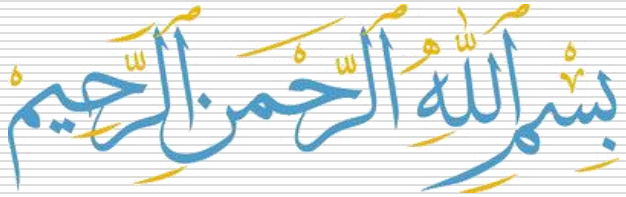

میکرو کنترلرهای AVR

مثالهایی به زبان C

دانشکده برق و رباتیک

دانشگاه صنعتی شاهرود

حسین خسروی

Example 1

Write a program to send ASCII code of characters 1,2,…,9,a,b,x,y to PORTB

```
#include <avr/io.h>

int main(void)
{
    unsigned char myList[]= "123456789abxy";
    unsigned char z;
    DDRB = 0xFF;
    for( z = 0; z < 13; z++)
         PORTB = myList[z] ;

    while(1);

    return 0;
}
```

Example 2

Write a program to send numbers -4 to +4 to PORTB

```
#include <avr/io.h>
int main(void)
{
   char mynum[] = {-4,-3,-2,-1,0,+1,+2,+3,+4};
   unsigned char z;
   DDRB = 0xFF;

   for(z = 0;z <= 8; z++)
      PORTB = mynum[z];
   while(1);
   return 0;
}
```

Output on ports: 0xFC, 0xFD, 0xFE, 0xFF, 0, 1, 2, 3, 4

Example 3

Toggle PORTB 100,000 times

```
#include <avr/io.h>
int main(void)
{
    unsigned long z;
    DDRB = 0xFF;
    for(z = 0; z < 100000; z++)
    {
      PORTB = 0x55;
      PORTB = 0xAA;
    }
    while(1);
    return 0;
}
```

int z; can not be used, why?

Example 4 – make delay using for loop

Toggle PORTB every 100ms

```
#include <avr/io.h>
void delay100ms(void);
int main(void)
{
    DDRB = 0xFF;
    PORTB = 0xAA;
    while (1)
    {
        delay100ms();
        PORTB = ~PINB;
    }
    return 0;
}
void delay100ms(void){
    unsigned int i ;
    for(i=0; i<42150; i++); // try different numbers on compiler
}
```

Remarks

- Two parameters affect on delay generated by for loop
  - Operating frequency of micro
  - Compiler
- How to generate exact delay?
  - Use timers
  - Use predefined functions (third party libraries)
    - _delay_ms() and _delay_us() in Atmel Studio
      - located in delay.h
    - delay_ms() and delay_us() in CodeVision

Example 5 – A better delay using built in functions

Use macro instead of function

```
#define F_CPU 8000000UL
#include <util/delay.h>
#include <avr/io.h>
#define delay_ms(d)  _delay_ms(d)//delay in milliseconds

int main(void)
{
   DDRB = 0xFF;
   while (1)
   {
      delay_ms(10);
      PORTB = ~ PORTB;
   }
   return 0;
}
```

Example 6 – sensor and lamp

A sensor is attached to PORTB1 and a LED to PORTC7. Write a program to trun-on LED whenever the sensor fires.

```
#include <avr/io.h>
int main(void)
{
    DDRB = DDRB & 0b11111101;
    DDRC = DDRC | 0b10000000;
    while(1)
    {
        if (PINB & (0b00000010))
            PORTC = PORTC | 0b10000000;
        else
            PORTC = PORTC & 0b01111111;
    }
    return 0;
}
```

Don’t forget indentation

Example 6 – using bitwise operators

0b10000000 = 1 << 7
0b11111101 = ~(1 << 1)

```
#include <avr/io.h>
#define LED 7
#define SENSOR 1
int main(void)
{
    DDRB = DDRB & ~(1 << SENSOR);
    DDRC = DDRC | (1 << LED);
    while(1)
    {
        if(PINB & (1 << SENSOR))
            PORTC |= (1 << LED);
        else
            PORTC &= ~(1 << LED);

    }
    return 0;
}
```

<avr/sfr_defs.h>: Special function registers

- `#define bit_is_set(sfr, bit) (_SFR_BYTE(sfr) & _BV(bit))`
- `#define bit_is_clear(sfr, bit) (!(_SFR_BYTE(sfr) & _BV(bit)))`
- `#define loop_until_bit_is_set(sfr, bit) do { } while (bit_is_clear(sfr, bit))`
- `#define loop_until_bit_is_clear(sfr, bit) do { } while (bit_is_set(sfr, bit))`

Example 7 – LCD (simple view)

Send a message to LCD. Data port of LCD is connected to PORTB.

The EN pin of LCD (attached to PORTC5) must see a H to L transition to latch data.

More detail will described later.

```
#include <avr/io.h>
int main(void){
    unsigned char message[] = “Speak less and Think more";
    unsigned char z ;
    DDRB = 0xFF;
    DDRC = DDRC | 0b00100000 ;
    for ( z = 0; z < 25; z++)
    {
        PORTB = message[z];
        PORTC = PORTC | 0b00100000;
        PORTC = PORTC & 0b11011111;
    }
    while (1);
    return 0;
}
```

Example 8 - Packed BCD to ASCII Conversion

Write a program to convert packed BCD 0x29 to ASCII and display the bytes on PORTB and PORTC

```
#include <avr/io.h>

int main(void)
{
   unsigned char x, y;
   unsigned char mybyte = 0x29;

   DDRB = DDRC = 0xFF;
   x = mybyte & 0x0F;
   PORTB = x | 0x30;
   y = mybyte & 0xF0;
   y = y >> 4;
   PORTC = y | 0x30;
   return 0;
}
```

Checksum Byte in ROM

Write a C program to calculate the checksum byte for the data 0x25, 0x62, 0x3F, and 0x52 and send it to PORTB.

```
#include <avr/io.h>
int main(void)
{
    unsigned char mydata[] = {0x25,0x62,0x3F,0x52};
    unsigned char sum = 0;
    unsigned char x;
    unsigned char chksumbyte;
    DDRB = 0xFF;
    for(x=0;x<4;x++)
        sum = sum + mydata[x];
    chksumbyte = ~sum + 1;
    PORTB = chksumbyte;
    return 0;
}
```

Checksum Byte in ROM

Write a program to perform the checksum operation to ensure data integrity. If data is good, send ASCII character 'G' to P0. Otherwise send 'B' to PORTD.

```
#include <avr/io.h>
int main(void)
{
    unsigned char mydata[] = {0x25,0x62,0x3F,0x52,0xE8};
    unsigned char chksum = 0;
    unsigned char x;
    DDRD = 0xFF;
    for(x=0;x<5;x++)
      chksum = chksum + mydata[x];
    if(chksum == 0)
      PORTD = 'G';
    else
      PORTD = 'B';
    return 0;
}
```

Binary (hex) to Decimal and ASCII Conversion

Write a program to convert 11111101 (FD hex) to decimal and display the digits on PORTB, C and D.

```
#include <avr/io.h>
int main(void)
{
    unsigned char x, binbyte, d1, d2, d3;
    DDRB = DDRC = DDRD =0xFF;
    binbyte = 0xFD;
    x = binbyte / 10;
    d1 = binbyte % 10;
    d2 = x % 10;
    d3 = x / 10;
    PORTB = d1;
    PORTC = d2;
    PORTD = d3;
    return 0;
}
```

These 6 lines could be shortened as follow:

```
PORTB  = binbyte % 10;
PORTC = x % 10;
PORTD = x / 10;
```

Built in functions for data conversion

atoi, atol, ltoa, atof, itoa and printf

```
int z = atoi("125");
float f = atof("14.75");
long l = atol("123000");
char* str = itoa(110);
```

Data Serialization

- Serializing data is a way of sending a byte of data one bit at a time through a single pin of microcontroller
  - Using the serial port (Chap. 11)
  - Transfer data one bit a time and control the sequence of data and spaces in between them
    - In many new generations of devices such as LCD, ADC, and ROM the serial versions are becoming popular since they take less space on a PCB

Example - Data Serialization

Write a C program to send out the value 0x44 serially one bit at a time via PORTC.3 . The LSB should go out first.

```
#include <avr/io.h>
#define serPin 3
int main(void)
{
    unsigned char conbyte = 0x44;
    DDRC |= (1<<serPin) ;
    for(unsigned char x=0; x < 8; x++)
      {
        if( conbyte & 0x01)
           PORTC |= (1<<serPin);
        else
           PORTC &= ~(1<<serPin);
        conbyte = conbyte >> 1;
      }
return 0;
}
```

Example - Data Serialization

Write a C program to send out the value 0x44 serially one bit at a time via PORTC.3 . The MSB should go out first.

```
#include <avr/io.h>
#define serPin 3
int main(void)
{
    unsigned char conbyte = 0x44;
    DDRC |= (1<<serPin) ;
    for(unsigned char x=0; x < 8; x++)
      {
        if(conbyte & 0x80)
           PORTC |= (1<<serPin);
        else
           PORTC &= ~(1<<serPin);
        conbyte = conbyte << 1;
      }
return 0;
}
```

Example – Receive Data

Write a C program to bring in a byte of data serially one bit at a time via PORTD.4 . The LSB should come in first.

```
#include <avr/io.h>
#define serPin 4
int main(void)
{
    unsigned char x;
    unsigned char data = 0;
    DDRD &= ~(1<<serPin);
    for(x = 0; x < 8; x++)
    {
       if(PIND &(1<<serPin))
           data |= 1 << x;
       /*data |= (PIND &(1<<serPin)) << (7-serPin);
       if(x < 7)
          data = data >> 1;*/
    }
    return 0;
}
```

Memory Assignment (RAM, EEPROM and Flash)

- RAM
  - By default, all variables are allocated in RAM
- Flash
  - Code will be written into Flash
  - To allocate some **constant** data in Flash
    - .DB can be used in assembly
    - For C, refer to the compiler
      - CodeVision: flash int IDs[20];
- EEPROM
  - Some compilers define simple way to access eeprom
    - CodeVision: eeprom int SecurityCode = 5426;
  - Standard way is somehow difficult!

EEPROM

- The EEPROM Address Register – EEARH and EEARL
- The EEPROM Data Register – EEDR
- The EEPROM Control Register – EECR
  - Bits 7..4 – Res: Reserved Bits
  - Bit 3 – EERIE: EEPROM Ready Interrupt Enable (Later)
  - Bit 2 – EEMWE: EEPROM Master Write Enable
    - Must be 1 for writing
  - Bit 1 – EEWE: EEPROM Write Enable
  - Bit 0 – EERE: EEPROM Read Enable

Bit	7	6	5	4	3	2	1	0
	–	–	–	–	EERIE	EEMWE	EEWE	EERE
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	X	0

Writing to EEPROM

- The following procedure should be followed when writing the EEPROM
  - 1. Wait until EEWE becomes zero.
  - 2. Write new EEPROM address to EEAR (optional).
  - 3. Write new EEPROM data to EEDR (optional).
  - 4. Write a logical one to the EEMWE bit while writing a zero to EEWE in EECR.
  - 5. Within **four clock cycles after setting EEMWE**, write a logical one to EEWE.

EEPROM Write

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
    /* Wait for completion of previous write */
    while(EECR & (1<<EEWE));
    /* Set up address and data registers */
    EEAR = uiAddress;
    EEDR = ucData;
    /* Write logical one to EEMWE */
    EECR |= (1<<EEMWE);
    /* Start eeprom write by setting EEWE */
    EECR |= (1<<EEWE);
}
```

EEPROM Read

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
    /* Wait for completion of previous write */
    while(EECR & (1<<EEWE))
    ;
    /* Set up address register */
    EEAR = uiAddress;
    /* Start eeprom read by writing EERE */
    EECR |= (1<<EERE);
    /* Return data from data register */
    return EEDR;
}
```

EEPROM using predefined functions

- Atmel Studio
  - <avr/eeprom.h>
  - uint8_t eeprom_read_byte (const uint8_t *p);
  - void eeprom_write_byte (uint8_t *p, uint8_t value);
  - void eeprom_read_block (void *dst, const void *src, size_t n);
  - void eeprom_write_float (float *p, float value);
  - …

- Run Sample Code

Something better

EEMEM attribute can be used instead of addressing, but still functions must be used.

```
#include <avr/eeprom.h>
uint8_t  EEMEM NonVolatileChar;
uint16_t EEMEM NonVolatileInt;
uint8_t  EEMEM NonVolatileString[10];

int main(void)
{
    uint8_t SRAMchar;

    SRAMchar = eeprom_read_byte(&NonVolatileChar);
}
```

تمرین کاغذی سری دوم

- Chapter 7
  - 1, 3, 6, 9, 12, 14, 16, 19, 22, 25, 28

- Due date: 1400/01/05