка Си, графическую оболочку, автоматический генератор программ и встроенный программатор, ориентированные на работу с семейством микроконтроллеров AVR.

Наряду со стандартными библиотеками языка Си и системой справок по языку компилятор имеет библиотеки для работы с периферийными устройствами (ЖКИ -

индикаторами с встроенными контроллерами, датчиками температуры, часами реального времени, энергонезависимой памятью EEPROM, шиной SPI и др.). Также имеется автоматический генератор программ для инициализации внутренних и периферийных ресурсов микроконтроллера: портов, системы прерываний, встроенного АЦП, таймеров, ЖКИ - индикатора и т.д. Для систем, использующих последовательную передачу данных (RS232, RS422, RS485), имеется встроенный в компилятор буферTerminal.

Генерируемые объектные файлы COFF позволяют осуществлять с помощью отладчика AVR Studio отладку программы непосредственно в коде Си.

Внешний вид окна программы CodeVisionAVR показан на рис.1.

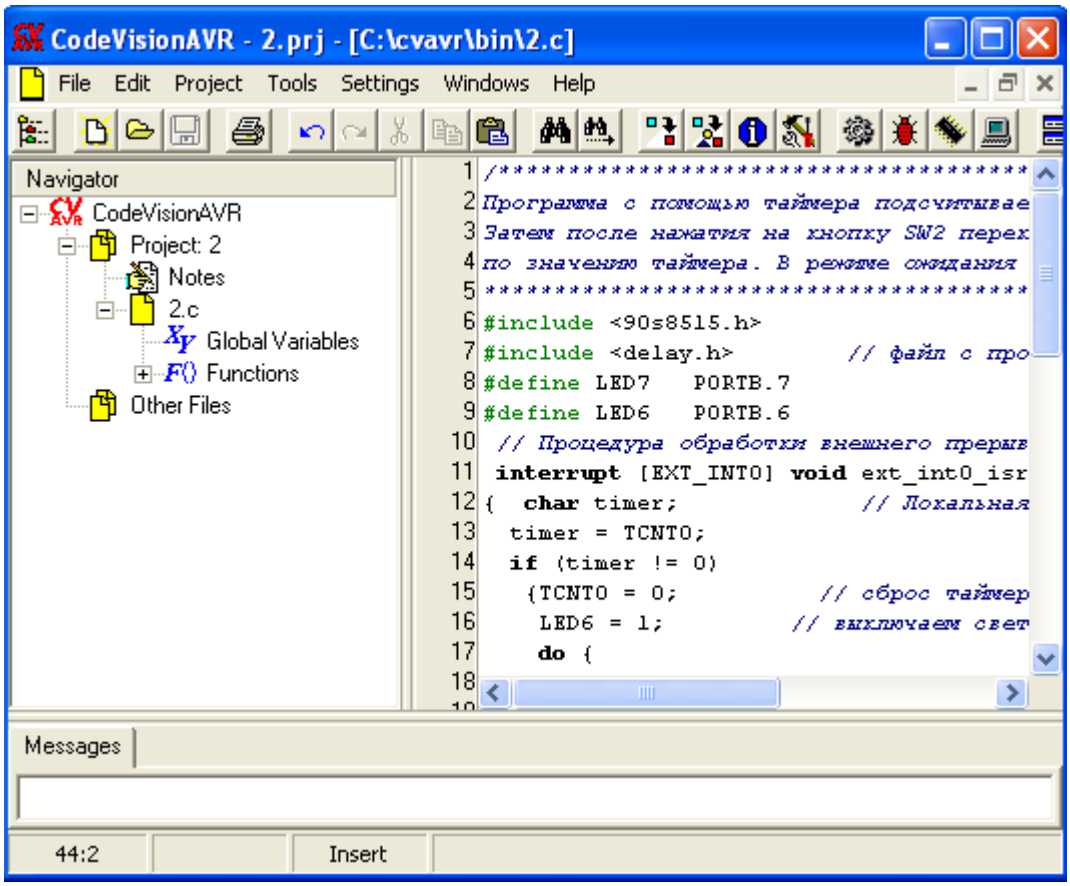

Рис. 1. Окно программы CodeVisionAVR

Создание проекта

Разработаем микроконтроллерное устройство, управляющее 2-я светодиодами, один из которых показывает готовность к работе, второй переключается по числу нажатий кнопки

управления. Проект предполагает работу с портами ввода-вывода, таймером, обработку прерывания, работу с подпрограммой, энергосберегающий режим работы МК.

Схема устройства приведена на рис.2.. В ней предусмотрены две кнопки. Кнопкой SW0 задают число миганий, кнопкой SW2 запускают процесс мигания светодиода LED7. Для подсчета числа нажатий на кнопку SW0 необходим счетчик, в качестве которого используем таймер T0 в режиме подсчёта внешних событий. Для индикации используем два светодиода: SW6 для информирования о готовности схемы и LED7 для мигания. Таким образом, микроконтроллер должен иметь возможность обработки внешнего прерывания от кнопки SW2, таймер с входом внешних событий от кнопки SW0 и две линии порта для управления светодиодами. Выберем микроконтроллер AT90S8515, частоту 1МГц. Кнопки считаем идеальными без эффекта «дребезга контакта». “Подтягивающие” резисторы на входах PD2, PB0 подключаются при программировании режима работы портов.

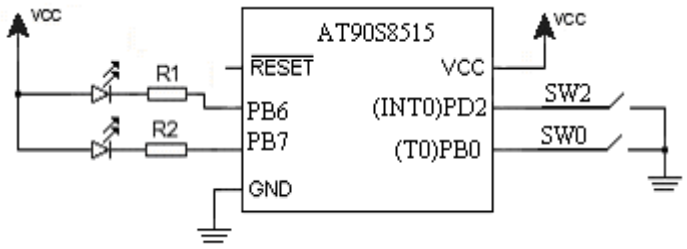

Рис.2. Схема устройства

Работа над проектом в CodeVisionAVR начинается с выбора команды меню *File /New*. В диалоговом окне выбираем *Project* и нажимаем *OK*. Дальнейшие действия зависят от выбора *No* или *Yes* в появившемся окне. Если не использовать автоматический генератор программ, выбираем *No,* выполняя затем последовательность действий п.1, в противном случае выполняем действия согласно п.2.

1. В окне *Create New Project* задаем имя проекта. После сохранения появляется окно *Configure Project* (рис. 3), в котором на вкладке *Files* можно добавить существующие файлы в проект или удалить их из проекта. На вкладке *C Compiler* указывается тип микроконтроллера

(*Chip*), частота его работы (*Clock*), размер доступной памяти (*SRAM*), а также параметры компиляции: метод оптимизации кода – размер (*Size*) или скорость (*Speed*), соглашения по компиляции (*Code Generation*) и форматы выходных файлов (*File Output Formats*), использование буфера Terminal.

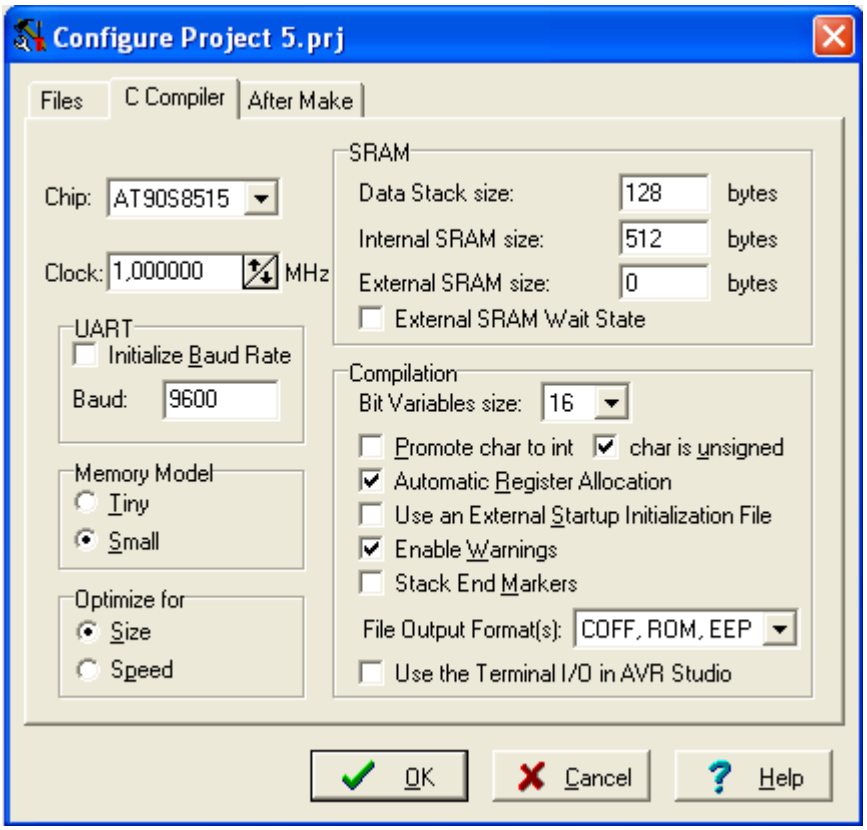

Рис. 3. Окно компилятора Си

Для моделирования проекта устанавливаем тип МК, частоту, оптимизацию по размеру кода, формат COFF выходного объектного файла, после чего нажимаем ОК. Пустой текстовый файл готов для ввода программы.

2. В окне *CodeWizardAVR* на соответствующих вкладках задаём имя проекта, параметры внутренних ресурсов микроконтроллера: его тип, рабочую частоту, параметры порта ввода/вывода, внешнее прерывание по низкому уровню, переключение таймера/счётчика по перепаду из 1 в 0 на входе T0. Выбираем команду меню *File /Generate, Save and Exit*. Три раза указываем в окнах *Save…* имя проекта. В итоге открывается файл-заготовка со строками инициализации требуемых ресурсов.

Для задания параметров проекта в целом необходимо выбрать команду меню *Project /Configure*. Дальнейшие действия так же, как в п.1.

Вводим программу, текст которой представлен ниже:

Программа 1

```
/*********************************************************************
Программа 1 с помощью таймера подсчитывает число нажатий на кнопку SW0.
Затем после нажатия на кнопку SW2 переключает светодиод LED7
по значению таймера. В режиме ожидания включен светодиод LED6.
*********************************************************************/
#include <90s8515.h>
#include <delay.h>          // файл с процедурами задержки
#define LED7   PORTB.7
#define LED6   PORTB.6
// Процедура обработки внешнего прерывания
interrupt [EXT_INT0] void ext_int0_isr(void)
{ char timer;              // Локальная переменная
  timer = TCNT0;
  if (timer != 0)
   {TCNT0 = 0;         // сброс таймера/счётчика
    LED6 = 1;          // выключаем светодиод LED6
    do {
      LED7 = 0;
      delay_ms(500);  // задержка
      LED7 = 1;
      delay_ms(500);
         } while (--timer != 0);
    LED6 = 0;          // включаем светодиод LED6
   }
}
void main(void)
{
  // инициализация портов
```

```
  DDRB=0xC0;     //PB7,PB6 для LED7,LED6
  PORTB=0x81;    //PB0(SW0) - ввод событий
  DDRD=0xFB;     //PD2(SW2) для индикации
  PORTD=0xFF;
  // инициализация таймера 0
  TCCR0=0x06;
  TCNT0=0x00;
  GIMSK=0x40;      // инициализация прерывания INT0
  MCUCR=0x20;      // разрешение перехода в режим Idle
  #asm("sei");     // глобальное разрешение прерываний
  for (;;) {
      #asm("sleep");     // переход в режим Idle
      #asm("nop");
            }
}
```

Компиляция

Для компиляции программы необходимо выбрать команду меню *Project /Compile (F9)*. Результаты компиляции выводятся в окно *Information*, а в окне *Navigator* в дереве проекта появляются синие ветви. Щелкая по значкам можно перемещаться по заголовкам процедур и именам глобальных переменных, что удобно при работе с большим проектом.

Ветвь дерева с ошибками (Errors) выделена красным цветом. Список ошибок также выводится в окне *Messages* внизу экрана. Чтобы исправить ошибку, нужно щелкнуть по листу дерева. Соответствующая строка программы выделится серым цветом, а рядом с ней появится курсор для ввода.

После компиляции будут созданы файлы с расширениями: .asm (файл Ассемблера), .map (адреса ОЗУ расположения глобальных переменных), .vec (список векторов прерываний), .inc.

В случае успешной компиляции выводится сообщение об отсутствии ошибок. В окне *Navigator* ветвь сообщений об ошибках пропадает, а список файлов пополняется файлом _.c - копией исходного файла на Си.

Для окончательной сборки проекта и получения файлов для отладки и программирования МК нужно выбрать в меню команду *Project /Make* (Shift+F9). Появится окно *Information*, в

котором сообщается о создании дополнительных файлов с расширениями .rom (программирование Flash-памяти МК – программа в формате «адрес : слово»), .eep (программирование EEPROM), .lst (листинг программы), .err (аналог

содержимого окна *Information*), .obj (объектный файл) и .cof (символьный файл для отладки в среде AVR Studio).

Работа с CodeVisionAVR на этом заканчивается, по крайней мере, до того момента, когда нужно будет исправить ошибки, найденные при отладке.

Отладка в AVR Studio

1. Запускаем AVR Studio 4, проект создавать не нужно.

2. Открываем объектный файл проекта (Open File) и выбираем фильтр Object Files в окне открытия файлов. AVR Studio 4 поддерживает семь типов объектных файлов. Выбираем файл с расширением .obj, либо с расширением .cof. Открывается окно *Select device and debug platform*, в котором нужно указать платформу для отладки: AVRSimulator и AT9x8515. После этого создаётся проектный файл и загружается отладчик. Дальнейшие действия зависят от выбранного типа файла.

3. Выбираем файл для загрузки в AVR Studio.

Выбрав файл с расширением .obj, отладку можно осуществлять в кодах Ассемблера. Следует обратить внимание на то, что проект содержит 3 файла (.asm, .inc и .vec) и отладка начинается с файла .vec, содержащего векторы прерываний. Курсор отладки (жёлтая стрелка) установлен на команде **rjmp_reset**. После нажатия кнопки *Step Into(F11)* курсор переходит на обработчик сброса, расположенный в файле .asm, и дальнейшая отладка продолжается в обычном режиме. Чтобы пропустить команды, добавленные компилятором, и перейти к отладке кода, написанного программистом, нужно установить курсор на первую команду процедуры main () и нажать кнопку *Run to cursor*. Все команды до курсора будут выполнены. Проект можно просмотреть в дизассемблере, который можно открыть, выбрав в меню *View /Disassembler*. Отладку можно продолжить в этом окне.

Для отладки программы на языке Си выбираем файл с расширением .cof.

4. Устанавливаем параметры отладки в меню *Debug /AVRSimulator Options*: ATx8515, частота 1МГц, протоколирование порта PB с выводом на экран.

5. Контроль содержимого используемых регистров ввода/вывода МК можно осуществить несколькими путями:

а) непосредственно просматривая их содержимое на вкладке I/O панели Workspace,

б) контролируя область памяти регистров ввода/вывода (*View / Memory [I/O]*),

в) присвоив их значения переменным, которые можно в дальнейшем просматривать в окне *Watch.*

В нашем случае имеем одну переменную timer, которая принимает значение таймера/счётчика TCNT0. Открываем окно *Watch*, выбрав команду меню *View /Watch*, и перетаскиваем мышью эту переменную в столбец *Name*. Пока переменная находится вне зоны видимости отладчика, в столбце *Value* будет записано *Not in Scope*. Подготовка к отладке завершена.

6. В начале процесса отладки и в случае нажатия кнопки сброса *Reset* курсор отладки устанавливается на первой строке процедуры *main* Выполнение программы можно контролировать, открыв окно дизассемблера, однако, имея программу на языке Си, значительно удобнее отлаживать программу, используя окно Си-программы.

7. Нажимая кнопку *Step Into(F11)* наблюдаем за изменением содержимого регистров ввода/вывода МК. Одно из преимуществ отладки на Си – это то, что используются только программные инструкции, что способствует ускорению процесса отладки программы. В цикле for (;;) командой Ассемблера **sleep** МК переводится в режим пониженного энергопотребления.

9. Занесём в счётчик TCNT0 значение 2. При разомкнутой кнопке SW0 (состояние «1») эмулируем замыкание кнопки SW2 (состояние «0»), что приводит к вызову обработчика прерывания. При вызове процедуры *delay_ms()* отладчик не входит в неё, но этот вызов задерживает его работу и изменяет значение счетчика циклов.

10. Установив курсор на команде **sleep**, выполним команду *Run to Cursor* (выполнить до курсора). После останова симуляции, что можно обнаружить по желтому индикатору в строке состояния, в окне *Output* получим список сообщений (отчет), в котором значение 0x81(10000001) соответствует готовности схемы (светодиод LED7 выключен, LED6 – включен), значение 0xC1(11000001) – погашены оба светодиода, 0x41(01000001) – LED6 погашен, LED7 включен. Список сообщений:

AVR Simulator PORTB - 000003189:81
AVR Simulator PORTB - 000003276:C1
AVR Simulator PORTB - 000003278:41
AVR Simulator PORTB - 000506301:C1
AVR Simulator PORTB - 001009328:41
AVR Simulator PORTB - 001512351:C1
AVR Simulator PORTB - 002015377:81

Длительность включения/ светодиода LED7 составляет 0,503с.

11. Изменим значение задержки в процедуре *delay_ms()*. Редактировать объектные файлы в AVR Studio 4 нельзя, поэтому снова открываем CodeVisionAVR, исправляем соответствующие строки и заново выполняем компиляцию. Если AVR Studio не был закрыт, то появится сообщение, что объектный файл был изменён, и его нужно перезагрузить, что и делаем.

12. Выполним компиляцию в CodeVisionAVR, установив в окне *Configure* тип выходного файла Intel HEX. Загрузив полученный файл с расширением .hex в микроконтроллер STK500 и изменив на вкладке Board окна STK500 частоту, проверяем работу программы. В исходном состоянии светодиод LED6 включен, LED7 – выключен. Нажимаем несколько раз кнопку SW0. Нажав затем кнопку SW2, наблюдаем мигание светодиода LED7 с частотой 1 Гц. В это время светодиод LED6 выключен. После завершения мигания LED6 вновь включается, а устройство переходит в режим ожидания.