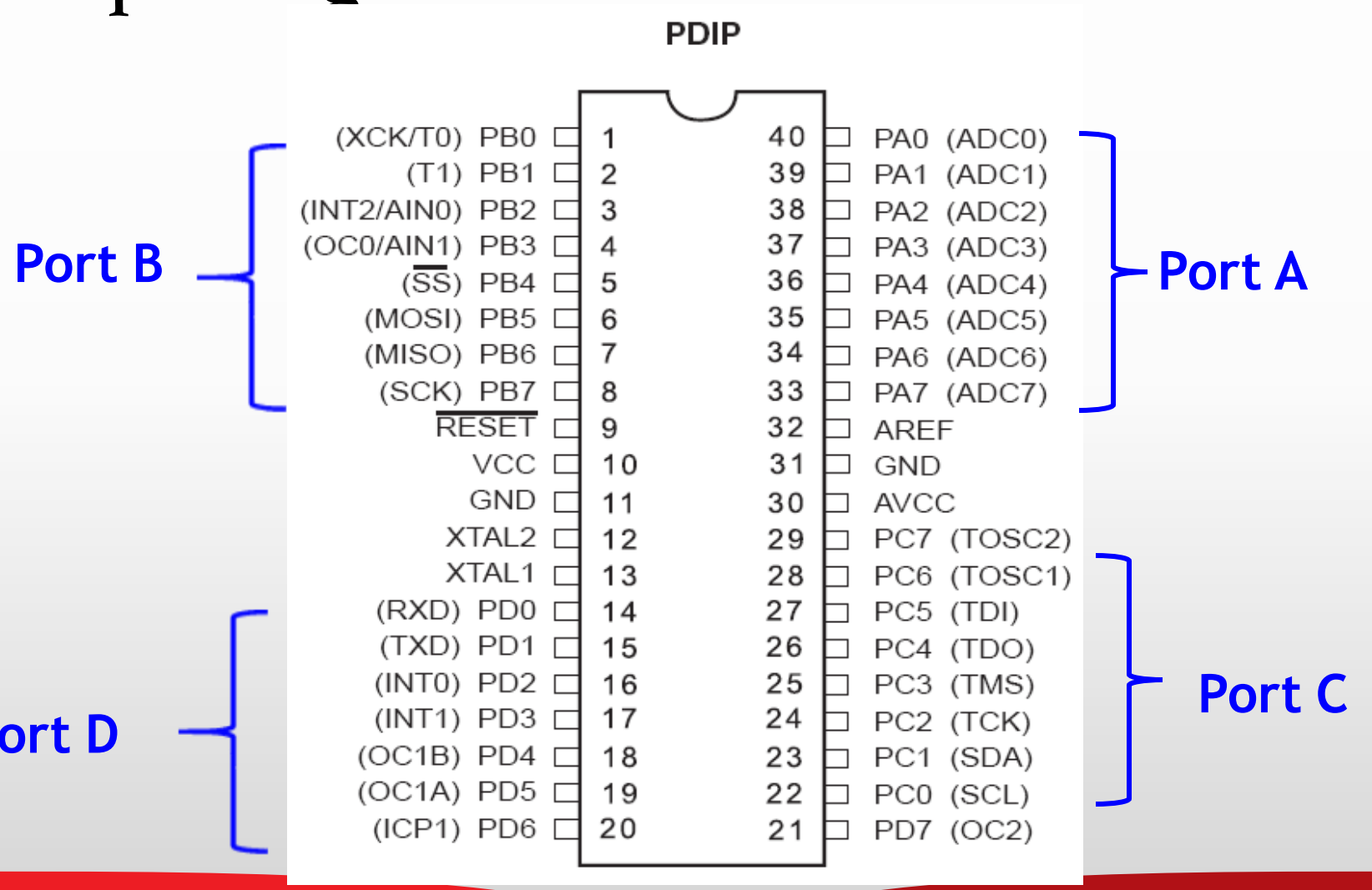
INSTRUKSI

TTH2D3

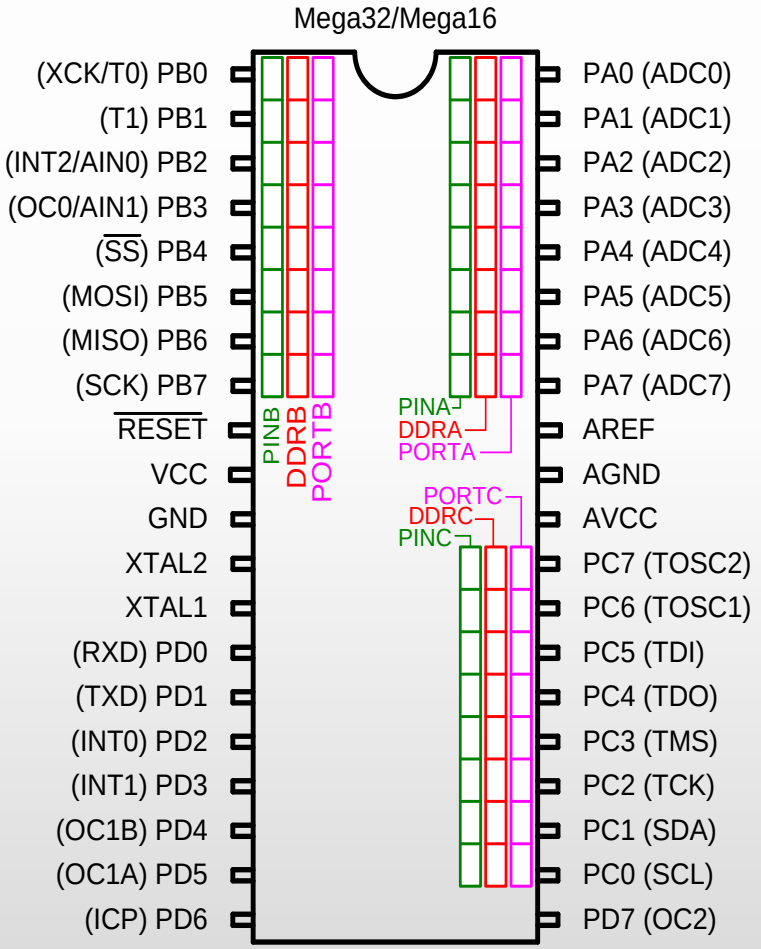
Mikroprosesor

AT Mega 32 pin diagram



ATMega32 Pin out & Descriptions

- There are 4 ports that provide parallel I/O interfaces to outside world: **Port A, Port B, Port C & Port D.**
  - Each port provides 8 **bidirectional** digital I/O lines which are connected to ATmega32 pins provided that *alternate functions* are not selected on that port.
  - Eventhough **bidirectional**, at any time the I/O line can either be **Input or Output**.
  - The **Directions** of each I/O lines must be **configured** (input or output) before they are used.
  - Naming convention:
    - PORTx ≡ I/O reg for Port x where x = A, B, C, D.
    - PORTxn ≡ Port x bit n where $n = 0 - 7$.

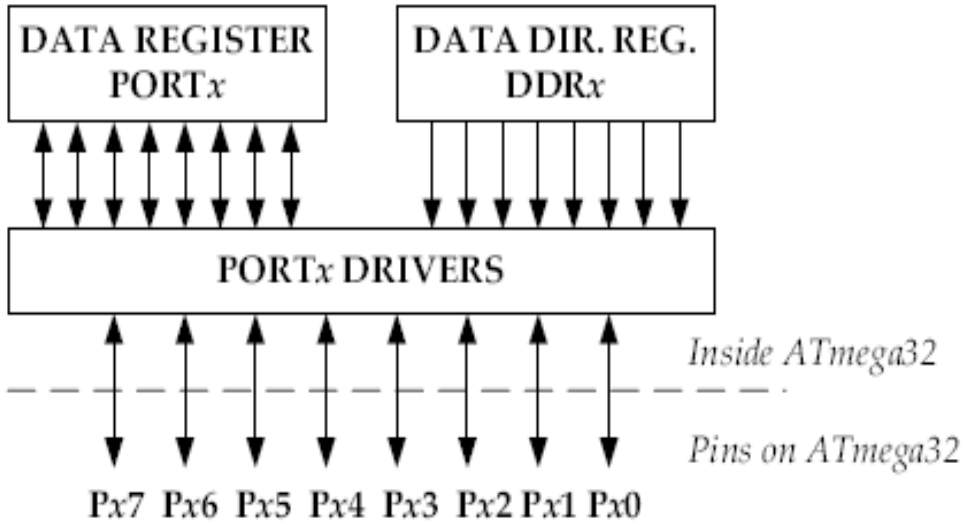


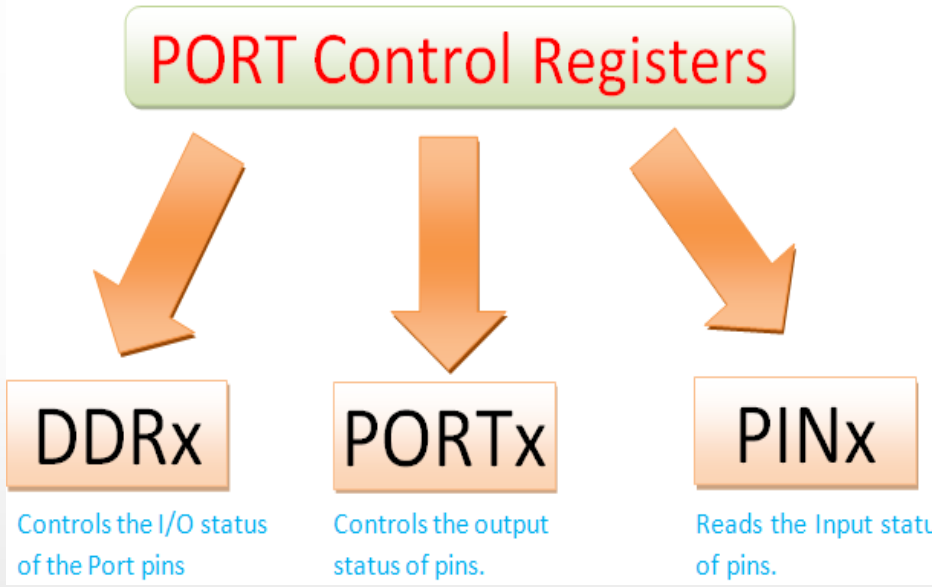
Port description

- Port A (PA0-PA7) – serves as an 8-bit **bi-directional** digital I/O port. Port A can be programmed to serve as **alternate function** as analog input for the ADC.
- Port B (PB0-PB7) – serves as an 8-bit **bi-directional** digital I/O port. with optional internal pull-ups. Port B pins are tri-stated when reset. Port B can be programmed to serve as **alternate function**:.
- Port C (PC0-PC7) – serves as an 8-bit **bi-directional** digital I/O port. with optional internal pull-ups. Port C pins are tri-stated when reset. Port C can be programmed to serve as **alternate function**:.
- Port D (PD0-PD7) – serves as an 8-bit **bi-directional** digital I/O port. with optional internal pull-ups. Port D pins are tri-stated when reset. Port D can be programmed to serve as **alternate function**:.

ATMega32 Pin out & Descriptions

- The general programmer view of Port A, B, C & D :





- Each Port *x* has three 8-bit **Registers** associated with it.

  Register ≈ a memory :

  - **DDR***x* – *Data Direction Register* for Port *x* (Read/Write).
  - **PORT***x* –*Data Register* for Port *x* (Read/Write).
  - **PIN***x* – *Port Input Pins Register* for Port *x* (Read only).

Single pin accessing

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

ASSEMBLY

Instruksi Dasar

- LDI (Load Immediate) : menuliskan konstanta ke register, sebelum konstanta tersebut dikeluarkan ke port I/O.

Contoh : LDI R16,0xFF

- OUT : menuliskan data register ke port I/O.

Contoh : OUT DDRA,R16

- IN : menuliskan data port I/O ke dalam register.

Contoh : IN R16, PORTA

- SBI (Set bit in I/O) : membuat logika high pada sebuah bit port I/O.

Contoh : SBI PORTA,0 (bit ke-0 port A diberi logika high / set)

Instruksi Dasar

- CBI (Clear bit in I/O) : membuat logika low pada sebuah bit port I/O.

Contoh : CBI PORTA,1 (bit ke-1 port A diberi logika low /clear)

- SBIS (Skip if bit in I/O is set) : lompati satu instruksi di bawahnya jika bit port I/O dalam kondisi high

Contoh : SBIS PORTA,2

RJMP LAGI

Instruksi "RJMP LAGI" akan dilompati jika bit 2 port A diberi logika *high*

Instruksi Dasar

- **SBIC** (*Skip if bit in I/O is cleared*) : lompati satu instruksi di bawahnya jika bit port I/O dalam kondisi *low*

Contoh : SBIC PORTA,2

RJMP LAGI

Instruksi Aritmatika

No	Instruksi	Operand	Deskripsi	Operasi	Flags	Clock
1.	ADD	Rd, Rr	Menambahkan 2 register	Rd ←Rd+Rr	Z,C,N,V,H	1
2.	ADC	Rd, Rr	Menambahkan 2 register+carry flagnya	Rd ← Rd+Rr+C	Z,C,N,V,H	1
3.	ADIW	Rdl, K	Add Immediate to Word	Rdh:Rdl← Rdh:Rdl+K	Z,C,N,V,S	2
4.	SUB	Rd, Rr	Mengurangi 2 register	Rd ←Rd-Rr	Z,C,N,V,H	1
5.	SUBI	Rd, K	Subtract constant from register	Rd ←Rd-K	Z,C,N,V,H	1
6.	SBC	Rd, Rr	Subtract with Carry 2 registers	Rd ←Rd-Rr-C	Z,C,N,V,H	1

Instruksi Aritmatika

No	Instruksi	Operand	Deskripsi	Operasi	Flags	Clock
7.	SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ←Rd-Rr-C	Z,C,N,V,H	1
8.	SBIW	Rdl, K	Subtract Immediate from Word	Rdh:Rdl← Rdh:Rdl-K	Z,C,N,V,S	2
9.	AND	Rd, Rr	Logical AND Reg.	Rd ←Rd • Rr	Z,N,V	1
10.	ANDI	Rd, K	Logical AND Regist+Constant	Rd ←Rd • K	Z,N,V	1
11.	OR	Rd, Rr	Logical OR Reg.	Rd ←Rd V Rr	Z,N,V	1
12.	ORI	Rd, K	Logical OR Regist+Constant	Rd ←Rd V K	Z,N,V	1

Instruksi Aritmatika

No	Instruksi	Operand	Deskripsi	Operasi	Flags	Clock
13.	EOR	Rd, Rr	Exclusive OR Registers	Rd ←Rd ⊕ K	Z,N,V	1
14.	COM	Rd	One's Complement	Rd ←0xFF-Rd	Z,C,N,V	1
15.	NEG	Rd	Two's Complement	Rd ←0x00-Rd	Z,C,N,V,H	1
16.	SBR	Rd, K	Set Bit(s) in Reg.	Rd ←Rd V K	Z,N,V	1
17.	CBR	Rd, K	Cleaar Bit(s) in Register	Rd ←Rd • (0XFF - K)	Z,N,V	1
18.	CLR	Rd	Clear Register	Rd ←Rd⊕Rd	Z,N,V	1
19.	SER	Rd	Set register	Rd ←0xFF	None	1
20.	INC	Rd	Increment	Rd ←Rd+1	Z,N,V	1

Instruksi Aritmatika

No	Instruksi	Operand	Deskripsi	Operasi	Flags	Clock
21.	DEC	Rd	Decrement	Rd ←Rd-1	Z,N,V	1
22.	MUL	Rd, Rr	Multiply Unsigned	R1:R0 ←RdxRr	Z,C	2
23.	MULS	Rd, Rr	Multiply Signed	R1:R0 ←RdxRr	Z,C	2
24.	MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ←RdxRr	Z,C	2
25.	FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (RdxRr)<<1	Z,C	2
26.	FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (RdxRr)<<1	Z,C	2
27.	FMULSU	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (RdxRr)<<1	Z,C	2

Instruksi Percabangan

1.	RJMP	k	Relative Jump	PC←PC+k+1	None	2
2.	IJMP	K	Indirect Jumpt to (Z)	PC←Z	None	2
3.	RCALL	k	Relative Subroutine Call	PC←PC+k+1	None	3
4.	ICALL		Indirect Call to (Z)	PC←Z	None	3
5.	RET		Subroutine Return	PC←Stack	None	4
6.	RETI		Interrupt Return	PC←Stack	I	4
7.	CP	Rd, Rr	Compare	Rd - Rr	Z,N,V,C, H	1
8.	CPI	Rd, K	Compare Register with immediate	Rd-K		
9.	CPSE	Rd, Rr	Compare, Skip if Equal	If (Rd=Rr) PC←PC+2 or 3	None	1/2/3
10.	CPC	Rd, Rr	Compare with Carry	Rd – Rr - C	Z,N,V,C, H	1

Instruksi Percabangan

11.	SBIC	P, b	Skip if bit in I/O register is Cleared	If (P(b)=0) PC←PC+2 or 3	None	1/2/3
12.	SBIS		Skip if bit in I/O register is Set	If (P(b)=1) PC←PC+2 or 3	None	1/2/3
13.	SBRC	Rr, b	Skip if bit in register is Cleared	If (P(b)=0) PC←PC+2 or 3	None	1/2/3
14.	SBRS	Rr, b	Skip if bit in register is Set	If (P(b)=1) PC←PC+2 or 3	None	1/2/3
15.	BRBS	S,k	Branch if Status Flag Set	If (SREG(s)=1) then PC←PC+k+1	None	1/2/3
16.	BRBC	S, k	Branch if Status Flag Cleared	If (SREG(s)=0) then PC←PC+k+1	None	1/2
17.	BREQ	k	Branch if Equal	If (Z=1) then PC←PC+k+1	None	1/2

Instruksi Percabangan

18.	BRNE	k	Branch if Not Equal	If (Z=0) then PC←PC+k+1	None	1/2
19.	BRCS	k	Branch if Carry Set	If (C=1) then PC←PC+k+1	None	1/2
20.	BRCC	k	Branch if Carry Cleared	If (C=0) then PC←PC+k+1	None	1/2
21.	BRSH	k	Branch if Same or Higher	If (C=0) then PC←PC+k+1	None	1/2
22.	BRLO	k	Branch if Lower	If (C=1) then PC←PC+k+1	None	1/2
23.	BRMI	k	Branch if Minus	If (N=1) then PC←PC+k+1	None	1/2
24.	BRPL	K	Branch if Plus	If (N=0) then PC←PC+k+1	None	1/2
25.	BRHS	K	Branch if Half Carry Flag Set	If (H=1) then PC←PC+k+1	None	1/2
26.	BRHC	k	Branch if Half Carry Flag Cleared	If (H=0) then PC←PC+k+1	None	1/2

Instruksi Percabangan

27.	BRGE	k	Branch if Greater or Equal	Signed if (N⊕V=0) then PC←PC+k+1	None	1/2
28.	BRLT	k	Branch if Less than Zero	Signed if (N⊕V=1) then PC←PC+k+1	None	1/2
29.	BRTS	k	Branch if T flag Set	If (T=1) then PC←PC+k+1	None	1/2
30.	BRTC	k	Branch if T flag Cleared	If (T=0) then PC←PC+k+1	None	1/2
31.	BRVS	k	Branch if Overflow flag is Set	If (V=1) then PC←PC+k+1	None	1/2
32.	BRVC	k	Branch if Overflow flag is Cleared	If (V=0) then PC←PC+k+1	None	1/2
33.	BRIE	k	Branch if Interrupt Enabled	If (I=1) then PC←PC+k+1	None	1/2
34.	BRID	k	Branch if Interrupt Dissabled	If (I=0) then PC←PC+k+1	None	1/2

Instruksi Transfer Data

1.	IN	Rd, P	In Port	Rd ←P	None	1
2.	OUT	P, Rr	Out Port	P←Rr	None	1
3.	MOV	Rd, Rr	Move Between Registers	Rd←Rr	None	1
4.	MOVW	Rd, Rr	Copy register Word	Rd+1:Rd ←Rr+1:Rr	None	1
5.	LDI	Rd. k	Load Immediate	Rd←k	None	1
6.	LD	Rd, X	Load Indirect	Rd←(X)	None	2
7.	LD	Rd, X+	Load Indirect and Post-Inc.	Rd←(X), X←X+1	None	2
8.	LD	Rd, -X	Load Indirect and Pre-Dec.	X←X-1, Rd←(X)	None	2
9.	LD	Rd, Y	Load Indirect	Rd←(Y)	None	2
10.	LD	Rd, Y+	Load Indirect and Post-Inc.	Rd←(Y), Y←Y+1	None	2
11.	LD	Rd, -Y	Load Indirect and Pre-Dec.	Y←Y-1, Rd←(Y)	None	2

Instruksi Transfer Data

12.	LDD	Rd, Y+q	Load Indirect with Displacement	Rd←(Y+q)	None	2
13.	LD	Rd, Z+	Load Indirect and Post-Inc.	Rd←(Z), Z←Z+1	None	2
14.	LD	Rd, -Z	Load Indirect and Pre-Dec.	Z←Z-1, Rd←(Z)	None	2
15.	LD	Rd, Z	Load Indirect	Rd←(Z)	None	2
16.	LDD	Rd, Z+q	Load Indirect with Displacement	Rd←(Z+q)	None	2
17.	LDS	Rd, k	Load Direct from SRAM	Rd←(k)	None	2
18.	ST	X, Rr	Store Indirect	(X) ←Rr	None	2
19.	ST	X+,Rr	Store Indirect and Post-Inc.	(X)←Rr,X←X+1	None	2
20.	ST	-X,Rr	Store Indirect and Pre-Dec.	X←X-1,(X)←Rr	None	2

Instruksi Transfer Data

21.	ST	Y,Rr	Store Indirect	(Y) ←Rr	None	2
22.	ST	Y+,Rr	Store Indirect and Post-Inc.	(Y)←Rr,Y←Y+1	None	2
23.	ST	-Y,Rr	Store Indirect and Pre-Dec.	Y←Y-1,(Y)←Rr	None	2
24.	STD	Y+q, Rr	Store Indirect with Displacement	(Y+q) ←Rr	None	2
25.	ST	Z,Rr	Store Indirect	(Z) ←Rr	None	2
26.	ST	Z+,Rr	Store Indirect and Post-Inc.	(Z)←Rr,Z←Z+1	None	2
27.	ST	-Z,Rr	Store Indirect and Pre-Dec.	Z←Z-1,(Z)←Rr	None	2
28.	STD	Z+q, Rr	Store Indirect with Displacement	(Z+q) ←Rr	None	2
29.	STS	k, Rr	Store Direct to SRAM	(k) ←Rr	None	2

Instruksi Transfer Data

30.	PUSH	Rr	Push register on Stack	STACK←Rr	None	2
31.	POP	Rd	Pop Register from Stack	Rd←STACK	None	2
32.	LPM		Load Program Memory	R0←(Z)	None	3
33.	LPM	Rd, Z	Load Program Memory	Rd←(Z)	None	3
34.	LPM	Rd,Z+	Load Program Memory and Post-Inc	Rd←(Z),Z←Z+1	None	3
35.	SPM		Store Program Memory	(Z)←R1:R0	None	-

Instruksi Tes Bit

1.	SBI	P,b	Set Bit in I/O register I/O	(P,b)←1	None	2
2.	CBI	P,b	Clear Bit in I/O register I/O	(P,b)←0	None	2
3.	LSL	Rd	Logical Shift left	Rd(n+1)←Rd(n), Rd(0)←0	Z,C,N, V	1
4.	LSR	Rd	Logical Shift Right	Rd(n)←Rd(n+1), Rd(7)←0	Z,C,N, V	1
5.	ROL	Rd	Rotate Left Through Carry	Rd(0)←C, Rd(n+1)←Rd(n), C← Rd(7)	Z,C,N, V	1
6.	ROR	Rd	Rotate Right Through Carry	Rd(7)←C, Rd(n)←Rd(n+1), C← Rd(0)	Z,C,N, V	1
7.	ASR	Rd	Arithmetic Shift Right	Rd(n)←Rd(n+1), n=0…6	Z,C,N, V	1
8.	SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4) , Rd(7..4)←Rd(3..0)	None	1

Instruksi Kontrol MCU

No	Instruksi	Operand	Deskripsi	Operasi	Flags	Clock
1.	NOP		No Operation		None	1
2.	SLEEP		Sleep	(see specific descr. for sleep function)	None	1
3.	WDR		Watchdog Reset	(see specific descr. for WDR/Timer)	None	1
4.	BREAK		Break	For On-chip debug only	None	N/A

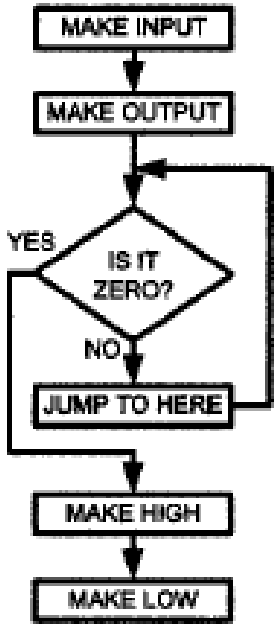
Contoh

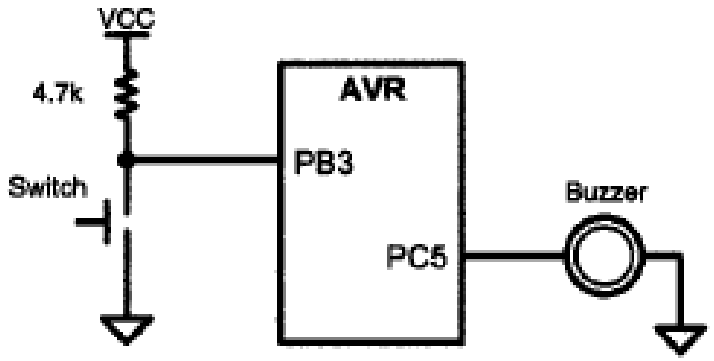
```
1   .include"C: \Appnotes\m8535def.inc"
2   .org 0x0000 ; Original address program
3   rjmp main       ; lompat (menuju) ke prog. main
4
5   main:
6   ldi r16,low(ramend)
7   out spl,r16
8   ldi r16,high(ramend)
9   out sph,r16
10
11  ldi r16,0x00
12  ldi r17,0xff    ; isi register 17 adalah bit 1 semua 0xff
13  dalam hexa atau bias ditulis 0b11111111 (biner)
14  out ddrb,r17    ; Port B sebagai Output
15
16  satu:
17  out portb,r17   ; Pin-pin pada Port B berlogika high
18  rcall delay     ; Pemanggilan program delay
19  dua:
20  out portb,r16   ; Pin-pin pada Port B berlogika low
21  rcall delay
22  rjmp satu       ; lompat menuju subrutin prog. satu
23
24  delay:
25  ldi r18,5
26  delay1:
```

Contoh

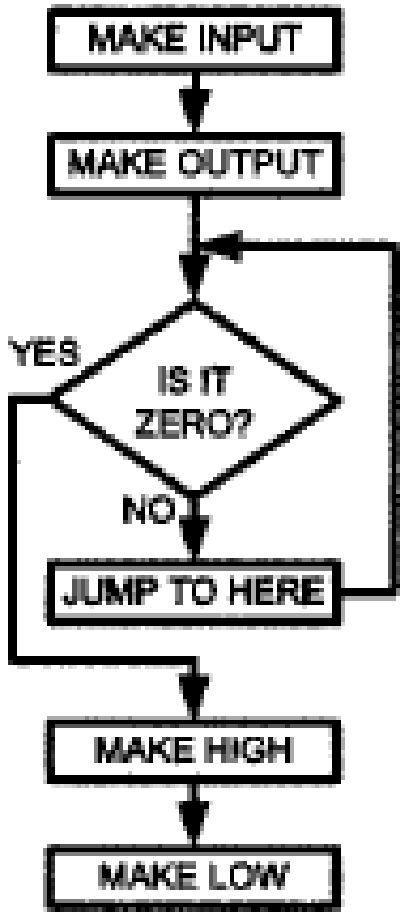
Example 4-5

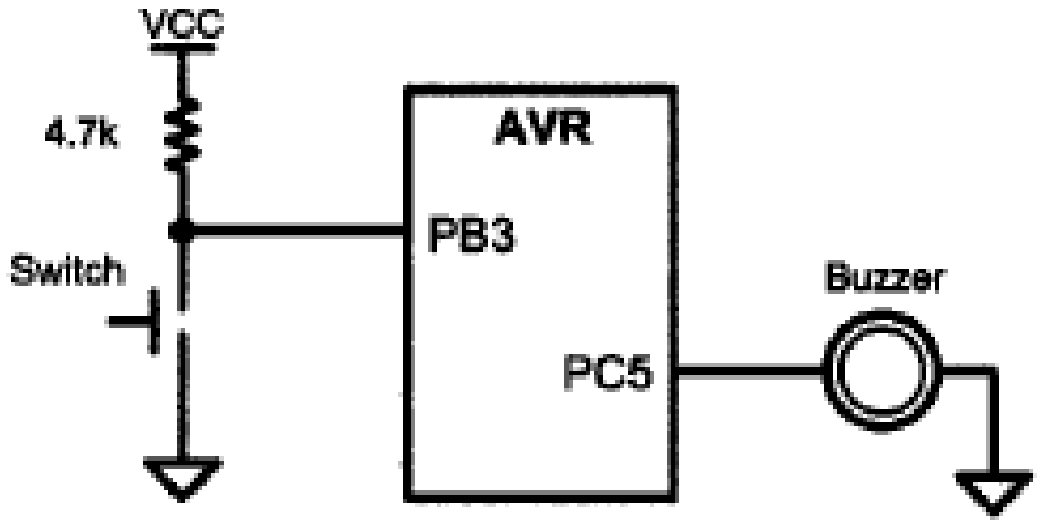
Assume that bit PB3 is an input and represents the condition of a door alarm. If it goes LOW, it means that the door is open. Monitor the bit continuously. Whenever it goes LOW, send a HIGH-to-LOW pulse to port PC5 to turn on a buzzer.





Contoh


MAKE INPUT
MAKE OUTPUT
YES
IS IT ZERO?
NO
JUMP TO HERE
MAKE HIGH
MAKE LOW


VCC
4.7k
Switch
AVR
PB3
PC5
Buzzer

Dasar Bahasa C

Format penulisan

```c
/* program sederhana untuk mejelaskan
    format penulisan program c
    ......
    ......
*/
#include <avr/io.h>      //file include io
#include ......          //preprocesor include
#define on 1             //menggantikan 1 dengan kata on
#define off 0            //preprocesor define
.....
unsigned char data       //variable global
.....
void inisialisasi(void); //prototype fungsi
unsigned int kuadrat (unsigned char);
.....
```

```c
unsigned char x_pangkat_y (char x, char y){ // fungsi
    char z;
    .....
    .....
}
int main (void){    //fungsi utama
    unsigned int temp;      //variable lokal
    .....
    inisialisasi();         //memanggil fungsi inisialisasi
    .....
    temp=kuadrat(15);       //memanggil fungsi kuadrat
    while(1){
        .....
        .....
    }
    return();
}
void inisialisasi (void){  //fungsi
    .....
    .....
}
unsigned int kuadrat (unsigned char x){ //fungsi
    unsigned int y;
    y=x*x;
    return(y);
}
```

Bagian Komentar

- Digunakan untuk memberikan keterangan pada program agar mudah dibaca dan akan diabaikan oleh komputer
- Cara penulisan:

  /* .... */ untuk komentar bentuk paragraf

  // untuk komentar bentuk per baris (sebelum enter)

```
1  /* program sederhana untuk mejelaskan
2       format penulisan program c
3       ......
4       ......
5  */
```

Bagian Preprocessor (6-10)

- Preprocessor #include biasanya digunakan untuk menyertakan file header (.h) atau file library. Berguna untuk memberitahu kompiler agar membaca file yang di include kan lebih dahulu agar mengenali definisi-definisi yang digunakan dalam program agar tidak dianggap error
- File header io.h adalah file yang berisi segala infirmasi / definisi tentang register-register fungsi khusus (SFR) dan bit-bit atau pin-pin mikrokontroller
- Preprocessor #define digunakan untuk mendefinisikan konstanta atau makro

Bagian Preprocessor (6-10)

```
6   #include <avr/io.h>      //file include io
7   #include ......          //preprocesor include
8   #define on 1             //menggantikan 1 dengan kata on
9   #define off 0            //preprocesor define
```

Bagian Deklarasi Variable Global

- Variable global dideklarasikan di luar semua fungsi termasuk fungsi utama dan letaknya harus diatas.
- Sifat variable global yaitu dapat diakses (dibaca/ditulis) oleh semua pernyataan dalam program
- Cara penulisan
  - `TipeData  namaVariable;`

```
11  unsigned char data      //variable global
12  .....
13  void inisialisasi(void); //prototype fungsi
14  unsigned int kuadrat (unsigned char);
```

Bagian Prototype Fungsi (13-14)

- Berguna untuk mendeklarasikan fungsi yang ditulis dibawah fungsi main
- Jika ditulis diatas fungsi main maka tidak usah mendeklarasikan fungsi tersebut, langsung ditulis saja seperti fungsi "x_pangkat_y"
- Cara penulisan:
  - `TipeData  namaFungsi(TipeData, ..., ...); atau`
  - `TipeData  namaFungsi(TipeData namaParameter, ...);`

```
16  unsigned char x_pangkat_y (char x, char y){ // fungsi
17      char z;
18      .....
19      .....
20  }
```

Bagian Fungsi Utama / Main

- Fungsi pertama yang akan dieksekusi dengan urutan dari atas kebawah dan akan "loncat" tergantung pada instruksi lompatan tertentu atau terjadi interupsi jika interupsi diaktifkan
- Penulisan fungsi (call):
  - Tanpa nilai balik (output) dan tanpa parameter (input)

```
namaFungsi();
```

  - Dengan nilai balik (output) dan tanpa parameter (input)

```
variabelPenampung = namaFungsi();
```

  - Dengan nilai balik (output) dan dengan parameter (input)

```
variablePenampung = namaFungsi(var_atau_konstanta, ..., ...);
```

```
21  int main (void){     //fungsi utama
22      unsigned int temp;       //variable lokal
23      .....
24      inisialisasi();          //memanggil fungsi inisialisasi
25      .....
26      temp=kuadrat(15);        //memanggil fungsi kuadrat
27      while(1){
28          .....
29          .....
30      }
31      return();
32  }
```

Bagian Subprogram/fungsi

- Fungsi yang telah di prototypekan ditulis dibawah fungsi main
- Prototype fungsi berguna untuk memudahkan programmer dalam menuliskan program yang besar
- Jika membuat banyak fungsi dan tanpa kita protypekan maka harus ditulis diatas fungsi main dan ini menylitkan untuk dibaca dan diperbarui sehingga kita butuh prototype fungsi

```
33  void inisialisasi (void){  //fungsi
34      .....
35      .....
36  }
37  unsigned int kuadrat (unsigned char x){ //fungsi
38      unsigned int y;
39      y=x*x;
40      return(y);
41  }
```

Penulisan Program C

Dalam penulisan program c, sebenarnya terbagi 2 kategori:

1. **Kategori deklarasi (declaration)**

Deklarasi adalah membuat dan memberi tahu kepada compiler tentang sesuatu yang digunakan nanti dalam penulisan program agar digunakan semestinya dan tidak dianggap error atau asing

```
6   #include <avr/io.h>      //file include io
7   #include ......          //preprocesor include
8   #define on 1             //menggantikan 1 dengan kata on
9   #define off 0            //preprocesor define
```

Penulisan Program C

2. Kategori pernyataan (statement)

Pernyataan adalah membuat instruksi-instruksi program dengan menggunakan keyword seperti instruksi operasi aritmatika, logika, operasi bit atau instruksi percabangan dan looping atau pembuatan fungsi

Kata Kunci (keywords)

- Kata-kata yang termasuk ke dalam keywords tidak bisa digunakan sebagai pengenal (identifier)

`auto`	`double`	`int`	`long`	`const`	`float`	`short`	`unsigned`
`break`	`else`	`long`	`switch`	`continue`	`for`	`signed`	`void`
`case`	`enum`	`register`	`typedef`	`default`	`goto`	`sizeof`	`volatile`
`char`	`extern`	`retrun`	`union`	`do`	`if`	`static`	`while`

Pengenal (identifier)

- Pengenal digunakan untuk memberi nama variabel, fungsi, konstanta dll
- Bahasa C bersifat case sensitive
  - Konstruksi pengenal: huruf, angka, garis bawah (_)
- Tiap pengenal bisa menggunakan gabungan ketiga hal tersebut dengan catatatn tidak boleh diawali dengan angka

Contoh:

```
Menit        //benar
MeniT        //benar dan berbeda dengan pengenal Menit
60detik      //salah
_60detik     //benar
Detik60      //benar
60_detik     //salah
```

Variabel

- Variabel adalah tempat untuk menyimpan dan mengakses data yang mewakili memori dalam mikrokontroller
- Variable harus dideklarasikan dengan**tipe data** beserta **nama variable** yang akan digunakan

Tipe Data	Byte	Bit	min	Max
Char	1	8	-128	127
Signed char	1	8	-128	127
Unsigned char	1	8	0	255
Int	2	16	-32768	32767
Signed int	2	16	-32768	32767
Unsigned int	2	16	0	65535
Long	4	32	-2147483648	2147483647
Signed long	4	32	-2147483648	2147483647
Unsigned long	4	32	0	4294967295
Float	4	32	1,28E-38	3,4E38

Sifat Variable

- Variabel Global: Variabel yang dapat diakses oleh seluruh blok fungsi dalam program
- Variable lokal: variabel yang hanya dapat diakses oleh blok fungsi yang bersangkutan atau terbatas di dalam tanda { } deklarasi fungsi itu berada