```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.03.9 Evaluation
Automatic Program Generator
© Copyright 1998-2008 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 4/6/2009
Author  : Freeware, for evaluation and non-commercial use only
Company :
Comments:  program to read and write internal eeprom memory.


Chip type               : ATmega32
Program type            : Application
AVR Core Clock frequency: 4.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 512
*****************************************************/

#include <mega32.h>

// Standard Input/Output functions
#include <stdio.h>
#include <delay.h>
// Declare your global variables here

eeprom char str[3] = {48,49,50};
eeprom int a=48;
int b=50;

int write_eeprom_byte(char address, char data)
{
//    EEARH=0x00;
//    EEARL = address ;

    while(EECR & 0x02 == 1) ;          // CHECK EEWE BIT

    while( SPMCR & 0x01 == 1);         // CHECK SPMEN BIT

    EEARH=0x00;                    // SET ADDRESS MSB BYTE
    EEARL = address ;              // SET LSB BYTE OF ADDRESS
    EEDR = data ;                 // WRITE data to be written to EEPROM
    EECR = 0x04 ;                  // SET EEMWE BIT
    #asm
        nop;
        nop;
```

```
   #endasm
   EECR = 0x06;                    // SET EEMWE & EEWE BITS

   while(EECR & 0x04 == 1);        // CHECK EEMWE IS CLRED


  //  delay_ms(10);                // general delay

   return(1);


}

//////////////////////////////////////////////////////////////////////

char eeprom_read_byte(char address)
{
   char data=0;

   while(EECR & 0x02 == 1) ;       // check EEWE BIT IS CLRed

   EEARH=0x00;                     // SET MSB BYTE OF ADDRESS
   EEARL = address ;               // SET LSB BYTE OF ADDRESS

   EECR = 0x01 ;                   // SET EERE BIT

   #asm
      nop;
      nop;

   #endasm

   data = EEDR;                    // read data from the EEPROM

   return(data);                   // RETURN EEPROM data value


}

//////////////////////////////////////////////////////////////////////


void main(void)
{
// Declare your local variables here
   int i=0;
   int ret=1;                      // to store function return value
   char read_byte=0;               // to store eeprom data value
   bit eeprom_write_enable=1;      // to enable write operation on eeprom
```

```
// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
```

```
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: Off
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x08;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x19;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

  // if 1, perform write opertion on internal EEPROM
```

```
if( eeprom_write_enable == 0)
{

   ret=write_eeprom_byte(0x10,0x30);      // write value to specified eeprom address
   if(ret == 1)                           // check write successful completion
      putchar(65) ;                       // display char A on hyperterminal

   ret=0;                                 // clr before reading again

   ret=write_eeprom_byte(0x20,0x31);      // write value to specified eeprom address
   if(ret == 1)                           // check write successful completion
      putchar(65) ;                       // display char A on hyperterminal


}



while (1)
     {

         read_byte = eeprom_read_byte(0x10);   // read data from specified eeprom address
         if(read_byte == 0x30)                 // verify read data with data written to eeprom
              putchar(read_byte);           // if write_data = read data, display on
hyperterminal

          read_byte = eeprom_read_byte(0x20);    // read data from specified eeprom
address
          if(read_byte == 0x31)                  // verify read data with data written to eeprom
              putchar(read_byte);               // if write_data = read data, display on
hyperterminal

        delay_ms(100);        // display data values every 100 msec on hyperterminal




     };
}
```