Министерство образования и науки
Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра конструирования узлов и деталей радиоэлектронной аппаратуры
(КУДР)

А.А. Бомбизов, А.Г. Лощилов

АНАЛОГО-ЦИФРОВОЙ ПРЕОБРАЗОВАТЕЛЬ

Методические указания к выполнению
лабораторной и самостоятельной работы
по дисциплине «Программирование микроконтроллеров»

Томск 2017

1 Введение

В настоящее время в проектировании всего разнообразия радиоэлектронных устройств уделяется большое внимание оценке некоторого внешнего воздействия. Для этого требуется преобразование входной физической величины в ее числовое представление.

Целью настоящей работы является освоение принципов взаимодействия с аналого-цифровым преобразователем, размещенном в микроконтроллерном устройстве.

2 Краткая теория

Аналого-цифровой преобразователь (АЦП, англ. Analog-to-digitalconverter, ADC) – устройство, преобразующее входной аналоговый сигнал в дискретный код (цифровой сигнал).

Понятие аналого-цифрового преобразования тесно связано с понятием измерения. Под измерением понимается процесс сравнения измеряемой величины с некоторым эталоном, при аналого-цифровом преобразовании происходит сравнение входной величины с некоторой опорной величиной (как правило, с опорным напряжением). Таким образом, аналого-цифровое преобразование может рассматриваться как измерение значения входного сигнала, и к нему применимы все понятия метрологии, такие как погрешности измерения.

АЦП имеет множество характеристик, из которых основными можно назвать **частоту преобразования** и **разрядность**. Частота преобразования обычно выражается в отсчетах в секунду (samples per second, SPS) (часто употребляется частота дискретизации, выражаемая в Гц), разрядность – в битах. Например, АЦП с разрядностью 8 бит тарирует входной диапазон напряжений в 256 градациях (от 0 до 255). Современные АЦП могут иметь разрядность до 24 бит и скорость преобразования до единиц GSPS (из-за сложности архитектуры добиться высоких значений одновременно обоих параметров очень сложно). Чем выше скорость и разрядность, тем труднее получить требуемые характеристики, тем дороже и сложнее преобразователь. Скорость преобразования и разрядность связаны друг с другом определенным образом, поэтому можно повысить эффективную разрядность преобразования, пожертвовав скоростью.

Существует множество типов АЦП, но основных можно выделить следующие:

- АЦП параллельного преобразования (прямого преобразования, flash ADC);
- АЦП последовательного приближения (SAR ADC);

- дельта-сигма АЦП (АЦП с балансировкой заряда).

Наибольшим быстродействием и самой низкой разрядностью обладают АЦП прямого (параллельного) преобразования. АЦП данного типа могут иметь скорость преобразования до 1 GSPS. Среднюю нишу в ряду разрядность/скорость занимают АЦП последовательного приближения. Типичными значениями является разрядность 12–18 бит при частоте преобразования 100 kSPS – 1 MSPS. Наибольшую точность достигают дельта-сигма АЦП, имеющие разрядность до 24 бит включительно и скорость от единиц SPS до единиц kSPS.

АЦП в микроконтроллере AVR, в том числе и в ATMega328, представляет собой преобразователь последовательного приближения с устройством выборки-хранения и фиксированным числом тактов преобразования, равным 13. Структурная схема АЦП изображена на рисунке 1.

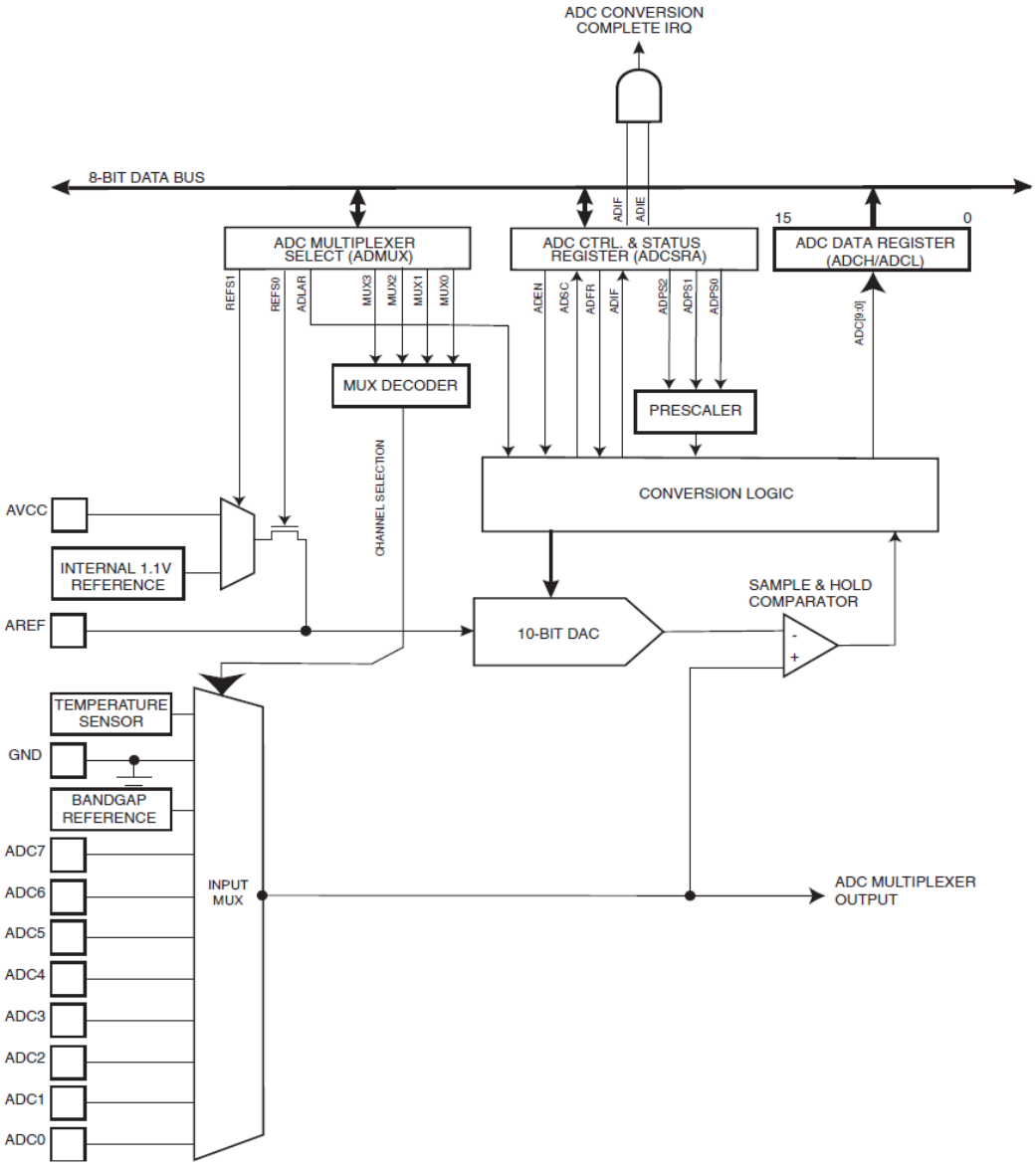

Рисунок 1– Структурная схема АЦП в ATMega328

Микроконтроллер ATMega328 содержит 8 мультиплексированных измерительных входа (рисунок 2). Это означает, что на входе единственного модуля АЦП установлен аналоговый мультиплексор, который может подключать этот вход к различным выводам микроконтроллера для осуществления измерений нескольких независимых аналоговых величин с разнесением по времени. Входы мультиплексора могут работать по отдельности (в несимметричном режиме для измерения напряжения относительно "земли") или (в некоторых моделях) объединяться в пары для измерения дифференциальных сигналов.

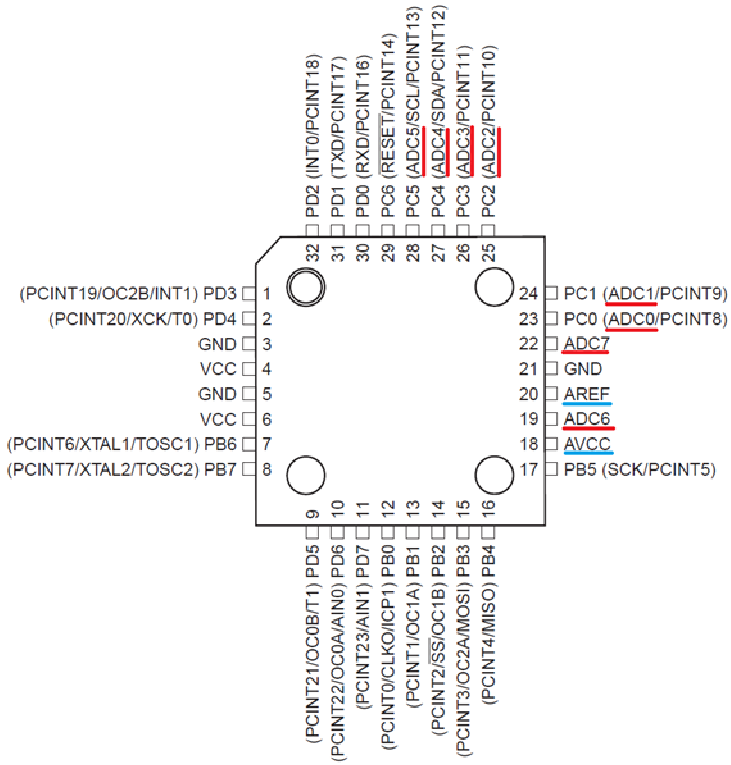

Рисунок 2 – Изображение микроконтроллера с выводами АЦП

Тактовая аналого-цифрового преобразователя частота формируется аналогично тому, как это делается для таймеров – с помощью специального предделителя тактовой частоты микроконтроллера, который может принимать коэффициенты деления от 1 до 128. Но в отличие от таймеров, выбор тактовой частоты АЦП не совсем произволен, т.к. быстродействие аналоговых компонентов ограничено. Поэтому коэффициент деления следует выбирать таким, чтобы при установленном кварце тактовая частота АЦП укладывалась в рекомендованный диапазон 50–200 кГц (т.е. максимум около 15 тыс. измерений в секунду с учетом 13 тактов на одно преобразование).

Разрешающая способность АЦП – 10 двоичных разрядов, чего для большинства типовых применений достаточно. Абсолютная погрешность преобразования зависит от ряда факторов и в идеальном случае не

превышает ±2 младших разрядов, что соответствует общей точности измерения примерно 8 двоичных разрядов.

Регистры управления АЦП

Микроконтроллер ATMega328 содержит следующие регистры [1]:

- **ADMUX** – регистр выбора входа мультиплексора и источника опорного напряжения (ИОН);
- **ADCSRA**и **ADCSRB** – регистры управления АЦП;
- **ADCL** и **ADCH** – регистры, в который АЦП помещает результат преобразования. В программе используется **ADC**;
- **DIDR0** – регистр управления входами мультиплексора.

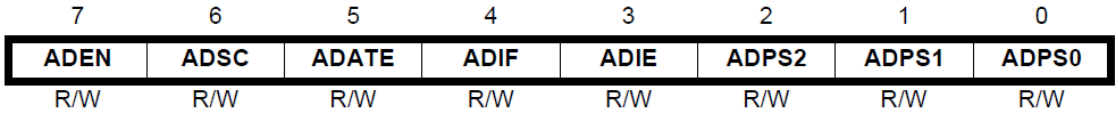

Рисунок 3 – Регистр ADCSRA

Для разрешения работы АЦП необходимо записать лог. 1 в разряд **ADEN** (7 бит) регистра **ADCSRA**, а для выключения – лог. 0. Если АЦП будет выключено во время цикла преобразования, то преобразование завершено не будет (в регистре данных АЦП останется результат предыдущего преобразования).

Установка бита **ADATE**(5) указывает на режим запуска АЦП от внешнего источника (таблица 2).

Если выбран режим запуска не от внешнего источника, то преобразование является однократным и запускается установкой бита **ADSC**(6). При непрерывном режиме установка этого бита запустит первое преобразование, затем они будут автоматически повторяться. В режиме однократного преобразования, а также независимо от установленного режима при запуске через прерывания установка бита **ADSC** только запускает одно преобразование. При наступлении прерывания, запускающего преобразование, бит **ADSC** устанавливается аппаратно. Преобразование начинается по фронту первого тактового импульса АЦП после установки **ADSC**. По окончании любого преобразования (как в одиночном, так и в непрерывном режиме) устанавливается бит **ADIF** (4) – флаг прерывания. Разрешение прерывания (**ADC_vect** – вектор прерывания от аналого-цифрового преобразователя) АЦП осуществляется установкой бита **ADIE** (бит 3) регистра **ADCSRA**. В данном случае прерывание от АЦП будет вызываться каждый раз при завершении преобразования и в обработчике прерывания, как правило, происходит чтение оцифрованного значения из АЦП в пользовательскую переменную.

Биты **ADPS2**–**ADPS0**(2–0) регистра ADCSRA служат для выбора режима работы предделителя тактовой частоты. То есть если при частоте

процессора равной 8 МГц установить коэффициент деления 64, то частота тактирования АЦП будет 125 кГц. Для этого достаточно установить биты ADPS2 и ADPS1в логические единицы (таблица 2).

Таблица 1 –Варианты конфигурации предделителя

ADPS2	ADPS1	ADPS0	Коэффициент деления
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

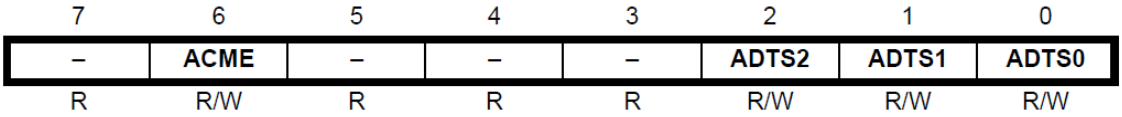

Рисунок4– Регистр ADCSRB

Бит **ACME**(6) регистра ADCSRB позволяет использовать мультиплексор АЦП в качестве входов аналогового компаратора при установке 1 (при этом АЦП должен быть выключен).

Биты **ADTS2–ADTS0**(2–0) регистра ADCSRB служат для выбора источника сигнала, по которому будет начинаться преобразование АЦП (таблица 2).

Таблица 2 – Источник сигнала запуска

ADTS2	ADTS1	ADTS0	Описание
0	0	0	непрерывное преобразование
0	0	1	прерывание от аналогового компаратора
0	1	0	внешнее прерывание INT0
0	1	1	прерывание по совпадению таймера/счетчика T0 с A
1	0	0	прерывание по переполнению таймера/счетчика T0
1	0	1	прерывание по совпадению таймера/счетчика T1 с B
1	1	0	прерывание по переполнению таймера/счетчика T1
1	1	1	прерывание по захвату таймера/счетчика T1

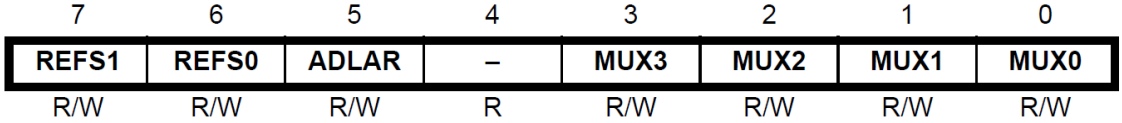

Рисунок5– Регистр ADMUX

Выбор источника опорного напряжения выполняется установкой битов **REFS1**(7 бит)**–REFS0**(6 бит) регистра **ADMUX**. Нулевые значения конфигурационных битов соответствуют внешнему источнику **AREF**

(см. рисунок 2), который может принимать значение в пределах от 2 В до напряжения питания аналоговой части микроконтроллера **AVcc** (не должно отличаться от питания цифровой части более чем на 0,3 В в большую или меньшую сторону). В качестве опорного напряжения можно выбрать питание самой аналоговой части одним из двух способов: либо просто соединить выводы **AREF** и **AVcc** микросхемы, либо установить биты **REF S1..0** в состояние 01 (тогда соединение осуществляется внутренними схемами). Предусмотрен и встроенный источник (биты **REF S1..0** устанавливаются в состояние 11, при этом к выводу **AREF** рекомендуется подключать фильтрующий конденсатор), имеющий номинальное напряжение 1,1 В (для микроконтроллера ATMega328). Все варианты конфигурации сведены в таблицу 3.

Таблица 3 – Конфигурация источника опорного напряжения

REF 1	**REF 0**	**Источник опорного напряжения**
0	0	Внешний ИОН, подключенный к выводу **AREF** , внутренний ИОН отключен
0	1	Напряжение питания **AVcc***
1	0	Зарезервировано, в работе не используется
1	1	Внутренний ИОН напряжением 1,1 В, подключен к выводу **AREF** *
*Если к выводу **AREF** подключен источник напряжения, данные варианты использоваться не могут		

Выбор каналов и режимов их взаимодействия в АЦП производится битами **MUX0..3** в регистре **ADMUX**. Варианты конфигурации битов для выбора необходимого входа отображены в таблице 4. Входы **ADC0**–**ADC7** изображены на рисунке 2.

После завершения преобразования (при установке в «1» флага **ADIF** регистра **ADCSR**) его результат сохраняется в регистре данных **АЦП**. Поскольку АЦП имеет 10 разрядов, этот регистр физически размещен в двух регистрах ввода/вывода **ADCH:ADCL**, доступных только для чтения. По умолчанию результат преобразования выравнивается вправо (старшие 6 разрядов регистра **ADCH** – незначащие). Однако он может выравниваться и влево (младшие 6 разрядов регистра **ADCL** – незначащие). Для управления выравниванием результата преобразования служит разряд **ADLAR** регистра **ADMUX**. Если этот разряд установлен в «1», то результат преобразования выравнивается по левой границе 16-разрядного слова, если сброшен в «0» – по правой границе.

Таблица 4 – Выбор входа

MUX3...0	**Выбранный вход**
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8 (Встроенный температурный датчик)
1110	Встроенный ИОН со значением 1,1 В
1111	0 В (общий)

Во многих средах программирования, в том числе и ArduinoIDE, результат, находящийся в регистрах **ADCH** и **ADCL**, адаптирован для чтения из одной внутренней переменной **ADC**.

Результат преобразования определяется формулой:

$$ADC = \frac{V_{IN} \cdot 1023}{V_{REF}}, \quad (1)$$

где V_{IN} – измеряемое напряжение на входе АЦП;

V_{REF} – выбранное опорное напряжение.

Обратный пересчет из значения в отсчетах в физическую величину выполняется по следующей формуле:

$$V_{IN} = \frac{V_{REF} \cdot ADC}{1023}. \quad (2)$$

3 Порядок выполнения работы

В ходе данной работыдолжны быть освоены основы управления аналого-цифровым преобразователем на примере измерителя напряжения, снимаемого с переменного резистора.

Для этого необходимо выполнить следующие действия:

3.1 Изучите предложенный в п. 2 теоретический материал.

3.2 Создайте новый проект.

3.3 Скопируйте исходные код, содержащийся в приложении А, и вставьте в новый проект.

3.4 Откомпилируйте проект и запрограммируйте микроконтроллер. На семисегментных индикаторах должно отобразиться число «100.0». Проверьте работоспособность алгоритма с выводом других чисел с плавающей точкой при помощи функции *SHRSendNum*.

3.5 Установите источник опорного напряжения **AVcc**.

3.6 Определите по [2] и [3] к какому выводу микроконтроллера и входу АЦП подключен переменный резистор.

3.7 Переключите внутренний мультиплексор АЦП на вывод, к которому подключен переменный резистор.

3.8 Переключите АЦП в режим запуска от внешнего источника.

3.9 Разрешите прерывание от АЦП.

3.10 Установите частоту тактирования АЦП в диапазоне 50–200 кГц. Тактовую частоту процессора принять 16 МГц.

3.11 Разрешите работу АЦП.

3.12 Переключите источник сигнала запуска в режим непрерывного измерения.

3.13 Разрешите прерывания.

3.14 Запустите первое преобразование.

3.15 Добавьте в программу функцию обработки прерывания от АЦП.

3.16 Создайте глобальную переменную типа *float* с именем *valADC*.

3.17 В функции обработки прерывания от АЦП сохраните значение из регистра ADC в переменную *valADC*.

3.18 В функции *loop()* выведете значение переменной *valADC* на семисегментные индикаторы.

3.19 Откомпилируйте и запустите программу. В результате на семисегментных индикаторах должно отображаться напряжение на переменном резисторе, выраженное в отсчетах АЦП.

3.20 Поверните настроечную ручку переменного резистора и проследите за изменением значения на семисегментном индикаторе.

3.21 Пересчитайте значение в переменной *valADC* таким образом, чтобы оно содержало значение напряжения на переменном резисторе. Опорное напряжение принять равным 5 В.

3.22 Откомпилируйте и запустите программу.

3.23 Оформите отчет, содержащий титульный лист и разделы: введение, ход выполнения работы, ответы на контрольные вопросы и выводы.

3.24 Защитите отчет у преподавателя.

4 Контрольные вопросы

4.1 Сколько тактов требуется АЦП для оцифровки физической величины?

4.2 Возможно ли одновременное измерение напряжения с двух входов?

4.3 Какую частоту оцифровки и почему рекомендуется устанавливать в ATMega328?

4.4 Помимо срабатывания прерывания как определить факт окончания оцифровки физической величины?

4.5 Какое максимальное значение может принимать регистр ADC (**ADCH**:**ADCL**)?

4.6 Каким образом перевести значение возвращаемое аналого-цифровым преобразователем в истинное значение?

Список литературы

1. ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES, IN-SYSTEM PROGRAMMABLE FLASH.– URL: http:\\www.atmel.com\images\Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf (Дата обращения: 10.01.2017).

2. Multi-shield schematic.– URL:http://makbit.com/web/wp-content/uploads/2015/12/Multi-A1010.zip (Дата обращения: 01.03.2017).

3. UNO Schematic – Arduino.– URL: https:\\www.arduino.cc\en\uploads\Main\arduino-uno-schematic.pdf (Дата обращения: 10.01.2017).

Приложение А

```
#defineSHR_SDIPB0
#define SHR_CLK PD7
#define SHR_LCHCLK PD4
#define SHR_DDRDATA DDRB
#define SHR_DDRCLK DDRD
#define SHR_PORTDATA PORTB
#define SHR_PORTCLK PORTD
#define SHR_DELAY 0

void SHRSetup()
{
 SHR_DDRDATA|=1<<SHR_SDI;
 SHR_DDRCLK|=1<<SHR_CLK;
 SHR_DDRCLK|=1<<SHR_LCHCLK;
 SHR_PORTDATA&=~(1<<SHR_SDI);
 SHR_PORTCLK&=~(1<<SHR_CLK);
 SHR_PORTCLK&=~(1<<SHR_LCHCLK);
}
void SHRSendBit(int bit)
{
 SHR_PORTDATA|=bit<<SHR_SDI;
 SHR_PORTCLK|=1<<SHR_CLK;
 SHR_PORTDATA&=~(bit<<SHR_SDI);
 SHR_PORTCLK&=~(1<<SHR_CLK);
}
```

```
void SHRSendByte(byte data)
{
  for(int i=0;i<8;++i)
  SHRSendBit((data&(1<<(7-i)))>>(7-i));
}

void SHRSendLatch()
{
  SHR_PORTCLK|=1<<SHR_LCHCLK;
  SHR_PORTCLK&=~(1<<SHR_LCHCLK);
}

void SHRSendSymb(int num,bool isDot)
{
  static char table[10]={127&~(1<<6),//0
            (1<<2)|(1<<1),//1
            (1<<6)|(1<<4)|(1<<3)|(1<<1)|(1<<0),//2
            (1<<6)|(1<<3)|(1<<2)|(1<<1)|(1<<0),//3
            (1<<6)|(1<<5)|(1<<2)|(1<<1),//4
            (1<<6)|(1<<5)|(1<<3)|(1<<2)|(1<<0),//5
            (1<<6)|(1<<5)|(1<<4)|(1<<3)|(1<<2)|(1<<0),//6
            (1<<2)|(1<<1)|(1<<0),//7
            127,
            127&~(1<<4)
            };
  int dot=(isDot<<7);
  SHRSendByte(~(table[num]|dot));
}

void SHRSelectSegment(int num)
{
  SHRSendByte(1<<num);
}

void SHRSendNum(float num)
{
  floattmp=1000;
  intnumAfterPoint=0;
  while((tmp>num)&&(tmp>1))
  {
    tmp/=10;
    numAfterPoint++;
  }
  intnumBeforePoint=4-numAfterPoint;
  for(int i=0;i<numAfterPoint;++i)
    num*=10;
  int intNum=num;

  int num4=intNum%10;
  intNum/=10;
  int num3=intNum%10;
  intNum/=10;
  int num2=intNum%10;
  intNum/=10;
  int num1=intNum%10;
```

```
  SHRSendSymb(num1,(numBeforePoint==1));
  SHRSelectSegment(0);
  SHRSendLatch();
  SHRSendSymb(num2,(numBeforePoint==2));
  SHRSelectSegment(1);
  SHRSendLatch();
  SHRSendSymb(num3,(numBeforePoint==3));
  SHRSelectSegment(2);
  SHRSendLatch();
  SHRSendSymb(num4,false);
  SHRSelectSegment(3);
  SHRSendLatch();
}

void setup()
{
  SHRSetup();

}

void loop()
{
  SHRSendNum(100);
}
```