Лабораторная работа 7. Программирование и отладка программ на языке Си для микроконтроллеров AVR

Цель работы:

- изучение типовых инструкций Си для настройки ресурсов микроконтроллеров AVR,
- знакомство с встроенным в AVR Studio 4 компилятором AVR GCC,
- отладка, модификация и прогон тестовых программ,

Введение

Большую популярность при программировании микроконтроллеров имеет язык Си. При его использовании сокращается время на разработку программы, что особенно заметно при написании больших программ, обеспечивается её 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: AVR GCC, CodeVisionAVR, ImageCraft C 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 и отладки использованы программы AVR Studio 4 версии 4.19 с встроенным компилятором AVR GCC и ISIS Proteus.

Создание проекта

Работа над проектом в AVR Studio 4 начинается с выбора типа проекта AVR GCC, указания папки для хранения проекта, имени проекта и имени файла. Затем выбирается тип микроконтроллера.

Для задания параметров проекта в целом необходимо выбрать команду меню *Project /Configuration Options*. На вкладке *General* устанавливаем тип МК ATmega 8515, частоту 4000000 Гц, оптимизацию -Os, формат .elf выходного файла, предусмотрев создание файлов с расширением .hex, .map, .lsl, после чего нажимаем ОК. В окно программы вводим текст

программы и сохраняем файл. Рассмотрим ряд примеров по созданию и работе над проектом с использованием языка Си.

Взаимодействие микроконтроллера со светодиодами

1. Запустив AVR Studio 4, создать в рабочей папке проект. В окно программы ввести программу на языке Си для последовательного переключения светодиодов STK500.

Программа использует таймер T1, при переполнении которого выполняется перезагрузка счетчика TCNT1, сдвиг влево значения статусной переменной светодиодов led_status, присоединение 1 в младшем разряде и вывод в порт PC.

Перезагружаемое значение TCNT1 рассчитывается, исходя из следующих представлений (рис. 1).

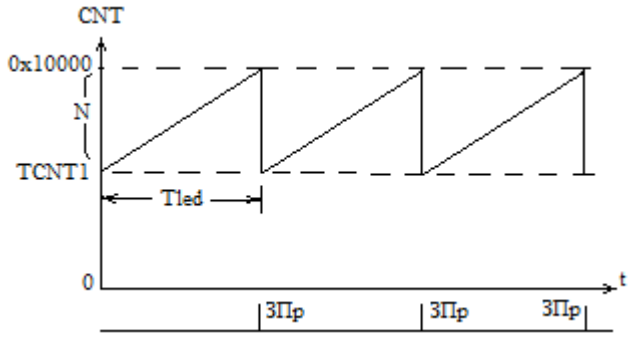

Рис.1. Диаграмма работы счетчика

16-разрядный счетчик, работая в суммирующем режиме, ведет отсчет от начального значения TCNT1 до значения переполнения, которое наступает при достижении значения 0x10000=65536 и сопровождается формированием запроса прерывания (Зпр) от таймера для выполнения функции прерывания, которая изменяет состояние светодиодов (переменной led_status), подключенных к выводам порта PC.

Число состояний, отсчитываемое таймером с момента загрузки до переполнения, составляет

$$N = T_{LED} / T_{CNT},$$

где T_{LED} – период обновления состояния светодиодов, T_{CNT} – период следования счетных импульсов.

Используя соотношение между частотой и периодом f = 1/ T, можно записать

$$N = f_{CNT} / f_{LED},$$

где f_{CNT} – следования счетных импульсов, f_{LED} – частота обновления состояния светодиодов.

Обозначив частоту работы микроконтроллера Xtal и коэффициент предделителя таймера K, получим расчётное выражение:

$$TCNT1 = 65536 – N = 0x10000 – Xtal / K/ f_{LED}$$

Для работы программы выбираем Xtal = 3686400 Гц, K=1024, T_{LED} = ½ с и следовательно fled = 2 Гц.

Программа 1.

```
/*Программа 7.1 управляет переключением светодиодов, подключенных к выходам порта PC.*/
#include <avr/interrupt.h>
#include <avr/io.h>

#define xtal 3686400
#define fled 2
unsigned char led_status=0xfe;

ISR(TIMER1_OVF_vect)
{
      TCNT1=0x10000-(xtal/1024/fled);
      led_status<<=1;
      led_status|=0x1;
      if (led_status==0xff) led_status=0xfe;
      PORTC=led_status;
}
 int main(void)
 {
      DDRC=0xff;
      PORTC=led_status;
      TCCR1A=0;
      TCCR1B=5;
      TCNT1=0x10000-(xtal/1024/fled);
      TIFR=0;
      TIMSK=0x80;
      GICR=0;
      sei();
      while (1);
}
```

Компиляция

1. Для компиляции программы необходимо выбрать команду меню *Build/ Build (F7)*. Результаты компиляции выводятся в окно *Build*. Сообщения компилятора также выводятся в окно *Message* внизу экрана.

После компиляции будут созданы файлы с расширениями .elf, .map, .hex.

В случае успешной компиляции выводится сообщение об отсутствии ошибок.

Выполнив компиляцию программы, загрузить выходной файл с расширением .hex в микроконтроллер STK500 и наблюдать работу программы. Изменить программу, уменьшив (увеличив) скорость переключения светодиодов в 2 раза по сравнению с первоначальной.

2. Проверить работу программы в симуляторе программы Proteus. Для этого, запустив программу ISIS Proteus, в окне редактора нарисовать схему устройства, содержащую используемый в программе микроконтроллер и 8 светодиодов, подключив их к выходам порта PC. Сохранить проект в той же папке, где сохранен проект AVR Studio 4. Выделив на схеме правой кнопкой мыши микроконтроллер и открыв щелчком левой кнопки окно с его свойствами, “прикрепить” к микроконтроллеру откомпилированный файл с расширением .elf из папки …\default. Запустив проект, в окне исходного файла, открываемом в меню отладки, наблюдать пошаговое исполнение инструкций программы. В автоматическом режиме исполнения наблюдать работу светодиодов.

3. Изменить последовательность переключения светодиодов, от старшего разряда к младшему с временем включения каждого светодиода, равным 2 с. Проверить работу программы по часам симулятора.

Обработка внешнего прерывания

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: LED6 для информирования о готовности схемы и LED7 для мигания. Таким образом, микроконтроллер должен иметь возможность обработки внешнего прерывания от кнопки SW2, таймер для подсчета входных сигналов от кнопки SW0 и две линии порта для управления светодиодами. Выберем микроконтроллер ATx8515, частоту 4 МГц. “Подтягивающие” резисторы на входах PD2, PB0 подключаются при программировании режима работы портов.

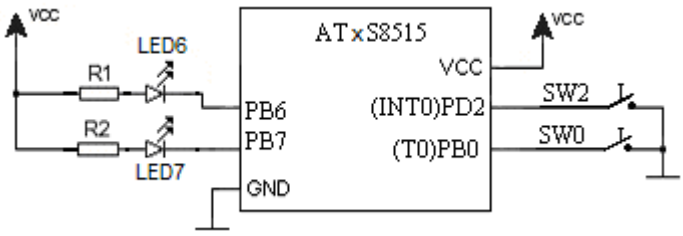

Рис.2. Схема устройства

Вводим программу, текст которой представлен ниже. (Примечание: для микроконтроллера AT90S8515 вместо имени регистра маски внешних прерываний GICR используем имя GIMSK).

Программа 2.

```
/*Программа 7.1 с помощью таймера подсчитывает число нажатий
на кнопку SW0(PINB0).Затем после нажатия на кнопку SW2(PIND2) переключает
светодиод LED7 по значению таймера.
В режиме ожидания включен светодиод LED6.*/

#include <avr/interrupt.h>
#include <avr/io.h>
#include <util/delay.h>

// Обработки внешнего прерывания INT0
ISR(INT0_vect)
{ char timer;             // локальная переменная
  timer = TCNT0;
  if (timer != 0)
   {TCNT0 = 0;            // сброс таймера/счётчика
    PORTB |= (1<<PB6);    //PORTB=0b11000001 (выключаем светодиод LED6)
    do {
      PORTB &= ~(1<<PB7);//PORTB=0b01000001 (включаем светодиод LED7)
      _delay_ms(500);     // задержка 500 мс
      PORTB |= (1<<PB7); //PORTB=0b11000001 (выключаем светодиод LED7)
      _delay_ms(500);
         } while (--timer != 0);
    PORTB &= ~(1<<PB6);  //PORTB=0b10000001 (включаем светодиод LED6)
   }
}
int main(void)
{
  // Инициализация портов
```

```
  DDRB=0xC0;            // PB7,PB6 для вывода на LED7,LED6 PB0- для ввода
  PORTB=0b10000001;     // выключаем LED7, PB0-подтягивающий резистор кнопки
  DDRD=0;
  PORTD=(1<<PD2);       // PD2-подтягивающий резистор
  // Инициализация таймера 0
  TCCR0=0x06;
  TCNT0=0x00;

  GICR=(1<<INT0);       // инициализация прерывания INT0 в GICR (или GIMSK)
  MCUCR=(1<<SE);       // разрешение перехода в режим Idle
  sei();               // глобальное разрешение прерываний
  for (;;) {
  asm("sleep");        // переход в режим Idle
  asm("nop");
              }
}
```

Отладка в AVR Studio

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

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

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 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.

1. Отладка начинается по команде *Debug/ Start Debugging* из меню AVR Studio 4. В начале процесса отладки и в случае нажатия кнопки сброса *Reset* курсор отладки устанавливается на первой строке процедуры *main.*

2. Устанавливаем параметры отладки, выполнив команду *Debug /AVR Simulator Options*: ATx8515, частота 4 МГц, протоколирование порта PORTB (Logging), с выводом на экран (To screen, Add Entry, OK).

3. 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.

4. Занесём в счётчик TCNT0 значение 2. Затем при разомкнутой кнопке SW0 (состояние «1») эмулируем замыкание кнопки SW2 (состояние «0»), что приводит к вызову обработчика прерывания. Установив контрольную точку в цикле for(;;) продолжим выполнение программы по команде Run (F5).

6. После останова симуляции, что можно обнаружить по желтому индикатору в строке состояния, в окне *Message* получим список сообщений (отчет) в формате Cycle:Data (число выполненных циклов на момент обновления содержимого порта : шестнадцатеричное значение порта), в котором значение 0x81(10000001) соответствует готовности схемы (светодиод LED7 выключен, LED6 – включен), значение 0xC1(11000001) – выключению обоих светодиодов, 0x41(01000001) – включению LED7 и выключению LED6.

Список сообщений:

AVR Simulator: PORTB - 000000000:00

AVR Simulator: PORTB - 000000015:81

AVR Simulator: PORTB - 000000061:C1

AVR Simulator: PORTB - 000000063:41

AVR Simulator: PORTB - 002000065:C1

AVR Simulator: PORTB - 004000070:41

AVR Simulator: PORTB - 006000072:C1

AVR Simulator: PORTB - 008000076:81

Длительность включения/выключения светодиода LED7, определяемая по разности значений числа циклов соседних строк, составляет 2000002 цикла, что при частоте 4 МГц составляет 0,5с.

7. Изменить значение задержки в процедуре *_delay_ms()*. Повторно выполнить компиляцию и прогон программы в режиме отладки. Оценить результат работы программы по протоколу.

8. Загрузив полученный файл с расширением .hex в микроконтроллер STK500, проверить работу программы. В исходном состоянии светодиод LED6 включен, LED7 – выключен. Нажмите несколько раз кнопку SW0. Нажав затем кнопку SW2, наблюдайте мигание светодиода LED7. В это время светодиод LED6 выключен. После завершения мигания LED6 вновь включается, а устройство переходит в режим ожидания.

Взаимодействие микроконтроллера AVR со схемой параллельного интерфейса 8255A

1. Запустить программу ISIS из пакета Proteus. Открыть графический шаблон проекта *8255* из папки …\AVR with 8255A IO expander.

2. Создать проект в AVR Studio 4 на платформе ATmega16.

В окно программы загрузить подготовленную программу, текст которой приведен ниже, и выполнить компиляцию для получения выходного файла с расширением .elf.

Программа осуществляет настройку микросхемы параллельного интерфейса 8255A для ввода данных с кнопок через порт B и вывода данных на светодиоды через порт A путем загрузки управляющего слова 0x82 в регистр управления 8255A. Затем выполняется цикл, в котором происходит непрерывный опрос кнопок и отображение их состояния на светодиодах.

Программа 3.

```
#include <avr/io.h>
#define uchar unsigned char
// Определения уровней сигналов бита (x) порта
#define sbit(x,PORT) ((PORT) |= (1<<x))
#define cbit(x,PORT) ((PORT) &= ~(1<<x))
// определения интерфейсных сигналов
// RST,LE,CS,RD,WR
#define srst sbit(0,PORTD)
#define crst cbit(0,PORTD)
#define sle  sbit(1,PORTD)
#define cle  cbit(1,PORTD)
#define scs  sbit(2,PORTD)
#define ccs  cbit(2,PORTD)
#define srd  sbit(3,PORTD)
#define crd  cbit(3,PORTD)
#define swr  sbit(4,PORTD)
#define cwr  cbit(4,PORTD)
// Вывод адресов/данных, ввод данных
#define  out  PORTC
#define  in   PINC

// 'Защёлкивание' адреса в регистре
void latch_it(void)
 { cle;
   asm("nop");
   sle;
   asm("nop");
   cle;
 }
```

```
int main()
 { uchar temp;
   SPL = 0x54;
   SPH = 0x04;
   // Инициализация порта PD
   DDRD = 0xff;
   // Неактивные входы 8255A
   srd;
   swr;
   scs;
   // Разрешение 8255A и сброс
   ccs;
   srst;
   asm("nop"); asm("nop"); asm("nop");
   crst;
   //
   DDRC=0xff;
   out = 0x03;  //адрес регистра управления 8255A
   latch_it();
   out = 0x82;
   cwr;
   asm("nop");
   swr;

   while(1)
    { //
      DDRC=0xff;
      out = 0x01; //адрес на вывод
      latch_it();
      DDRC = 0;   //KEY на ввод
      crd;
      asm("nop");
      temp = in; //данные KEY
      srd;
      //
      DDRC = 0xff;
      out = 0x00; //адрес на вывод
```

```
        latch_it();
        out = temp; //данные на LED
        cwr;
        asm("nop");
        swr;
      }
  }
```

3. Изучить текст программы и определить назначение каждой инструкции. В режиме отладки провести пошаговое выполнение программы (клавиша *F11*), контролируя выполнение каждой инструкции программы в регистрах портов PD и PC, переменной *temp*.

Повторить работу с пошаговым выполнением программы, контролируя выполнение команд в окне дизассемблера.

4. Перейти в окно программы ISIS. Полученный после компиляции компилятором AVR-GCC файл с расширением .elf из папки …*default* присоединить к микроконтроллеру в окне его свойств, которое открывается при выделении микроконтроллера правой кнопкой мыши и последующего щелчка левой.

Запустить режим выполнения программы *Debug/Execute*. Проверить работу программы, поочередно нажимая кнопки SW.

5. Построить временную диаграмму протокола взаимодействия микроконтроллера, регистра адреса и схемы параллельного интерфейса 8255A, включив в нее диаграммы управляющих сигналов LE, RD, WR и шинных данных D[0..7], KEY[0..7] и LED[0..7]. Для отображения состояния шины на осциллограмме необходимо каждой контролируемой шине присвоить метку (*Label*), выделив шину и выбрав в контекстно-зависимом меню опцию *Place Wire Label*. Для построения осциллограмм воспользоваться инструментом *Graph Mode/Digital*, установив на контролируемые линии и шины пробники и выполнив необходимые настройки из контекстно-зависимого меню окна моделирования (добавив трассы и задав время моделирования).

Оформление отчета

Отчет должен содержать:

а) схемы проектов и тексты программ с комментариями,

б) выводы и заключения по работе программ в симуляторе и на плате,

в) протокол взаимодействия микроконтроллера с устройством 8255A.

Требования при защите: уметь отвечать на вопросы по текстам программ на языке Си.

Контрольные вопросы

1. Предложите альтернативные операторы (группы операторов), которые могут быть применимы для инициализации портовых регистров, отдельных битов регистров и линий интерфейса.

2. Назовите последовательность действий для выполнения компиляции проекта и для отладки с использованием контрольных точек.

3. Назовите последовательность действий для запуска проекта на языке Си в среде симуляции Proteus ISIS.

4. Рассчитайте значения TCNT1 для программы 2, при которых время включения светодиода составит 1/4 с или 4 с.

Рекомендуемая литература

1. Шпак Ю.А. Программирование на языке С для AVR и PIC микроконтроллеров. 2-е издание. Корона-Век, МК-Пресс 2011 г.

2. AVR033: Getting Started with the CodeVisionAVR C Compiler. Rev. 2500C–AVR–04/08