AVR Microcontroller

Microprocessor Course

Chapter 4

AVR I/O PORT PROGRAMMING

Aban 1401

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

Examine Figure 4-1 for the ATmega32 40-pin chip. A total of 32 pins are set aside for the four ports PORTA, PORTB, PORTC, and PORTD. The rest of the pins are designated as VCC, GND, XTAL1, XTAL2, RESET, AREF, AGND, and AVCC.

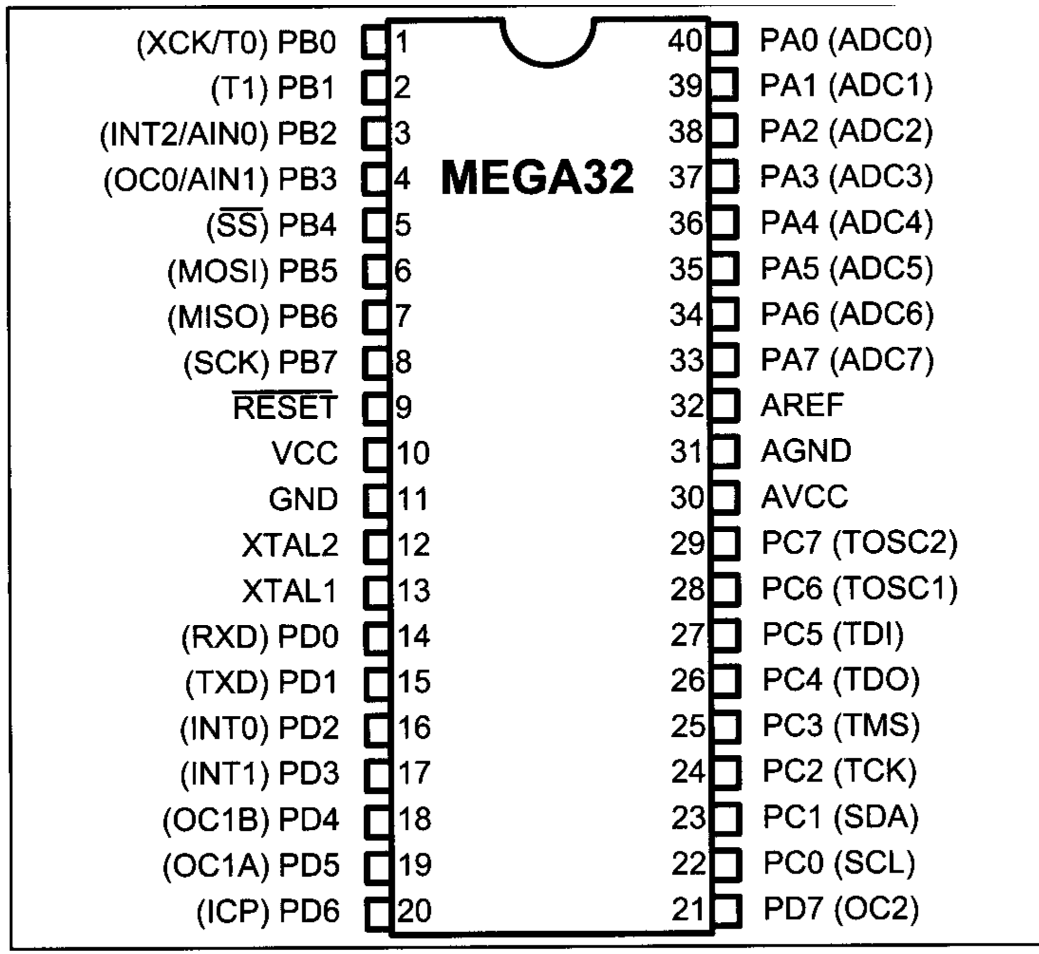


Figure 4-1. ATmega32 Pin Diagram

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

I/O port pins and their functions

The number of ports in the AVR family varies depending on the number of pins on the chip.

- The 8-pin AVR has port B only,
- while the 64-pin version has ports A through F,
- and the 100-pin AVR has ports A through L, as shown in Table 4-1.
- The 40-pin AVR has four ports.
  They are PORTA, PORTB, PORTC, and PORTD.

To use any of these ports as an input or output port, it must be programmed.

In addition to being used for simple I/O, each port has some other functions such as ADC, timers, interrupts, and serial communication pins.

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

Table 4-1: Number of Ports in Some AVR Family Members

Pins	**8-pin**	**28-pin**	**40-pin**	**64-pin**	**100-pin**
Chip	**ATtiny25/45/85**	**ATmega8/48/88**	**ATmega32/16**	**ATmega64/128**	**ATmega1280**
Port A			X	X	X
Port B	6 bits	X	X	X	X
Port C		7 bits	X	X	X
Port D		X	X	X	X
Port E				X	X
Port F				X	X
Port G				5 bits	6 bits
Port H					X
Port J					X
Port K					X
Port L					X

Note: X indicates that the port is available.

Figure 4-1 shows alternate functions for the ATmega32 pins.

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

For example, for Port B we have PORTB, DDRB, and PINB. Notice that DDR stands for Data Direction Register, and PIN stands for Port INput pins. Also notice that each of the I/O registers is 8 bits wide, and each port has a maximum of 8 pins; therefore PORTD each bit of the I/O registers affects one of the direction pins

Table 4-2: Register Addresses for ATmega32 Ports

Port	Address	Usage
PORTA	$3B	output
DDRA	$3A	direction
PINA	$39	input
PORTB	$38	output
DDRB	$37	direction
PINB	$36	input
PORTC	$35	output
DDRC	$34	direction
PINC	$33	input
PORTD	$32	output
DDRD	$31	direction
PIND	$30	input

AVR I/O PORT PROGRAMMING
4.1 I/O PORT PROGRAMMING IN AVR

DDRx register role in outputting data

The DDRx I/O register is used solely for the purpose of making a given port an input or output port. For example, to make a port an output, we write 1s to the DDRx register. In other words, to output data to all of the pins of the Port B, we must first put 0b11111111 into the DDRB register.

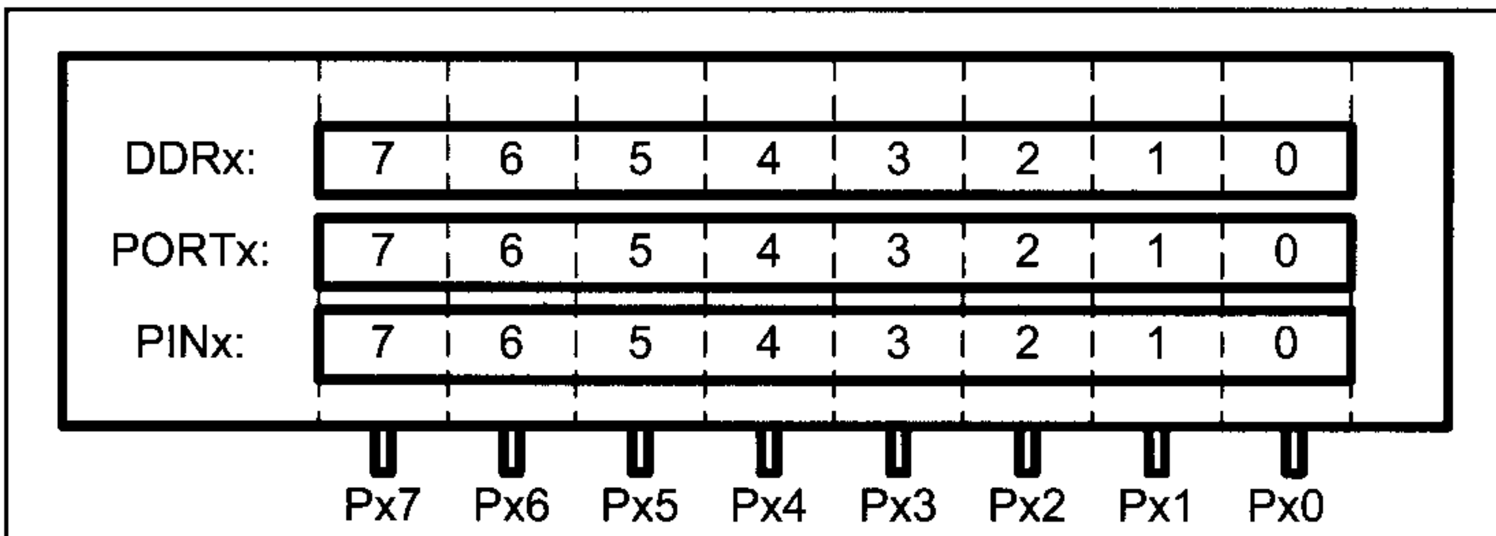


Figure 4-2. Relations Between the Registers and the Pins of AVR

AVR I/O PORT PROGRAMMING
4.1 I/O PORT PROGRAMMING IN AVR

The following code will toggle all 8 bits of Port B forever with some time delay between “on” and “off” states:

```
        LDI     R16,0xFF        ;R16 = OxFF = Ob11111111
        OUT     DDRB,R16        ;make Port B an output port (1111 1111)
L1:     LDI     R16,0x55        ;R16 = 0x55 = Ob01010101
        OUT     PORTB,R16       ;put 0x55 on port B pins
        CALL    DELAY
        LDI     R16,0xAA        ;R16 = OxAA = Ob10101010
        OUT     PORTB,R16       ;put OxAA on port B pins
        CALL    DELAY
        RJMP    L1
```

It must be noted that unless we set the DDRx bits to one, the data will not go from the port register to the pins of the AVR. This means that if we remove the first two lines of the above code, the 0x55 and 0xAA values will not get to the pins. They will be sitting in the I/O register of Port B inside the CPU.

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

DDR register role in inputting data

To make a port an input port, we must first put 0s into the DDRx register for that port, and then bring in (read) the data present at the pins.

Notice that upon reset, all ports have the value 0x00 in their DDR registers. This means that all ports are configured as input.

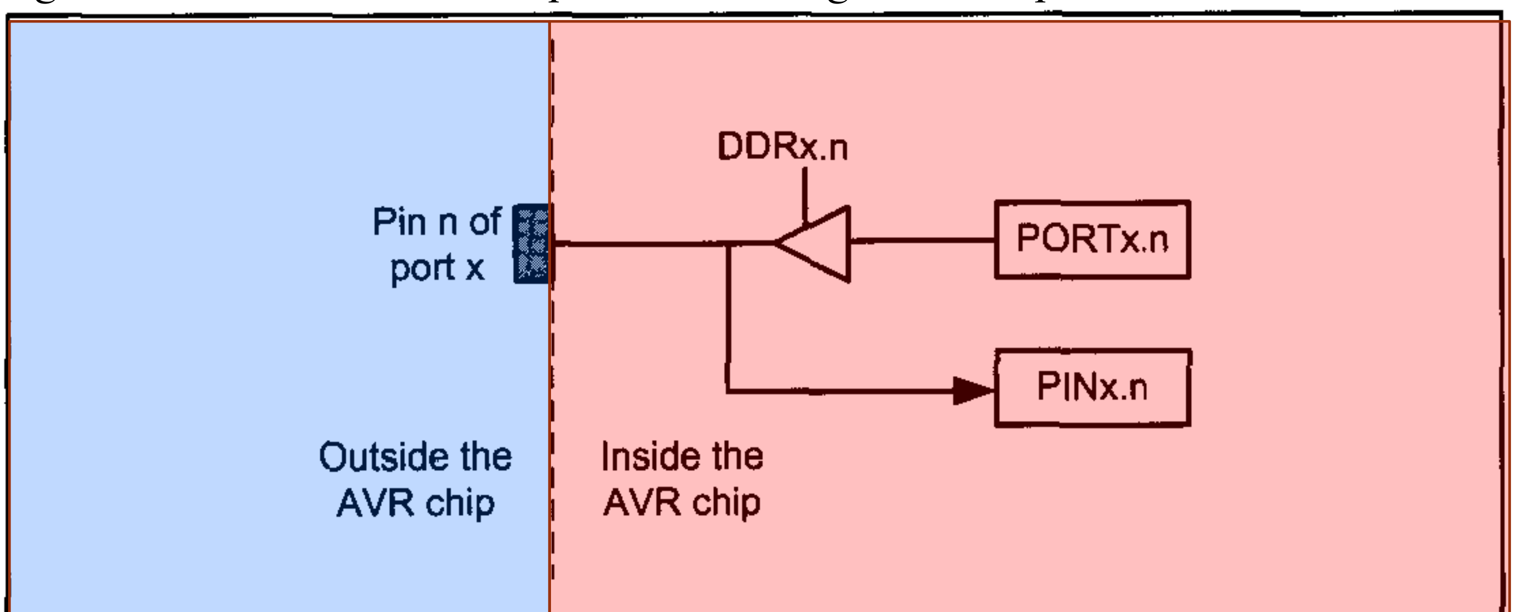


Figure 4-3. The I/O Port in AVR

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

PIN register role in inputting data

To read the data present at the pins, we should read the PIN register. It must be noted that to bring data into CPU from pins we read the contents of the PINx register, whereas to send data out to pins we use the PORTx register.

There is a pull-up resistor for each of the AVR pins. If we put 1s into bits of the PORTx register, the pullup resistors are activated. In cases in which nothing is connected to the pin or the connected devices have high impedance, the resistor pulls up the pin.

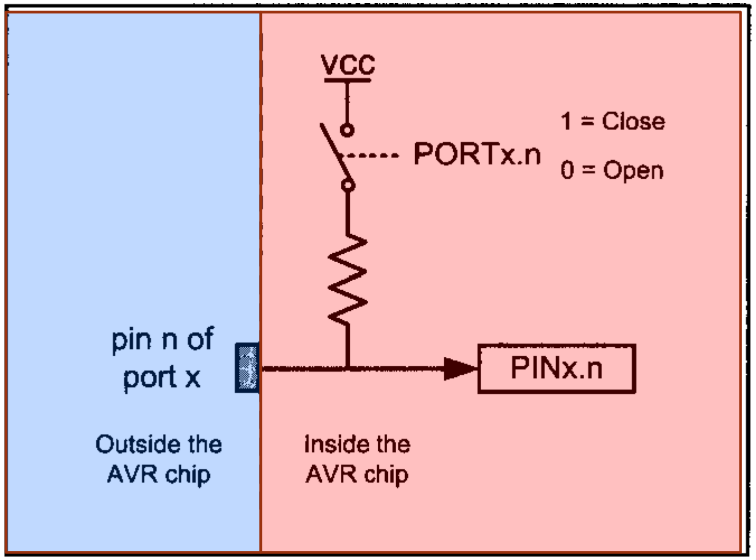


Figure 4-4. The Pull-up Resistor

AVR I/O PORT PROGRAMMING
4.1 I/O PORT PROGRAMMING IN AVR

If we want to make the pull-up resistors of port C active, we must put 1s into the PORTC register. The program becomes as follows:

```
.INCLUDE     "M32DEF.INC"
        LDI     R16,0xFF          ;R16 = 11111111 (binary)
        OUT     DDRB,R16          ;make Port B an output port
        OUT     PORTC,R16         ;make the pull-up resistors of C active
        LDI     R16,0x00          ;R16 = 00000000 (binary)
        OUT     DDRC,R16          ;Port C an input port (0 for I)
L2:     IN      R16,PINC          ;move data from Port C to R16
        LDI     R17,5
        ADD     R16,R17           ;add some value to it
        OUT     PORTB,R16         ;send it to Port B
        RJMP    L2                ;continue forever
```


mashhoun@iust.ac.ir Iran Univ of Science & Tech

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

The pins of the AVR microcontrollers can be in four different states according to the values of PORTx and DDRx, as shown in Figure 4-5.

PORTx \ DDRx	0	1
0	Input & high impedance	Out 0
1	Input & pull-up	Out 1

Figure 4-5. Different States of a Pin in the AVR Microcontroller

This is one of powerful features of the AVR microcontroller, since most of the other microcontrollers' pins (e.g., 8051) have fewer states.

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

Port A

Port A occupies a total of 8 pins (PA0-PA7). To use the pins of Port A as input or output ports, each bit of the DDRA register must be set to the proper value. For example, the following code will continuously send out to Port A the alternating values of 0x55 and 0xAA:

```
;toggle all bits of PORTA
.INCLUDE "M32DEF.INC"
        LDI     R16,0xFF        ;R16 = 11111111 (binary)
        OUT     DDRA,R16        ;make Port A an output port
L1:     LDI     R16,0x55        ;R16 = 0x55
        OUT     PORTA,R16       ;put 0x55 on Port A pins
        CALL    DELAY
        LDI     R16,0xAA        ;R16 = 0xAA
        OUT     PORTA,R16       ;put 0xAA on Port A pins
        CALL    DELAY
        RJMP    L1
```

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

Port A as input

In order to make all the bits of Port A an input, DDRA must be cleared by writing 0 to all the bits. In the following code, Port A is configured first as an input port by writing all 0s to register DDRA, and then data is received from Port A and saved in a RAM location:

```
.INCLUDE "M32DEF.INC"
.EQU MYTEMP 0x100                   ;save it here
        LDI     R16,0x00            ;R16 = 00000000 (binary)
        OUT     DDRA,R16            ;make Port A an input port (0 for In)
        NOP                         ;synchronizer delay
        IN      R16, PINA           ;move from pins of Port A to RI6
        STS     MYTEMP,R16          ;save it in MYTEMP
```

AVR I/O PORT PROGRAMMING
4.1 I/O PORT PROGRAMMING IN AVR

Port B

Port B occupies a total of 8 pins (PB0-PB7). To use the pins of Port B as input or output ports, each bit of the DDRB register must be set to the proper value.

For example, the following code will continuously send out the alternating values of 0x55 and 0xAA to Port B:

```
;toggle all bits of PORTA
.INCLUDE "M32DEF.INC"
        LDI     R16,0xFF        ;R16 = 11111111 (binary)
        OUT     DDRB,R16        ;make Port B an output port
L1:     LDI     R16,0x55        ;R16 = 0x55
        OUT     PORTB,R16       ;put 0x55 on Port B pins
        CALL    DELAY
        LDI     R16,0xAA        ;R16 = 0xAA
        OUT     PORTB,R16       ;put 0xAA on Port B pins
        CALL    DELAY
        RJMP    L1
```

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

Port B as input

In order to make all the bits of Port B an input, DDRB must be cleared by writing 0 to all the bits. In the following code, Port B is configured first as an input port by writing all 0s to register DDRB, and then data is received from Port B and saved in some RAM location:

```
.INCLUDE "M32DEF.INC"
.EQU MYTEMP 0x100                  ;save it here
      LDI      R16,0x00            ;R16 = 00000000 (binary)
      OUT      DDRB,R16            ;make Port B an input port (0 for In)
      NOP                          ;synchronizer delay
      IN       R16, PINB           ;move from pins of Port B to RI6
      STS      MYTEMP,R16          ;save it in MYTEMP
```


mashhoun@iust.ac.ir

AVR I/O PORT PROGRAMMING
4.1 I/O PORT PROGRAMMING IN AVR

The AVR multiplexes an analog-to-digital converter through Port A to save I/O pins. The alternate functions of the pins for Port A are shown in Table 4-3.

The AVR multiplexes some other functions through Port B to save pins. The alternate functions of the pins for Port B are shown in Table 4-4.

Table 4-3: Port A Alternate Functions

Bit	Function
PA0	ADC0
PA1	ADC1
PA2	ADC2
PA3	ADC3
PA4	ADC4
PA5	ADC5
PA6	ADC6
PA7	ADC7

Table 4-4: Port B Alternate Functions

Bit	Function
PB0	XCK/T0
PB1	T1
PB2	INT2/AIN0
PB3	OC0/AIN1
PB4	SS
PB5	MOSI
PB6	MISO
PB7	SCK

AVR I/O PORT PROGRAMMING
4.1 I/O PORT PROGRAMMING IN AVR

Port C

Port C occupies a total of 8 pins (PCO-PC7). To use the pins of Port C as input or output ports, each bit of the DDRC register must be set to the proper value. For example, the following code will continuously send out the alternating values of 0x55 and 0xAA to Port C:

```
;toggle all bits of PORTA
.INCLUDE "M32DEF.INC"
        LDI     R16,0xFF            ;R16 = 11111111 (binary)
        OUT     DDRC,R16            ;make Port C an output port
L1:     LDI     R16,0x55            ;R16 = 0x55
        OUT     PORTC,R16           ;put 0x55 on Port C pins
        CALL    DELAY
        LDI     R16,0xAA            ;R16 = 0xAA
        OUT     PORTC,R16           ;put 0xAA on Port C pins
        CALL    DELAY
        RJMP    L1
```

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

Port C as input

In order to make all the bits of Port C an input, DDRC must be cleared by writing 0 to all the bits. In the following code, Port C is configured first as an input port by writing all 0s to register DDRC, and then data is received from Port C and saved in a RAM location:

```
.INCLUDE "M32DEF.INC"
.EQU MYTEMP 0x100                   ;save it here
        LDI     R16,0x00            ;R16 = 00000000 (binary)
        OUT     DDRC,R16            ;make Port C an input port (0 for In)
        NOP                         ;synchronizer delay
        IN      R16, PINC           ;move from pins of Port C to RI6
        STS     MYTEMP,R16          ;save it in MYTEMP
```

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

Port D

Port D occupies a total of 8 pins (PD0-PD7). To use the pins of Port D as input or output ports, each bit of the DDRD register must be set to the proper value. For example, the following code will continuously send out to Port D the alternating values of 0x55 and 0xAA:

```
;toggle all bits of PORTA
.INCLUDE "M32DEF.INC"
        LDI     R16,0xFF            ;R16 = 11111111 (binary)
        OUT     DDRD,R16            ;make Port D an output port
L1:     LDI     R16,0x55            ;R16 = 0x55
        OUT     PORTD,R16           ;put 0x55 on Port D pins
        CALL    DELAY
        LDI     R16,0xAA            ;R16 = 0xAA
        OUT     PORTD,R16           ;put 0xAA on Port D pins
        CALL    DELAY
        RJMP    L1
```


mashhoun@iust.ac.ir Iran Univ of Science & Tech

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

Port D as input

In order to make all the bits of Port D an input. DDRD must be cleared by, writing 0 to all the bits. In the following code, Port D is configured first as an input port by writing all 0s to register DDRD, and then data is received from Port D and saved in a RAM location:

```
.INCLUDE "M32DEF.INC"
.EQU MYTEMP 0x100                       ;save it here
        LDI     R16,0x00                ;R16 = 00000000 (binary)
        OUT     DDRD,R16                ;make Port D an input port (0 for In)
        NOP                             ;synchronizer delay
        IN      R16, PIND               ;move from pins of Port D to RI6
        STS     MYTEMP,R16              ;save it in MYTEMP
```

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

Dual role of Ports C and D

The alternate functions of the pins for Port C are shown in Table 4-5. The alternate functions of the pins for Port D are shown in Table 4-6.

Table 4-5: Port C Alternate Functions

Bit	Function
PC0	SCL
PC1	SDA
PC2	TCK
PC3	TMS
PC4	TDO
PC5	TDI
PC6	TOSC1
PC7	TOSC2

Table 4-6: Port D Alternate Functions

Bit	Function
PD0	PSP0/C1IN+
PD1	PSP1/C1IN-
PD2	PSP2/C2IN+
PD3	PSP3/C2IN-
PD4	PSP4/ECCP1/P1A
PD5	PSP5/P1B
PD6	PSP6/P1C
PD7	PSP7/P1D

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

Example 4-1

Write a test program for the AVR chip to toggle all the bits of PORTB, PORTC, and PORTD every 1/4 of a second. Assume a crystal frequency of 1 MHz.

Solution:

```
;tested with AVR Studio for the ATmega32 and XTAL = 1 MHz
;to select the XTAL frequency in AVR Studio, press ALT+O
.INCLUDE "M32DEF.INC"
        LDI     R16, HIGH(RAMEND)
        OUT     SPH, R16
        LDI     R16, LOW(RAMEND)
        OUT     SPL, R16                ;initialize stack pointer

        LDI     R16, 0xFF
        OUT     DDRB, R16               ;make Port B an output port
        OUT     DDRC, R16               ;make Port C an output port
        OUT     DDRD, R16               ;make Port D an output port

        LDI     R16, 0x55               ;R16 = 0x55
L3:     OUT     PORTB, R16              ;put 0x55 on Port B pins
        OUT     PORTC, R16              ;put 0x55 on Port C pins
        OUT     PORTD, R16              ;put 0x55 on Port D pins
        CALL    QDELAY                  ;quarter of a second delay
        COM     R16                     ;complement R16
        RJMP    L3
```

AVR I/O PORT PROGRAMMING

4.1 I/O PORT PROGRAMMING IN AVR

```
;------------1/4 SECOND DELAY
QDELAY:
            LDI     R21, 200
D1:         LDI     R22, 250
D2:         NOP
            NOP
            DEC     R22
            BRNE    D2
            DEC     R21
            BRNE    D1
            RET
```

Calculations:

1 / 1 MHz = 1 μS

Delay = 200 × 250 × 5 MC × 1 μs = 250,000 μs (If we include the overhead, we will have 250,608 in. See Example 3-18 in the previous chapter.)

Use the AVR Studio simulator to verify the delay size.

AVR I/O PORT PROGRAMMING
4.2: I/O BIT MANIPULATION PROGRAMMING

I/O ports and bit-addressability

Sometimes we need to access only 1 or 2 bits of the port instead of the entire 8 bits. A powerful feature of AVR I/O ports is their capability to access individual bits of the port without altering the rest of the bits in that port. Table 4-7 lists the single-bit instructions for the AVR.

Table 4-7: Single-Bit (Bit-Oriented) Instructions for AVR

Instruction		Function
SBI	ioReg,bit	Set Bit in I/O register (set the bit: bit = 1)
CBI	ioReg,bit	Clear Bit in I/O register (clear the bit: bit = 0)
SBIC	ioReg,bit	Skip if Bit in I/O register Cleared (skip next instruction if bit = 0)
SBIS	ioReg,bit	Skip if Bit in I/O register Set (skip next instruction if bit = 1)

AVR I/O PORT PROGRAMMING

4.2: I/O BIT MANIPULATION PROGRAMMING

Although the instructions in Table 4-7 can be used for any of the lower 32 I/O registers, I/O port operations use them most often. We will see the use of these instructions throughout future chapters. Table 4-8 shows the lower 32 I/O registers

Address Mem.	Address I/O	Name
$20	$00	TWBR
$21	$01	TWSR
$22	$02	TWAR
$23	$03	TWDR
$24	$04	ADCL
$25	$05	ADCH
$26	$06	ADCSRA
$27	$07	ADMUX
$28	$08	ACSR
$29	$09	UBRRL
$2A	$0A	UCSRB

Address Mem.	Address I/O	Name
$2B	$0B	UCSRA
$2C	$0C	UDR
$2D	$0D	SPCR
$2E	$0E	SPSR
$2F	$0F	SPDR
$30	$10	PIND
$31	$11	DDRD
$32	$12	PORTD
$33	$13	PINC
$34	$14	DDRC
$35	$15	PORTC

Address Mem.	Address I/O	Name
$36	$16	PINB
$37	$17	DDRB
$38	$18	PORTB
$39	$19	PINA
$3A	$1A	DDRA
$3B	$1B	PORTA
$3C	$1C	EECR
$3D	$1D	EEDR
$3E	$1E	EEARL
$3F	$1F	EEARH

Table 4-8: The Lower 32 I/O Registers

AVR I/O PORT PROGRAMMING

4.2: I/O BIT MANIPULATION PROGRAMMING

SBI (set bit in I/O register)

To set HIGH a single bit of a given I/O register, we use the following syntax:

```
SBI   ioReg,bit num
```

where ioReg can be the lower 32 I/O registers (addresses 0 to 31) and bit_num is the desired bit number from 0 to 7. Although the bit-oriented instructions can be used for manipulation of bits D0-D7 of the lower 32 I/O registers, they are mostly used for I/O ports. For example the following instruction sets HIGH bit 5 of Port B:

```
SBI   PORTB,5
```

AVR I/O PORT PROGRAMMING

4.2: I/O BIT MANIPULATION PROGRAMMING

In Figure 4-6, you see the SBI instruction format.

SBI a, b

1001	1010	aaaa	abbb

$0 \leq a \leq 31$

$0 \leq b \leq 7$

Figure 4-6. SBI (Set Bit) Instruction Format

mashhoun@iust.ac.ir Iran Univ of Science & Tech

AVR I/O PORT PROGRAMMING

4.2: I/O BIT MANIPULATION PROGRAMMING

CBI (Clear Bit in I/O register)

To clear a single bit of a given I/O register, we use the following syntax:

```
CBI   ioReg,bit num
```

For example the following instruction toggles pin PB2 continuously:

```
       SBI  DDRB,2         ;bit =1,make PB2 an output
AGAIN: SBI  PORTB,2        ;bit set (PB2=high)
       call delay
       CBI  PORTB,2        ;bit clear (PB2=low)
       call delay
       RJMP AGAIN
```

AVR I/O PORT PROGRAMMING
4.2: I/O BIT MANIPULATION PROGRAMMING

In Figure 4-7, you see the CBI instruction format.

CBI a, b

1001	1000	aaaa	abbb

$0 \leq a \leq 31$

$0 \leq b \leq 7$

Figure 4-7. CBI (Clear Bit) Instruction Format

AVR I/O PORT PROGRAMMING

4.2: I/O BIT MANIPULATION PROGRAMMING

Notice that PB2 is the third bit of Port B (the first bit is PB0, the second bit is PB1, etc). This is shown in Table 4-9.

Table 4-9: Single-Bit Addressability of Ports for ATmega32/16

PORT	PORTB	PORTC	PORTD	Port Bit
PA0	PB0	PC0	PD0	D0
PA1	PB1	PC1	PD1	D1
PA2	PB2	PC2	PD2	D2
PA3	PB3	PC3	PD3	D3
PA4	PB4	PC4	PD4	D4
PA5	PB5	PC5	PD5	D5
PA6	PB6	PC6	PD6	D6
PA7	PB7	PC7	PD7	D7

AVR I/O PORT PROGRAMMING

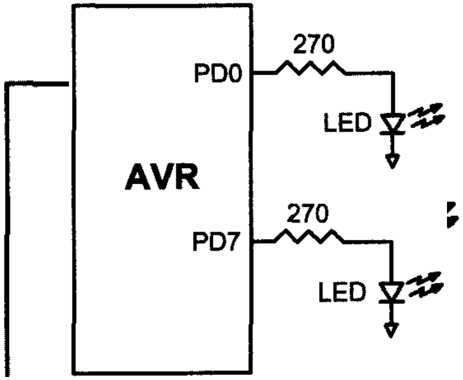
4.2: I/O BIT MANIPULATION PROGRAMMING

Example 4-2

An LED is connected to each pin of Port D. Write a program to turn on each LED from pin D0 to pin D7. Call a delay subroutine before turning on the next LED.

Solution:

```
.INCLUDE "M32DEF.INC"
LDI    R20, HIGH(RAMEND)
OUT    SPH, R20
LDI    R20, LOW(RAMEND)
OUT    SPL, R20     ;initialize stack pointer
LDI    R20, 0xFF
OUT    PORTD, R20   ;make PORTD an output port
SBI    PORTD,0      ;set bit PD0
CALL   DELAY        ;delay before next one
SBI    PORTD,1      ;turn on PD1
CALL   DELAY        ;delay before next one
SBI    PORTD,2      ;turn on PD2
CALL   DELAY
SBI    PORTD,3
CALL   DELAY
SBI    PORTD,4
CALL   DELAY
SBI    PORTD,5
CALL   DELAY
SBI    PORTD,6
CALL   DELAY
SBI    PORTD,7
CALL   DELAY
```

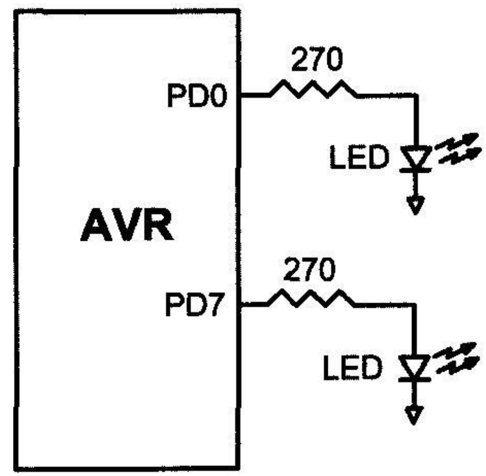


AVR I/O PORT PROGRAMMING
4.2: I/O BIT MANIPULATION PROGRAMMING

Example 4-2

An LED is connected to each pin of Port D. Write a program to turn on each LED from pin DO to pin D7. Call a delay subroutine before turning on the next LED.

```
.INCLUDE "M32DEF.INC"
        LDI     R20, HIGH(RAMEND)
        OUT     SPH, R20
        LDI     r20, LOW(RAMEND)
        OUT     SPL, R20            ;initialize stack pointer
        LDI     R20,0xFF
        OUT     PORTD, R20          ;make PORTD an output port
        SBI     PORTD,0             ;set bit PD0
        CALL    DELAY               ;delay before next one
        SBI     PORTD,1             ;turn on PD1
        CALL    DELAY               ;delay before next one
        SBI     PORTD,2             ;turn on PD2
        CALL    DELAY
        SBI     PORTD,3
        CALL    DELAY
        SBI     PORTD,4
        CALL    DELAY
        SBI     PORTD,5
        CALL    DELAY
        SBI     PORTD,6
        CALL    DELAY
        SBI     PORTD,7
        CALL    DELAY
```



Solution:

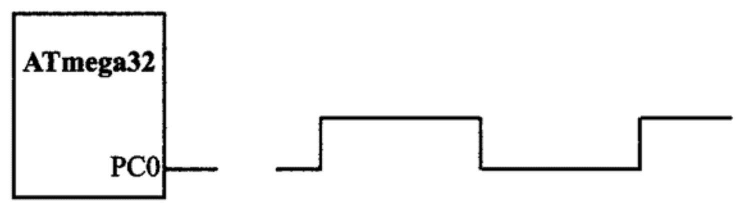
AVR I/O PORT PROGRAMMING
4.2: I/O BIT MANIPULATION PROGRAMMING

Example 4-3

Write the following programs: (a) Create a square wave of 50% duty cycle on bit 0 of Port C. (b) Create a square wave of 66% duty cycle on bit 3 of Port C.

Solution:

(a) The 50% duty cycle means that the "on" and "off" states (or the high and low portions of the pulse) have the same length. Therefore, we toggle PCO with a time delay between each state.



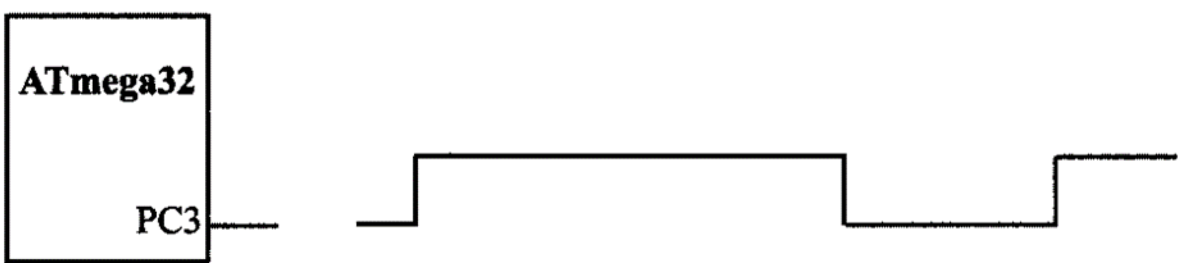

```
.INCLUDE "M32DEF.INC"
        LDI     R20, HIGH(RAMEND)
        OUT     SPH, R20
        LDI     R20, LOW(RAMEND)
        OUT     SPL, R20          ;initialize stack pointer
        SBI     DDRC,0            ;set bit 0 of DDRC (PCO = out)
HERE:   SBI     PORTC,0           ;set to HIGH PCO (PCO = 1)
        CALL    DELAY             ;call the delay subroutine
        CBI     PORTC,0           ;PCO = 0
        CALL    DELAY
        RJMP    HERE              ;keep doing it
```

AVR I/O PORT PROGRAMMING
4.2: I/O BIT MANIPULATION PROGRAMMING

(b) A 66% duty cycle means that the "on" state is twice the "off" state.

```

        SBI     DDRC,3                  ;set bit 3 of DDRC (PC3 = out)
HERE:   SBI     PORTC,3                 ;set to HIGH PC3 (PC3 = 1)
        CALL    DELAY                   ;call the delay subroutine
        CALL    DELAY                   ;call the delay subroutine
        CBI     PORTC,3                 ;PC3 = 0
        CALL    DELAY
        RJUMP   HERE                    ;keep doing it
```



AVR I/O PORT PROGRAMMING

4.2: I/O BIT MANIPULATION PROGRAMMING

SBIS (Skip if Bit in I/O register Set)

To monitor the status of a single bit for HIGH, we use the SBIS instruction. This instruction tests the bit and skips the next instruction if it is HIGH.

SBIS a,b

1001	1011	aaaa	abbb

$0 \leq a \leq 31$

$0 \leq b \leq 7$

Figure 4-8. SBIS (Skip If Bit in I/O Register Set) Instruction Format

AVR I/O PORT PROGRAMMING
4.2: I/O BIT MANIPULATION PROGRAMMING

Example 4-4

Write a program to perform the following: (a) Keep monitoring the PB2 bit until it becomes HIGH; (b) When PB2 becomes HIGH, write the value $45 to Port C, and also send a HIGH-to-LOW pulse to PD3.

Solution:

```
.INCLUDE "M32DEF.INC"

        CBI     DDRB, 2         ;make PB2 an input
        LDI     R16,0xFF
        OUT     DDRC,R16        ;make Port C an output port
        SBI     DDRD,3          ;make PD3 an output
AGAIN:  SBIS    PINB,2          ;skip if Bit PB2 is HIGH
        RJMP    AGAIN           ;keep checking if LOW
        LDI     R16,0x45        ;write 0x45 to port C
        OUT     PORTC,R16
        SBI     PORTD,3         ;set bit PD3 (H-to-L)
        CBI     PORTD,3         ;clear bit PD3
HERE:   RJMP    HERE
```

AVR I/O PORT PROGRAMMING

4.2: I/O BIT MANIPULATION PROGRAMMING

In this program, "SBIS PINB, 2" instruction stays in the loop as long as PB2 is LOW. When PB2 becomes HIGH, it skips the branch instruction to get out of the loop, and writes the value $45 to Port C. It also sends a HIGH-to-LOW pulse to PD3.

AVR I/O PORT PROGRAMMING

4.2: I/O BIT MANIPULATION PROGRAMMING

SBIC (Skip if Bit in I/O register Cleared)

To monitor the status of a single bit for LOW, we use the SBIC instruction. This instruction tests the bit and skips the instruction right below it if the bit is LOW.

SBIC a,b

1001	1001	aaaa	abbb

$0 \leq a \leq 31$

$0 \leq b \leq 7$

Figure 4-9. SBIC (Skip if Bit in I/O Register Cleared) Instruction Format

AVR I/O PORT PROGRAMMING
4.2: I/O BIT MANIPULATION PROGRAMMING

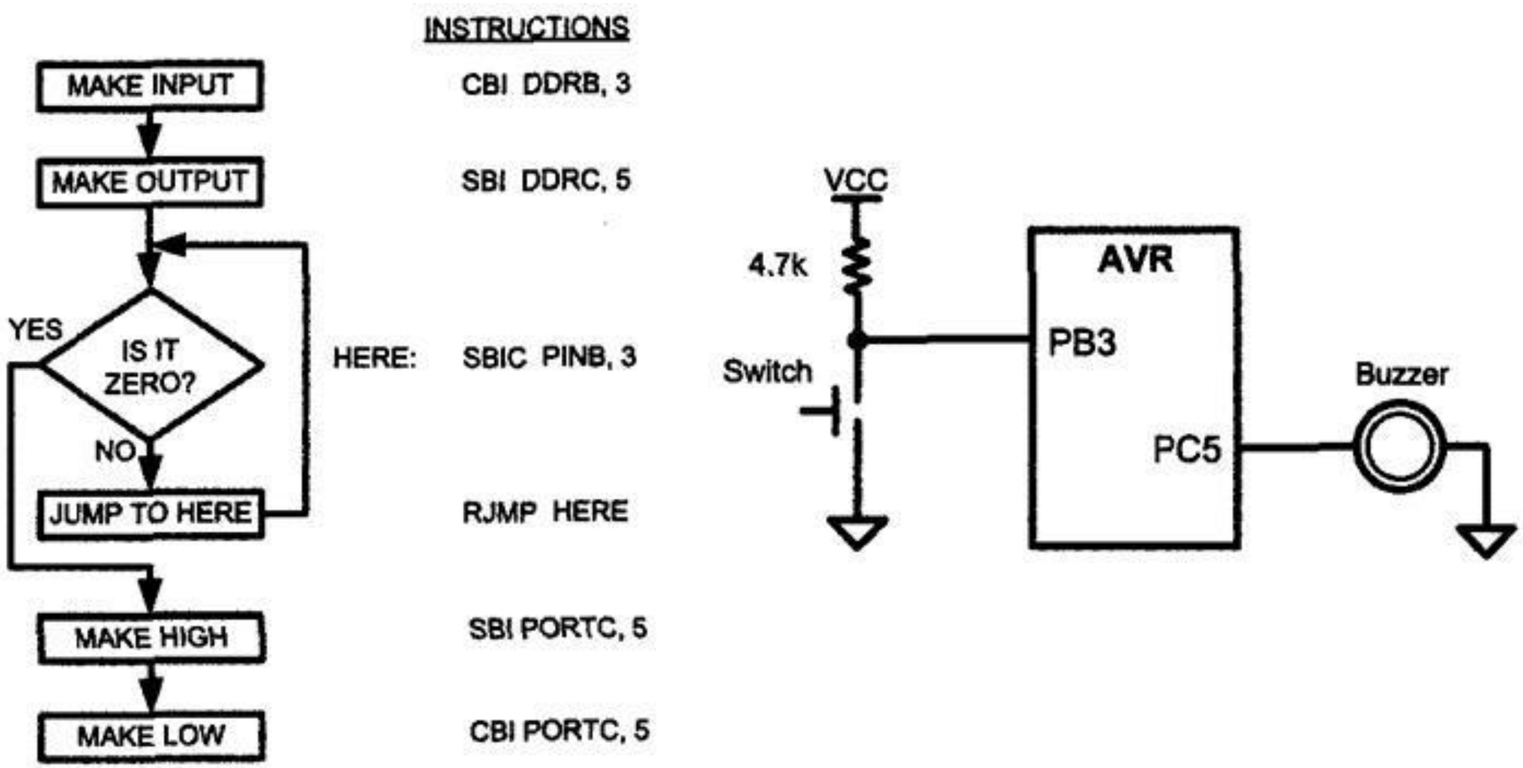
Example 4-5

Assume that bit PB3 is an input and represents the condition of a door alarm. If it goes LOW, it means that the door is open. Monitor the bit continuously. Whenever it goes LOW, send a HIGH-to-LOW pulse to port PC5 to turn on a buzzer.

Solution:

```
.INCLUDE "M32DEF.INC"
        CBI     DDRB,3          ;make PB3 an input
        SBI     DDRC,5          ;make PC5 an output
HERE:   SBIC    PINS,3          ;keep monitoring PB3 for HIGH
        RJMP    HERE            ;stay in the loop
        SBI     PORTC,5         ;make PC5 HIGH
        CBI     PORTC,5         ;make PC5 LOW for H-to-L
        RJMP    HERE
```

AVR I/O PORT PROGRAMMING
4.2: I/O BIT MANIPULATION PROGRAMMING



AVR I/O PORT PROGRAMMING
4.2: I/O BIT MANIPULATION PROGRAMMING

Example 4-b

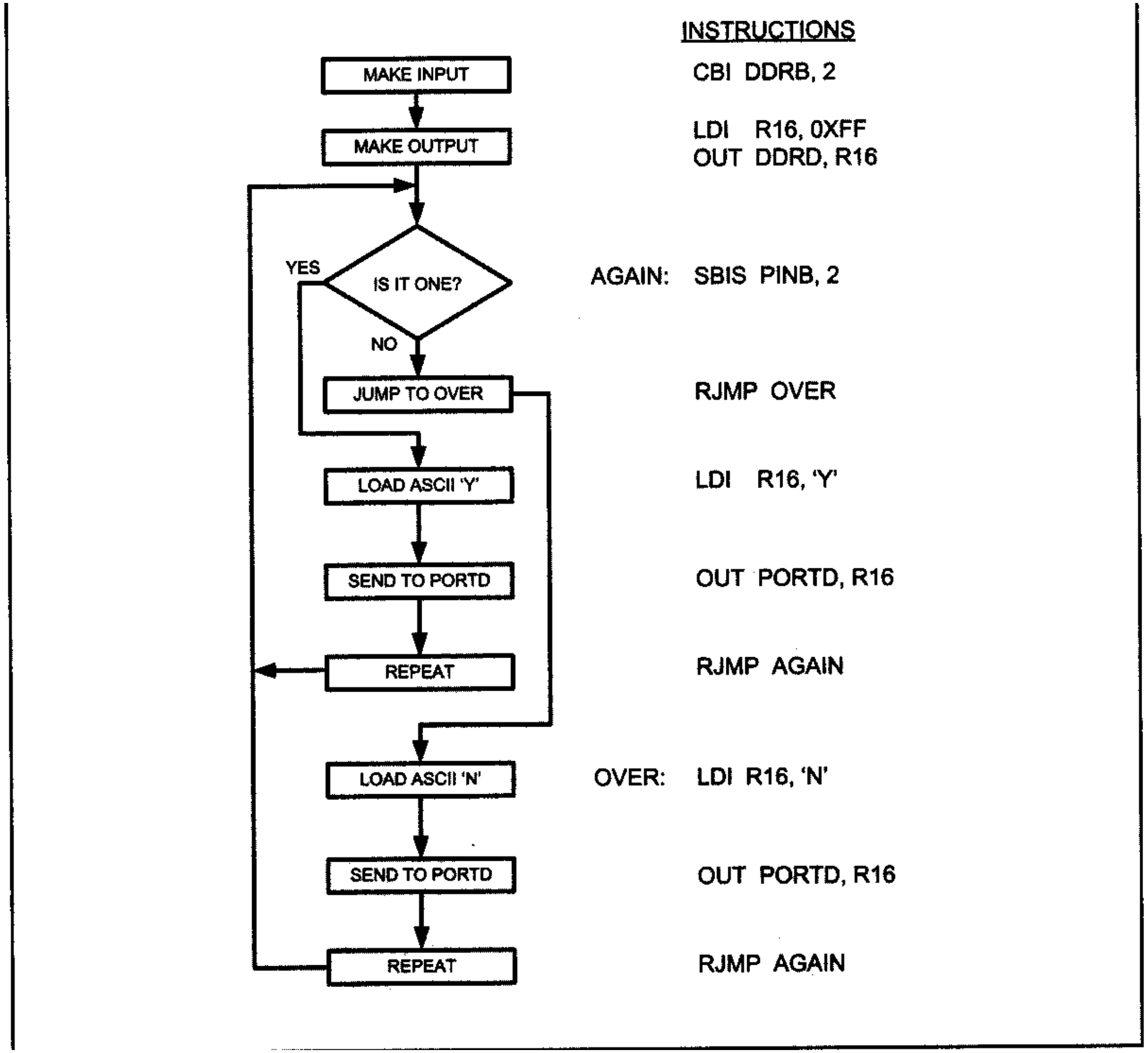
A switch is connected to pin PB2. Write a program to check the status of SW and perform the following: (a) If SW = 0, send the letter 'N' to PORTD. (b) If SW = 1, send the letter 'Y' to PORTD.

Solution:

```
.INCLUDE "M32DEF.INC"
        CBI     DDRB,2              ;make PB2 an input
        LDI     R16,0xFF
        OUT     DDRD, R16           ;make PORTD an output port

AGAIN:  SBIS    PINB,2              ;skip next if PB bit is HIGH
        RJMP    OVER                ;SW is LOW
        LDI     R16,'Y'             ;R16 = 'Y' (ASCII ;PORTD = 'Y')
        OUT     PORTD, R16          ;PORTD = 'Y'
        RJMP    AGAIN
OVER:   LDI     R16,'N'             ;R16 = 'N' (ASCII letter N)
        OUT     PORTD               ;PORTD = 'N'
        RJMP    AGAIN
```

AVR I/O PORT PROGRAMMING
4.2: I/O BIT MANIPULATION PROGRAMMING



AVR I/O PORT PROGRAMMING

4.2: I/O BIT MANIPULATION PROGRAMMING

Example 4-7

Rewrite the program of Example 4-6, using the SBIC instruction instead of SBIS.

Solution:

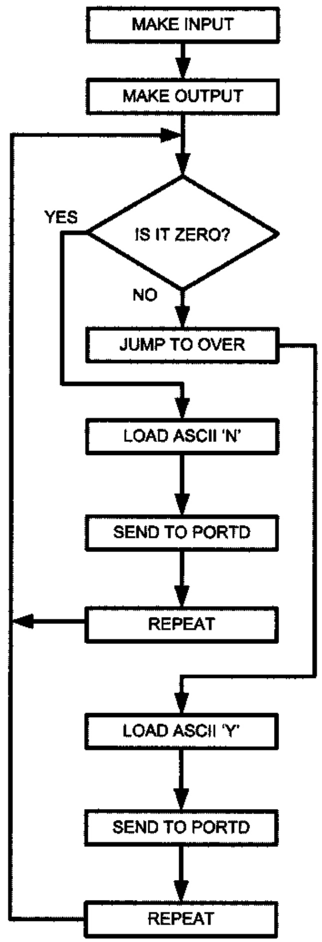
```
.INCLUDE "M32DEF.INC"

        CBI     DDRB,2              ;make PB2 an input
        LDI     R16,0xFF
        OUT     DDRD,R16            ;make PORTD an output port
AGAIN:  SBIC    PINB,2              ;skip next if PB bit is LOW
        RJMP    OVER                ;SW is HIGH
        LDI     R16, 'N'            ;R16 = 'N' (ASCII letter N)
        OUT     PORTD, R16          ;PORTD = 'N'
        RJMP    AGAIN

OVER:   LDI     R16,'Y'             ;R16 = 'Y' (ASCII letter Y)
        OUT     PORTD,R16           ;PORTD = 'Y'
        RJMP    AGAIN
```

AVR I/O PORT PROGRAMMING

4.2: I/O BIT MANIPULATION PROGRAMMING



INSTRUCTIONS

```
        CBI     DDRB, 2

        LDI     R16, 0XFF
        OUT     DDRD, R16

AGAIN:  SBIC    PINB, 2

        RJMP    OVER

        LDI     R16, 'N'

        OUT     PORTD, R16

        RJMP  AGAIN

OVER:   LDI     R16, 'Y'

        OUT     PORTD, R16

        RJMP    AGAIN
```

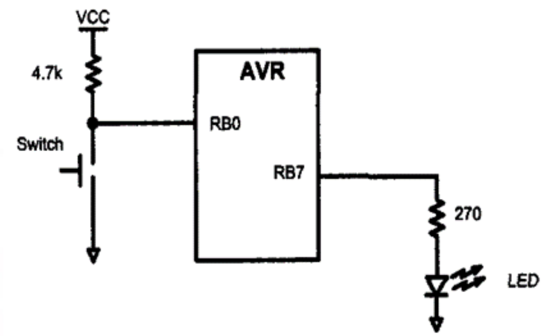
AVR I/O PORT PROGRAMMING
4.2: I/O BIT MANIPULATION PROGRAMMING

Reading a single bit

We can also use the bit test instructions to read the status of a single bit and send it to another bit or save it. This is shown in Examples 4-8 and **4-9.**

Example 4-8

A switch is connected to pin PB0 and an LED to pin PB7. Write a program to get the status of SW and send it to the LED.



Solution:

```
.INCLUDE "M32DEF.INC"

        CBI     DDRB,0              ;make
        SBI     DDRB,7              ;make
AGAIN:  SBIC    PINB,0              ;skip next if PB0 is clear
        RJMP    OVER
        CBI     PORTB,7
        RJMP    AGAIN
OVER:   SBI     PORTB,7
        RJMP    AGAIN
```