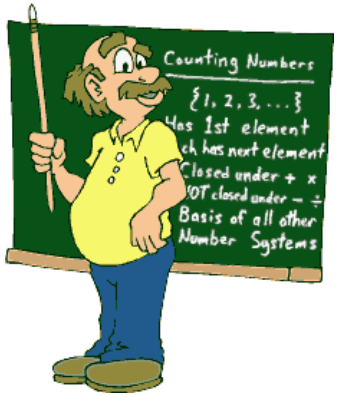
IECA

Embedded Computer Architecture

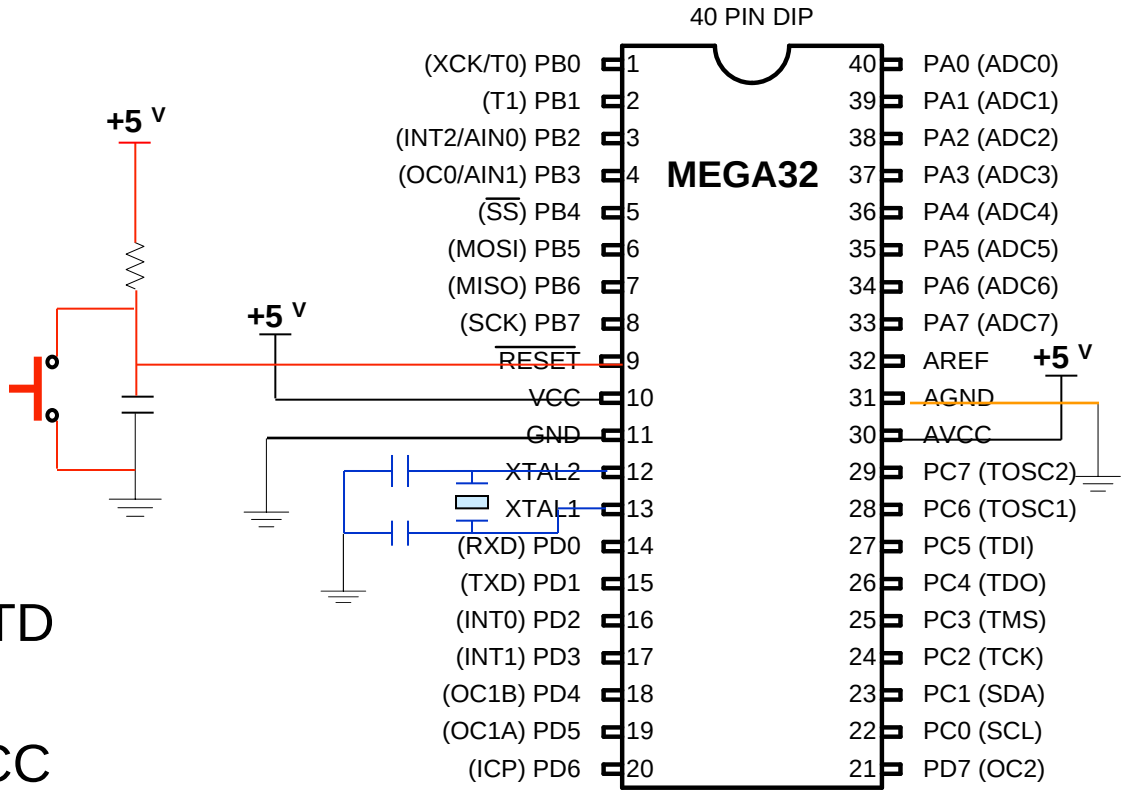
Lesson 8: Digital Ports



Version: 1-11-2015, Henning Hargaard

Mega32 Pinout

1. Important pins:
   1. Power
      - VCC
      - Ground
   2. Clock
      - XTAL1
      - XTAL2
   3. Reset
2. Digital I/O
   - PORTA, PORTB, PORTC, and PORTD
3. ADC pins
   - AREF, AGND, AVCC



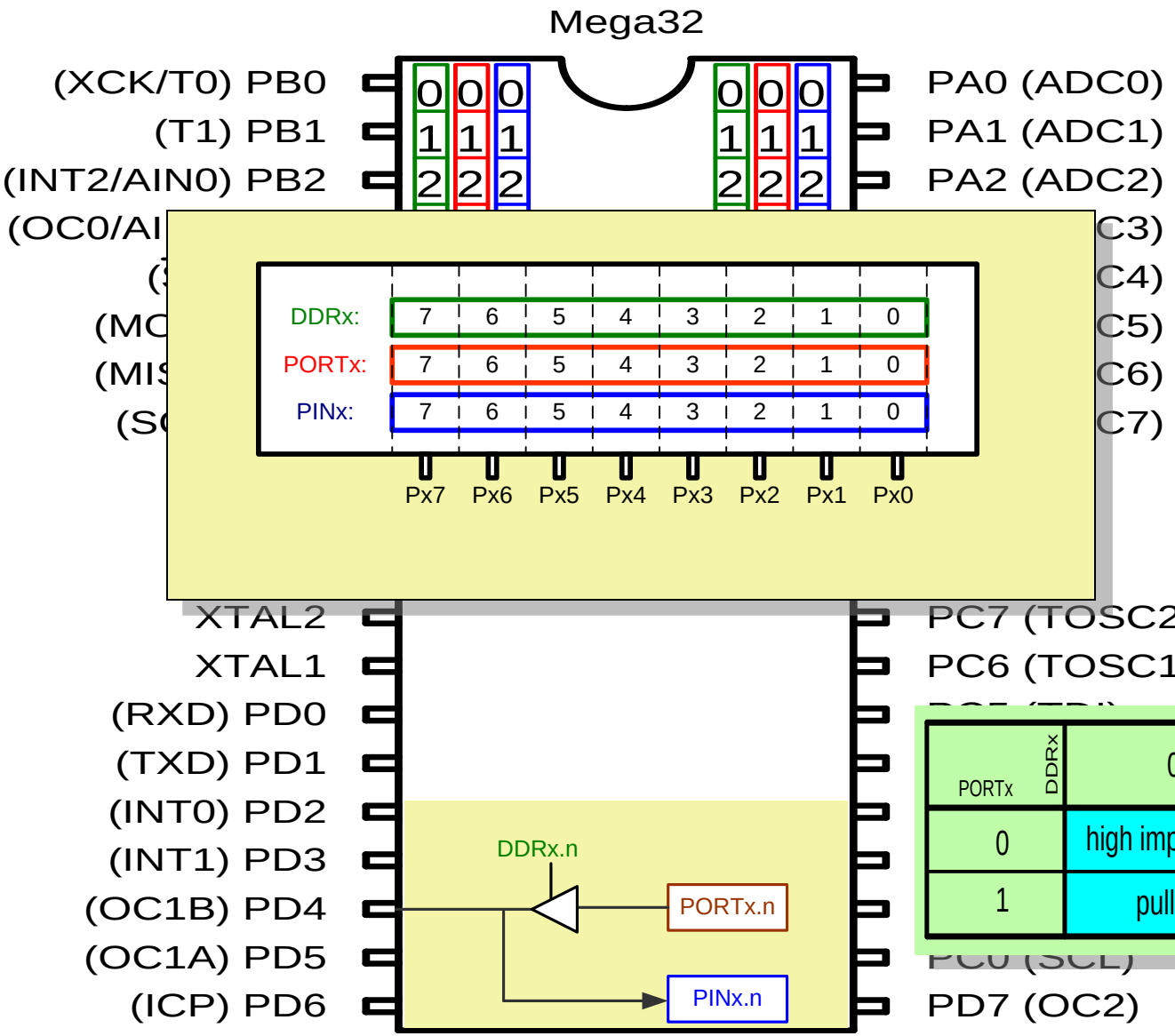
Various AVR controller Ports

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.

Digital ports and registers



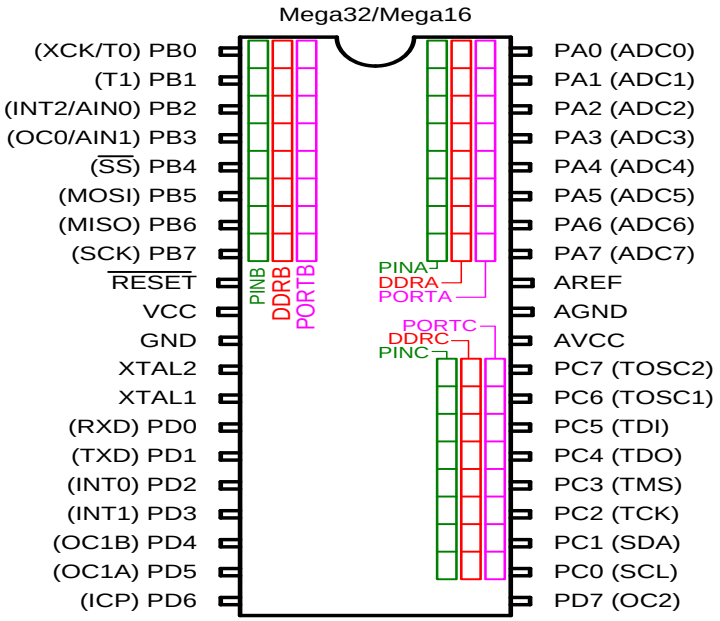
PORTx \ DDRx	0	1
0	high impedance	Out 0
1	pull-up	Out 1

Example 1

- Write a program that makes all the pins of PORTA one.

DDRA:	1	1	1	1	1	1	1	1
PORTA:	1	1	1	1	1	1	1	1

```
.INCLUDE "M32DEF.INC"
LDI   R20,0xFF   ;R20 = 11111111 (binary)
OUT   PORTA,R20  ;PORTA = R20
OUT   DDRA,R20   ;DDRA = R20
```



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

Example 2

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

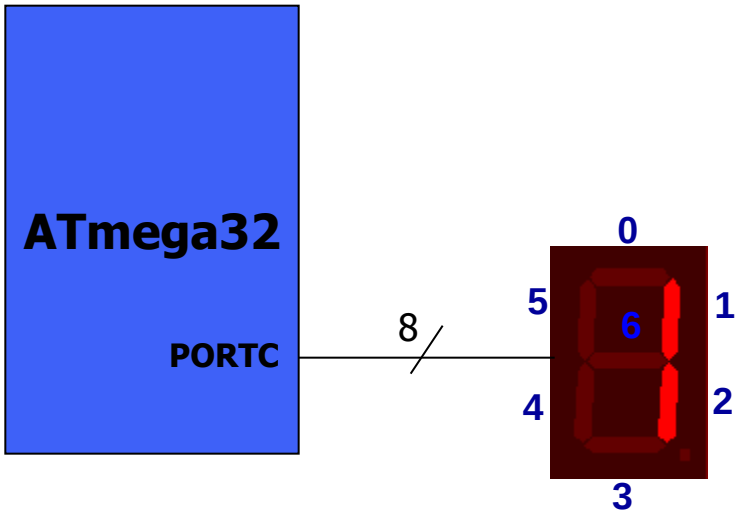
```
    LDI   R16,0xFF    ;R16 = 0xFF = 0b11111111
    OUT   DDRB,R16    ;make Port B an output port (1111 1111)
L1: LDI   R16,0x55    ;R16 = 0x55 = 0b01010101
    OUT   PORTB,R16   ;put 0x55 on port B pins
    CALL   DELAY
    LDI   R16,0xAA    ;R16 = 0xAA = 0b10101010
    OUT   PORTB,R16   ;put 0xAA on port B pins
    CALL DELAY
    RJMP L1
```

Example 3

- A 7-segment is connected to PORTA. Display 1 on the 7-segment.

DDRC	1	1	1	1	1	1	1	1
PORTC:	0	0	0	0	0	1	1	0

```
     .INCLUDE "M32DEF.INC"
     LDI   R20,0x06   ;R20 = 00000110 (binary)
     OUT   PORTC,R20  ;PORTC = R20
     LDI   R20,0xFF   ;R20 = 11111111 (binary)
     OUT   DDRC,R20   ;DDRC = R20
L1:  RJMP  L1
```



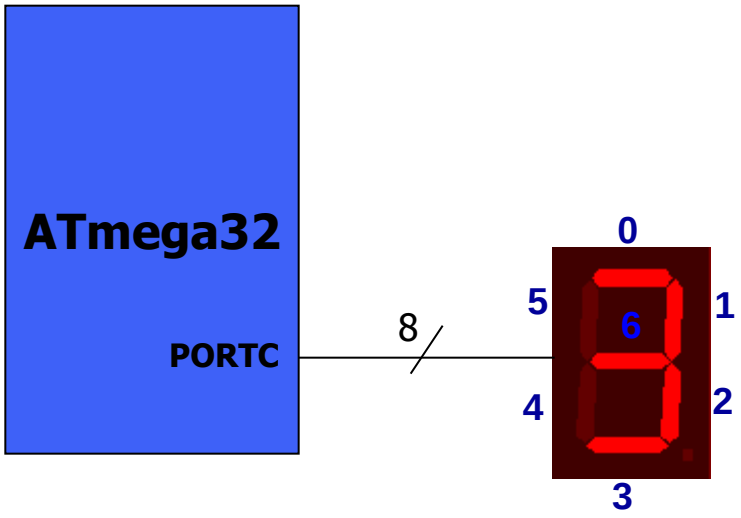
PORTx \ DDRx	0	1
0	high impedance	Out 0
1	pull-up	Out 1

Example 4

- A 7-segment is connected to PORTA. Display 3 on the 7-segment.

DDR:	1	1	1	1	1	1	1	1
PORTC:	0	1	0	0	1	1	1	1

```
     .INCLUDE "M32DEF.INC"
     LDI   R20,0x4F   ;R20 = 01001111 (binary)
     OUT   PORTC,R20  ;PORTC = R20
     LDI   R20,0xFF   ;R20 = 11111111 (binary)
     OUT   DDRC,R20   ;DDRC = R20
L1:  RJMP  L1
```

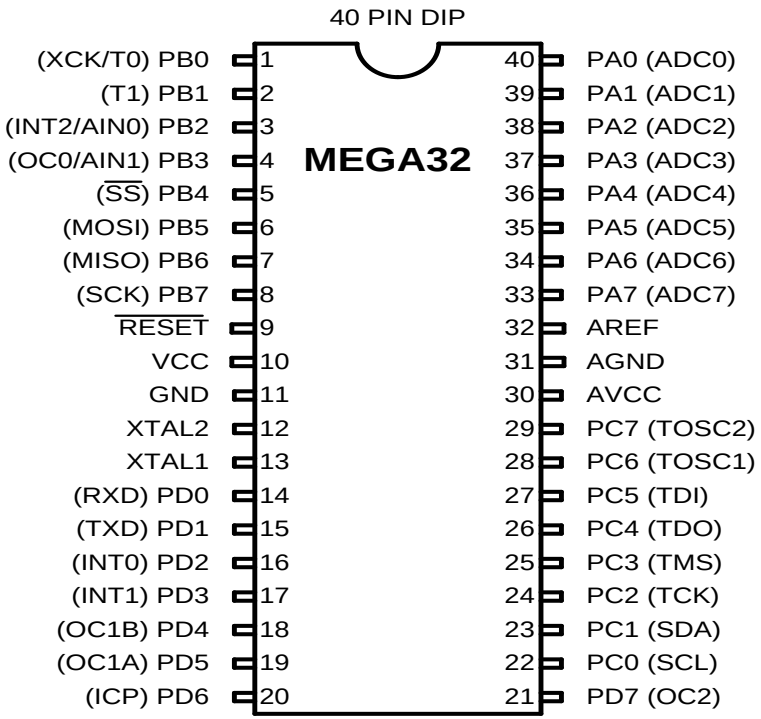


PORTx \ DDRx	0	1
0	high impedance	Out 0
1	pull-up	Out 1

Input example

- Write a program, that reads from PA and writes the data to PB.

```
    .INCLUDE "M32DEF.INC"
    LDI   R20,0x0    ;R20 = 00000000 (binary)
    OUT   DDRA,R20   ;DDRA = R20
    LDI   R20,0xFF   ;R20 = 11111111 (binary)
    OUT   DDRB,R20   ;DDRB = R20
L1: IN    R20,PINA   ;R20 = PINA
    OUT   PORTB,R20  ;PORTB = R20
    RJMP L1
```



PORTx / DDRx	0	1
0	high impedance	Out 0
1	pull-up	Out 1

Single-Bit I/O instructions

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)

I/O registers with options for ”single bit”

Address		Name
Mem.	I/O	
$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		Name
Mem.	I/O	
$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		Name
Mem.	I/O	
$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

SBI and CBI instructions

- SBI (Set Bit in IO register)
  - SBI ioReg, bit ;ioReg.bit = 1
  - Examples:
    - SBI PORTD,0 ;PORTD.0 = 1
    - SBI DDRC,5 ;DDRC.5 = 1
- CBI (Clear Bit in IO register)
  - CBI ioReg, bit ;ioReg.bit = 0
  - Examples:
    - CBI PORTD,0 ;PORTD.0 = 0
    - CBI DDRC,5 ;DDRC.5 = 0

Example

- Write a program that continuosly toggles PORTA.4.

```
     .INCLUDE "M32DEF.INC"
     SBI  DDRA,4
L1:  SBI  PORTA,4
     CBI  PORTA,4
     RJMP L1
```

Example

- LEDs are connected to the PORTD pins.
  Write a program that alternately turns ON each LED from D0 to D7. Call a DELAY each time a new LED lights up.

```
.INCLUDE "M32DEF.INC"
        LDI     R20, 0xFF
        OUT     DDRD, 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
```

SBIC and SBIS

- SBIC (Skip if Bit in IO register Cleared)
  - SBIC ioReg, bit ; if (ioReg.bit = 0) skip next instruction
  - Example:

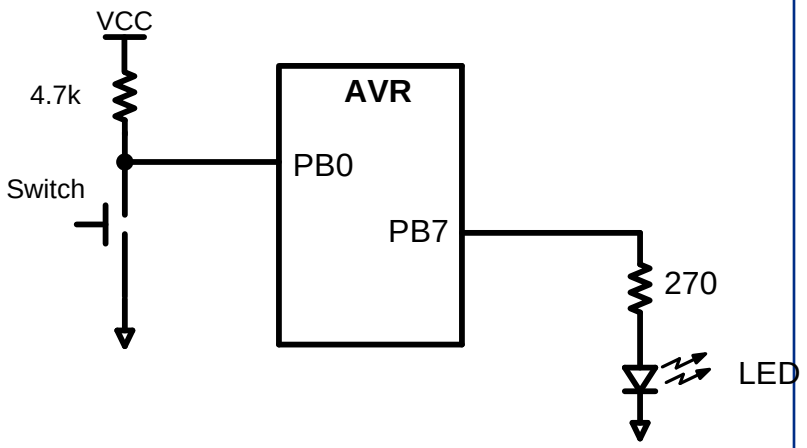
```
SBIC   PORTD,0   ;skip next instruction if PORTD.0=0
INC    R20
LDI    R19,0x23
```

- SBIS (Skip if Bit in IO register Set)
  - SBIS ioReg, bit ; if (ioReg.bit = 1) skip next instruction
  - Example:

```
SBIS   PORTD,0   ;skip next instruction if PORTD.0=1
INC    R20
LDI    R19,0x23
```

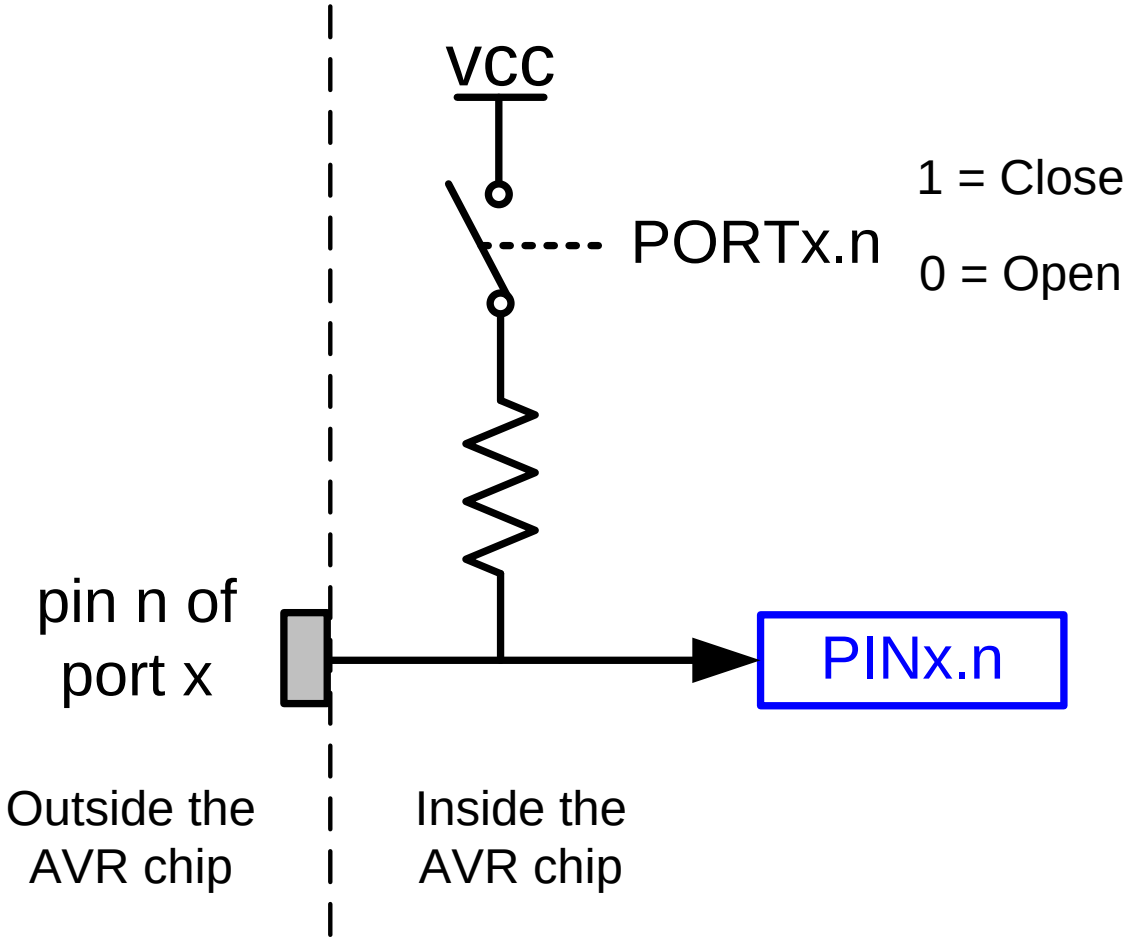
Example

- A switch is connected to PB pin 0 and a LED to PB pin 7. Write a program that turns OFF the LED, if the switch is pressed. Otherwise the LED should be ON.

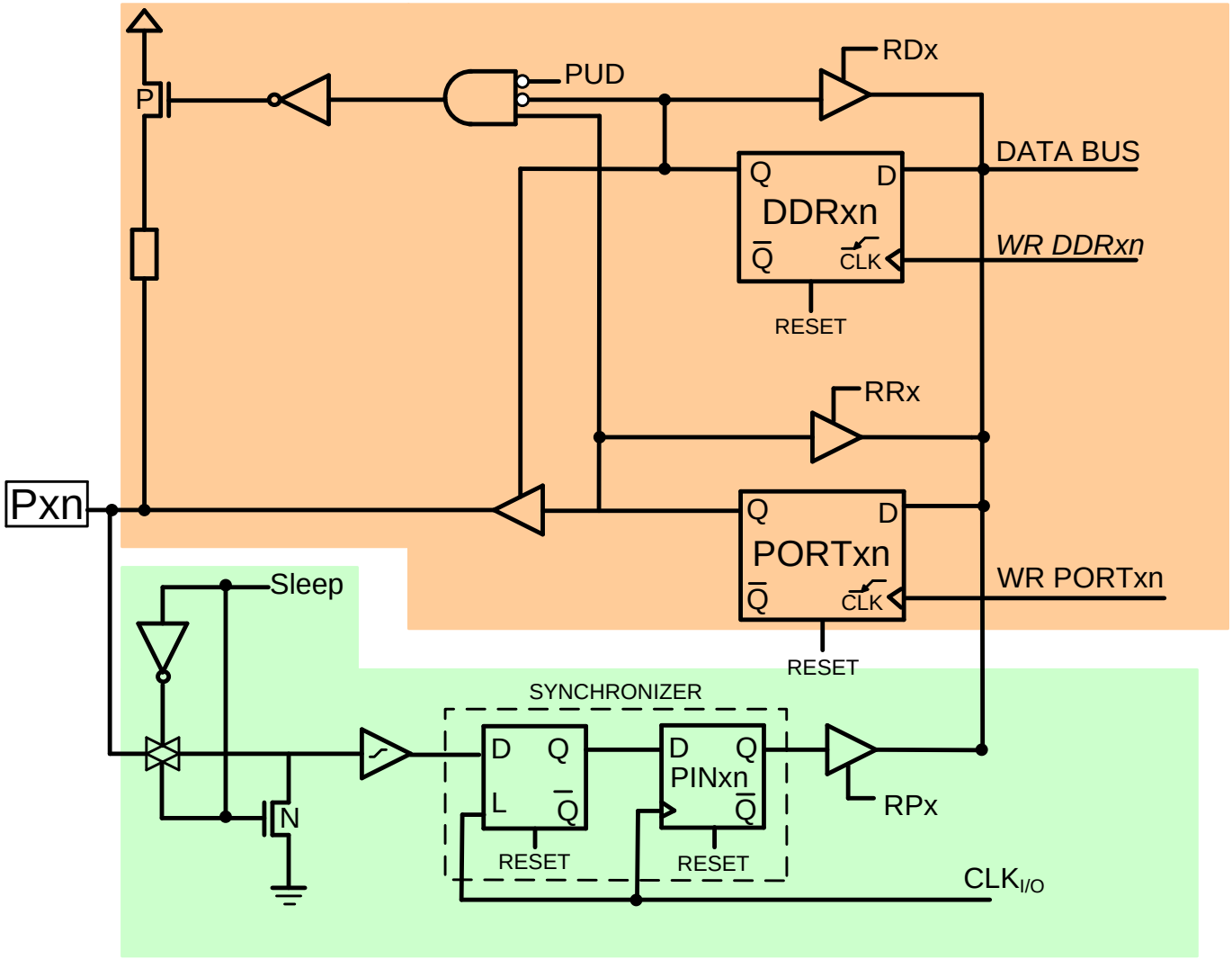



```
.INCLUDE "M32DEF.INC"
       CBI  DDRB,0          ;make PB0 an input
       SBI  DDRB,7          ;make PB7 an output
AGAIN: SBIC PINB,0          ;skip next if PB0 is clear
       RJMP OVER            ;(JMP is OK too)
       CBI  PORTB,7
       RJMP AGAIN           ;we can use JMP too
OVER:  SBI  PORTB,7
       RJMP AGAIN           ;we can use JMP too
```

Pull-up resistor



The structure of I/O pins



Alternate use of port pins

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

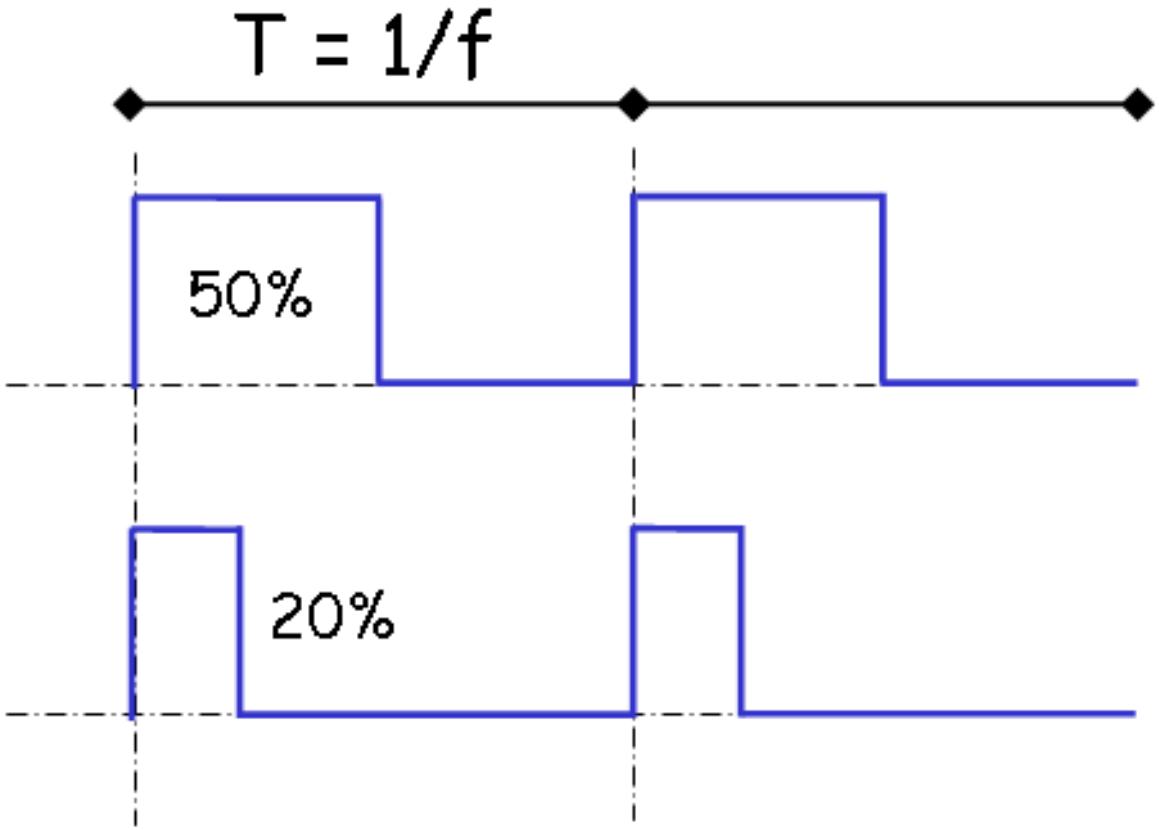
Table 4-5: Port C Alternate Functions

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

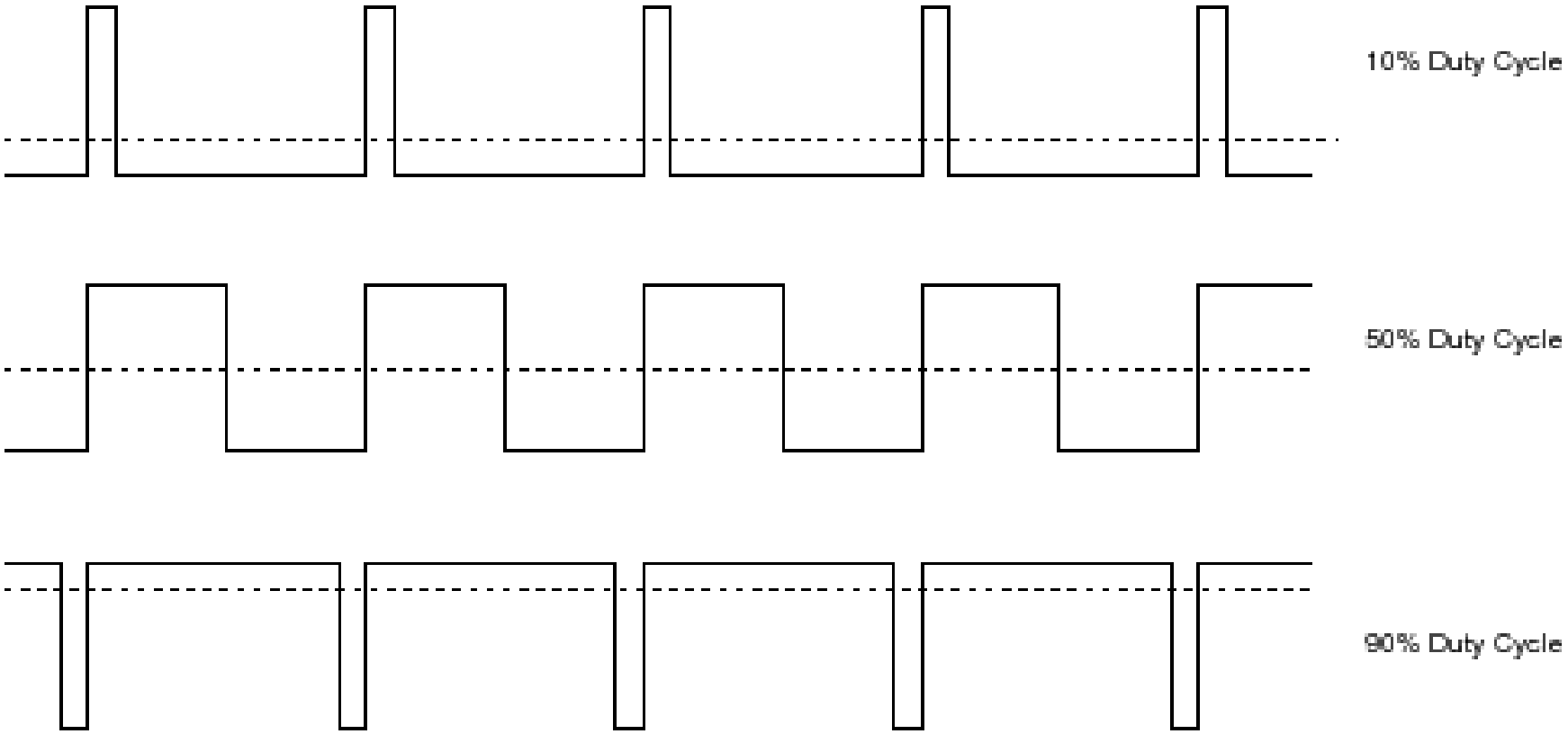
Bit	Function
PD0	PSP0/C1IN+
PD1	PSP1/C1IN-
PD2	PSP2/C2IN+
PD3	PSP3/C2IN-
PD4	PSP4/ECCP1/P1A
PD5	PSP5/P1B
PD6	PSP6/P1C
PD7	PSP7/P1D

PWM = Pulse Width Modulation



PWM: The goal is to control the ”duty cycle” (frequency is less important)

Various ”duty cycles”



Average DC

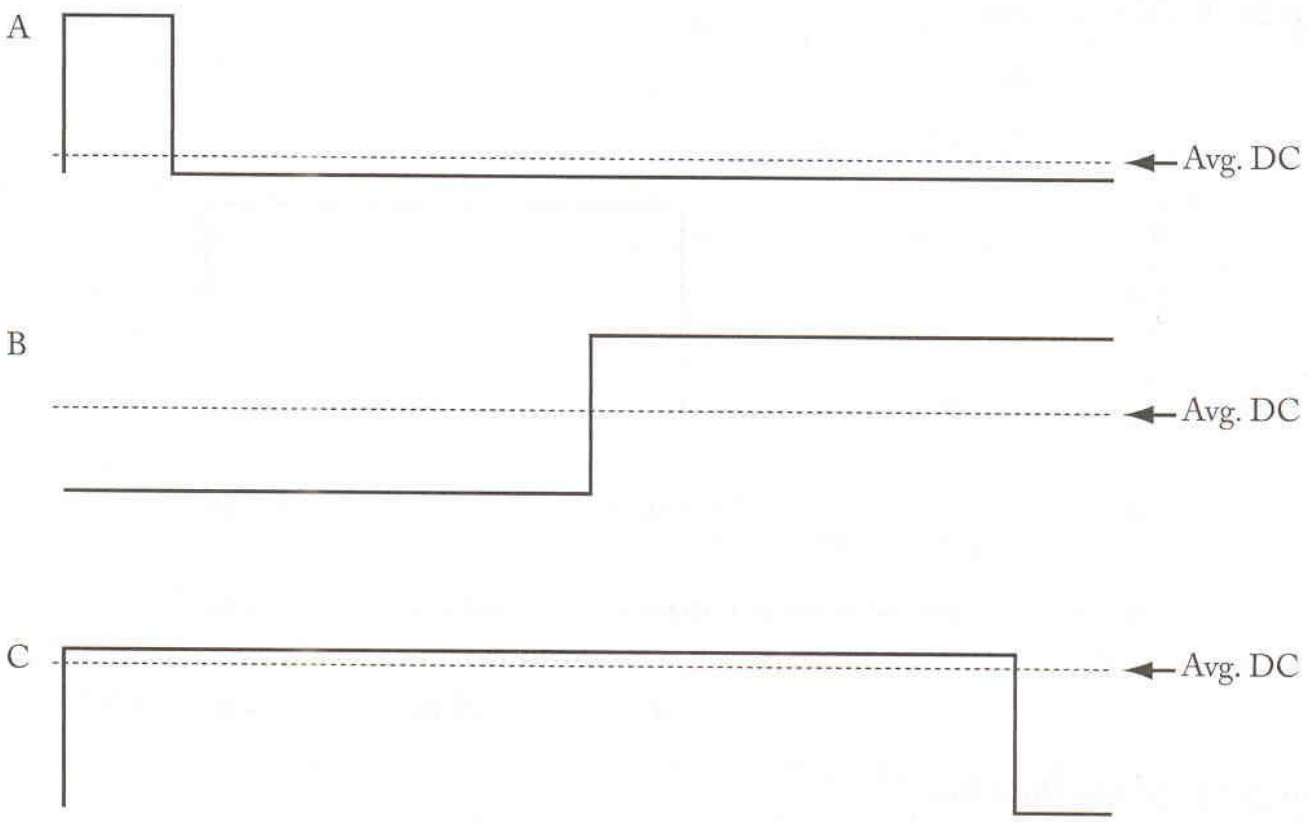
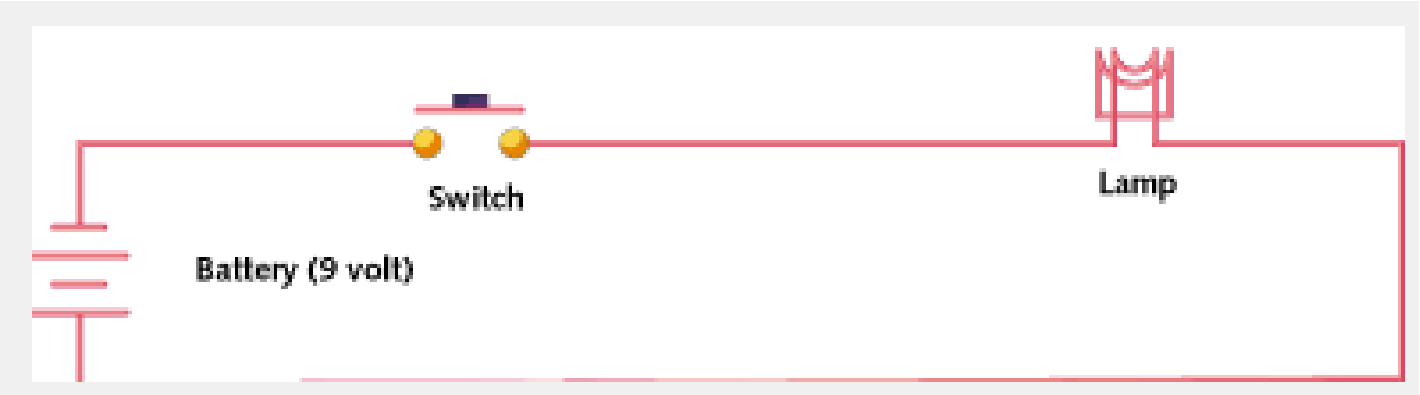
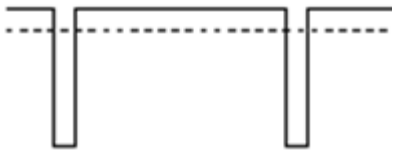


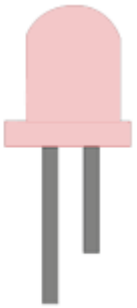
Figure 2–27 *PWM Waveforms*

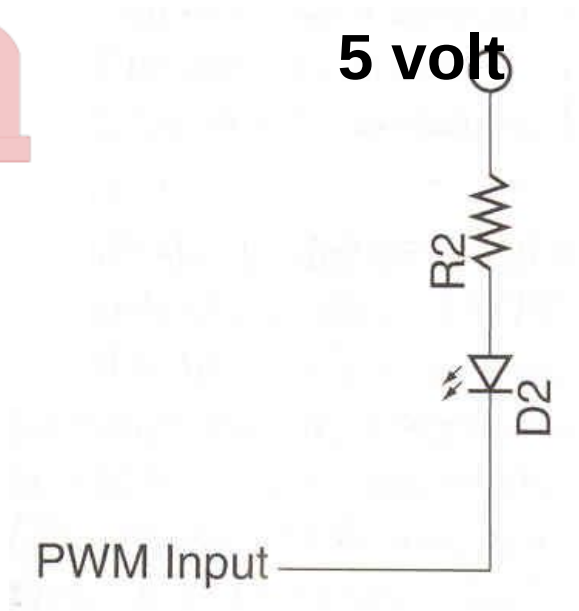
PWM : Controlling light intensity



Duty Cycle: 0%







PWM example

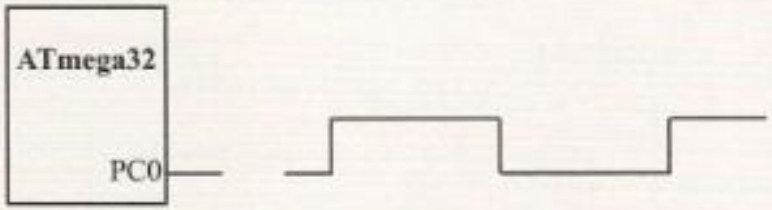
Example 4-3

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 PC0 with a time delay between each state.

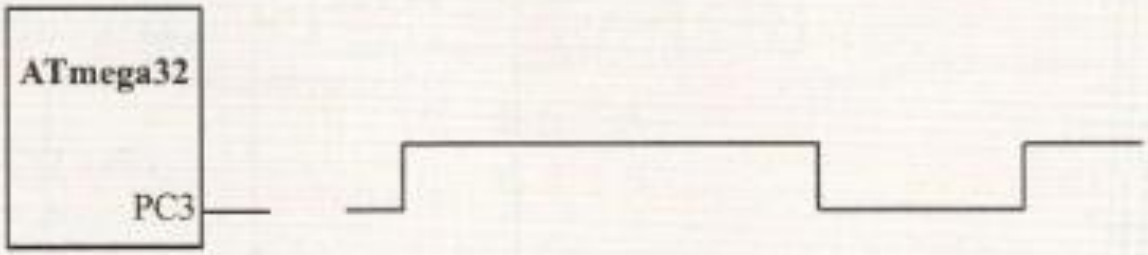
```
.INCLUDE "M32DEF.INC"

      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 (PC0 = out)
HERE: SBI   PORTC, 0    ;set to HIGH PC0 (PC0 = 1)
      CALL  DELAY       ;call the delay subroutine
      CBI   PORTC, 0    ;PC0 = 0
      CALL  DELAY
      RJMP  HERE        ;keep doing it
```



PWM example (..continued..)

(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
        RJMP   HERE         ;keep doing it
```



CPI - (Compare with Immediate)

Description:

This instruction performs a compare between register Rd and a constant. The register is not changed. All conditional branches can be used after this instruction.

Operation:

(i) Rd - K

	Syntax:	Operands:	Program Counter:
(i)	CPI Rd,K	16 ≤ d ≤ 31, 0≤ K ≤ 255	PC ← PC + 1

16-bit Opcode:

0011	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

Example:

```
        cpi     r19,3     ; Compare r19 with 3
        brne    error     ; Branch if r19<>3
        ...
error:  nop               ; Branch destination (do nothing)
```

End of lesson 8