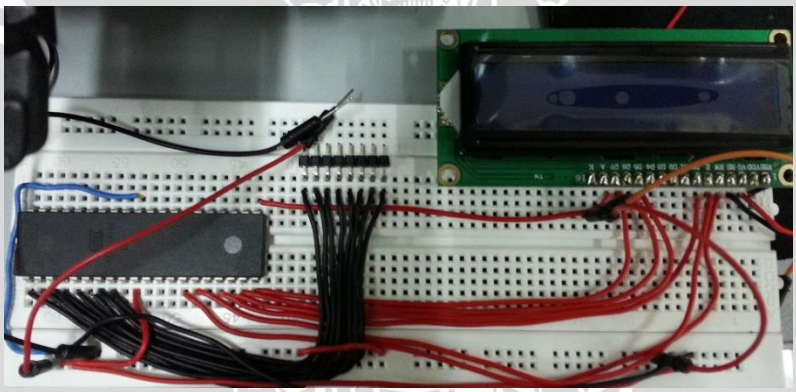
LAMPIRAN I

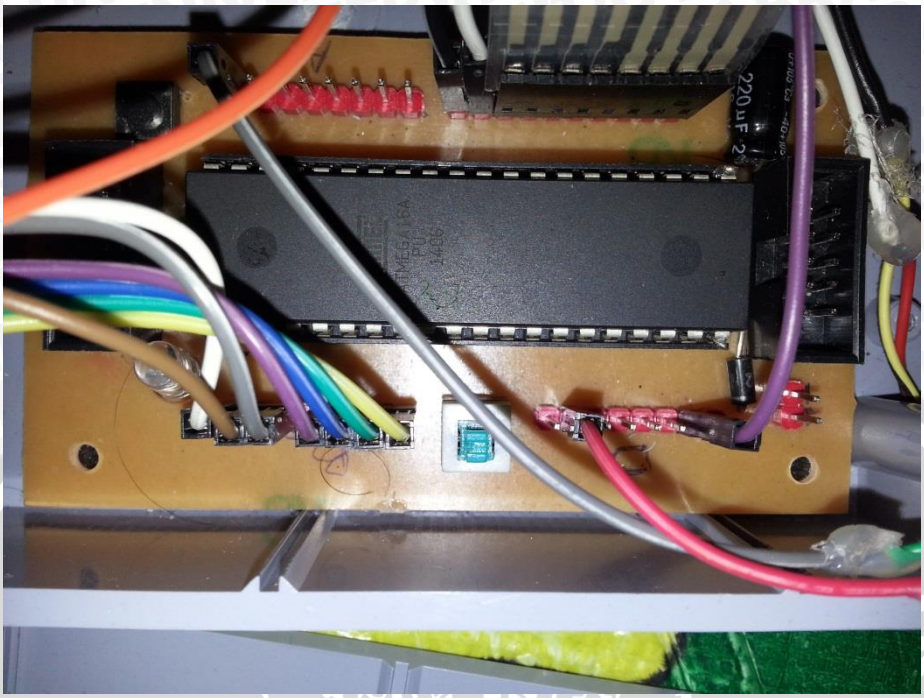
Dokumentasi Alat

Gambar 1. Alat dan bahan pembuatan sistem

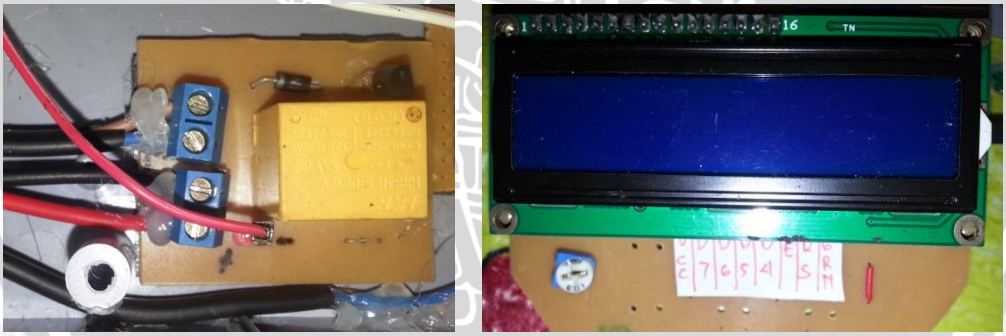


Gambar 2. Perancangan rangkaian elektrik mikrokontroler

Gambar 3. Perancangan dispenser



Gambar 4. Rangkaian minimum sistem ATmega16



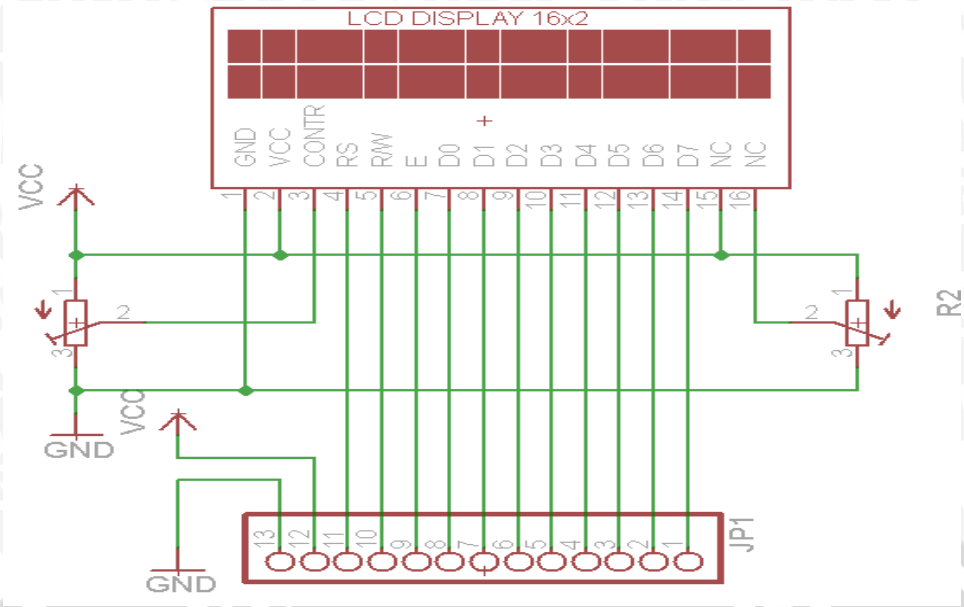
Gambar 5. Rangkaian *driver* relay dan LCD

Gambar 6. Rangkaian elektrik sistem

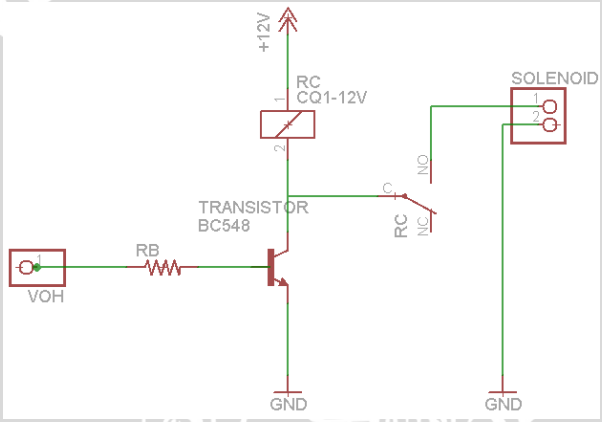
Gambar 7. Sistem keseluruhan

LAMPIRAN II

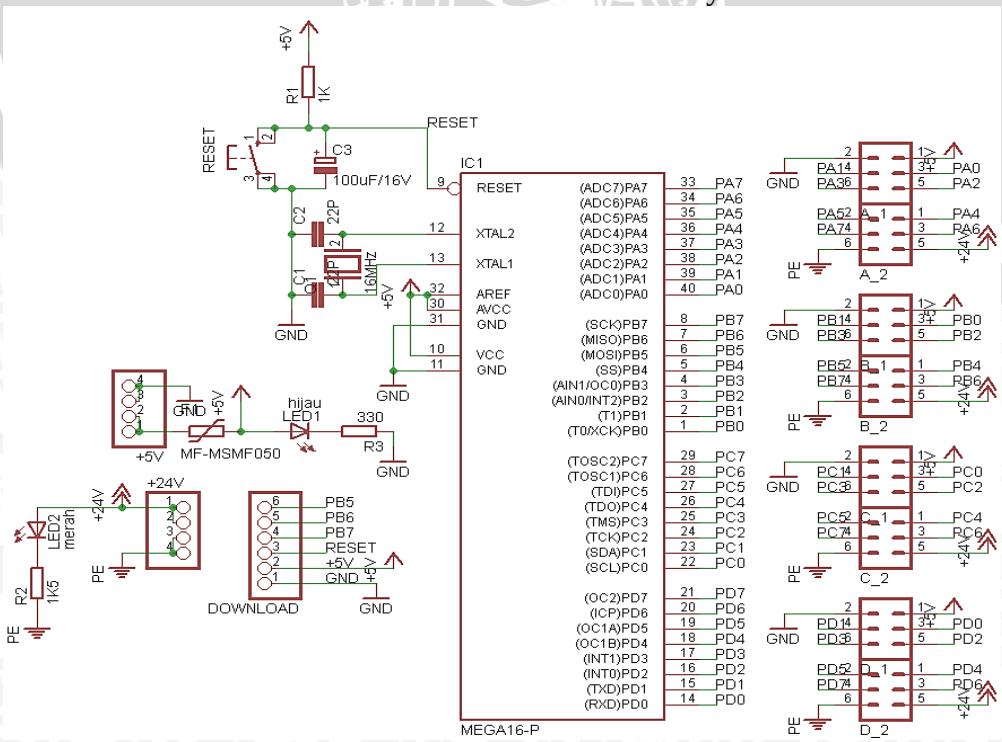
Skematik perancangan alat



Gambar 1. Skematik LCD



Gambar 2. Skematik *Driver* Relay



Gambar 3. Skematik minimum sistem ATmega16

LAMPIRAN III

Listing Program

LISTING PROGRAM MIKROKONTROLER ATMEGA16

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.3 Standard
Automatic Program Generator
© Copyright 1998-2011 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Skripsi
Version : V2.05.3
Date : 5/12/2015
Author : Rizal Pahlevi
Company : dispenser
Comments : Bismillah


Chip type : ATmega16A
Program type : Application
AVR Core Clock frequency : 16.000000 MHz
Memory model : Small
External RAM size : 0
Data Stack size : 256
*****************************************************/

#include <mega16a.h>
// Alphanumeric LCD functions
#include <alcd.h>
#include <delay.h>
#include <stdlib.h>
#include <stdio.h>
// 1 Wire Bus interface functions
#include <1wire.h>
// DS1820 Temperature Sensor functions
#include <ds1820.h>
#define Valve   PORTD.1
#define Max_DS1820  8
#define Switch1     PIND.6  //aktif low >> limit switch di hubungkan ke
ground
// Declare your global variables here
char cetak[16],flag_volume=0;
unsigned int
nominal[5]={0,0,0,0,0},flag_jenisvolume=0,flag_nominal=0,volume;
unsigned int flag_volumehuruf=0;
unsigned int volume_value[5]={0,0,0,0};
unsigned char ds1820_devices, ds1820_rom_codes
[Max_DS1820][9],j;
char i,x;
int Temp;
unsigned int counter_volume=0,delay_valve;

void Baca_Suhu()
  {
  for(i=0;i<ds1820_devices;i++)
    {
    Temp=ds1820_temperature_10(ds1820_rom_codes[i]);
    j='+';
    if(Temp<0)
      {
      j='-';
      Temp=-Temp;
```

```
        }
      sprintf(cetak,"Suhu=%c%i.%u\xdfC",j,Temp/100+8,Temp%100);
      lcd_clear();lcd_gotoxy(0,0);lcd_puts(cetak);
      }
   }

void Scan_Keypad()
   {
   if (flag_nominal==1)
   {
   PORTC = 0b11111110;
   delay_ms(30);
   if (PINC.4 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("D");
delay_ms(50);}
   if (PINC.5 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("C");
delay_ms(50);}
   if (PINC.6 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("B");
delay_ms(50);}
   if (PINC.7 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("A");
delay_ms(50);}
   PORTC = 0b11111101;
   delay_ms(30);
   if (PINC.4 == 0) {lcd_gotoxy(15,0);lcd_putsf("p"); delay_ms(50);}
   if (PINC.5 == 0)
{lcd_gotoxy(15,0);nominal[0]=9;flag_nominal++;lcd_putsf("9");
delay_ms(50);}
   if (PINC.6 == 0)
{lcd_gotoxy(15,0);nominal[0]=6;flag_nominal++;lcd_putsf("6");
delay_ms(50);}
   if (PINC.7 == 0)
{lcd_gotoxy(15,0);nominal[0]=3;flag_nominal++;lcd_putsf("3");
delay_ms(50);}
   PORTC = 0b11111011;
   delay_ms(30);
   if (PINC.4 == 0)
{lcd_gotoxy(15,0);nominal[0]=0;flag_nominal++;lcd_putsf("0");
delay_ms(50);}
   if (PINC.5 == 0)
{lcd_gotoxy(15,0);nominal[0]=8;flag_nominal++;lcd_putsf("8");
delay_ms(50);}
   if (PINC.6 == 0)
{lcd_gotoxy(15,0);nominal[0]=5;flag_nominal++;lcd_putsf("5");
delay_ms(50);}
   if (PINC.7 == 0)
{lcd_gotoxy(15,0);nominal[0]=2;flag_nominal++;lcd_putsf("2");
delay_ms(50);}
   PORTC = 0b11110111;
   delay_ms(30);
   if (PINC.4 == 0) {lcd_gotoxy(15,0);lcd_putsf("b"); delay_ms(50);}
   if (PINC.5 == 0)
{lcd_gotoxy(15,0);nominal[0]=7;flag_nominal++;lcd_putsf("7");
delay_ms(50);}
   if (PINC.6 == 0)
{lcd_gotoxy(15,0);nominal[0]=4;flag_nominal++;lcd_putsf("4");
delay_ms(50);}
   if (PINC.7 == 0)
{lcd_gotoxy(15,0);nominal[0]=1;flag_nominal++;lcd_putsf("1");
delay_ms(50);}
   delay_ms(30);
```

```
    }
    else if (flag_nominal==2)
    {
    PORTC = 0b11111110;
    delay_ms(30);
    if (PINC.4 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("D");
delay_ms(50);}
    if (PINC.5 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("C");
delay_ms(50);}
    if (PINC.6 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("B");
delay_ms(50);}
    if (PINC.7 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("A");
delay_ms(50);}
    PORTC = 0b11111101;
    delay_ms(30);
    if (PINC.4 == 0) {lcd_gotoxy(15,0);lcd_putsf("p"); delay_ms(50);}
    if (PINC.5 == 0)
{lcd_gotoxy(15,0);nominal[1]=9;flag_nominal++;lcd_putsf("9");
delay_ms(50);}
    if (PINC.6 == 0)
{lcd_gotoxy(15,0);nominal[1]=6;flag_nominal++;lcd_putsf("6");
delay_ms(50);}
    if (PINC.7 == 0)
{lcd_gotoxy(15,0);nominal[1]=3;flag_nominal++;lcd_putsf("3");
delay_ms(50);}
    PORTC = 0b11111011;
    delay_ms(30);
    if (PINC.4 == 0)
{lcd_gotoxy(15,0);nominal[1]=0;flag_nominal++;lcd_putsf("0");
delay_ms(50);}
    if (PINC.5 == 0)
{lcd_gotoxy(15,0);nominal[1]=8;flag_nominal++;lcd_putsf("8");
delay_ms(50);}
    if (PINC.6 == 0)
{lcd_gotoxy(15,0);nominal[1]=5;flag_nominal++;lcd_putsf("5");
delay_ms(50);}
    if (PINC.7 == 0)
{lcd_gotoxy(15,0);nominal[1]=2;flag_nominal++;lcd_putsf("2");
delay_ms(50);}
     PORTC = 0b11110111;
     delay_ms(30);
     if (PINC.4 == 0) {lcd_gotoxy(15,0);lcd_putsf("b"); delay_ms(50);}
     if (PINC.5 == 0)
{lcd_gotoxy(15,0);nominal[1]=7;flag_nominal++;lcd_putsf("7");
delay_ms(50);}
     if (PINC.6 == 0)
{lcd_gotoxy(15,0);nominal[1]=4;flag_nominal++;lcd_putsf("4");
delay_ms(50);}
     if (PINC.7 == 0)
{lcd_gotoxy(15,0);nominal[1]=1;flag_nominal++;lcd_putsf("1");
delay_ms(50);}
     delay_ms(30);
     }
     else if (flag_nominal==3)
     {
     PORTC = 0b11111110;
     delay_ms(30);
     if (PINC.4 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("D");
delay_ms(50);}
```

```
   if (PINC.5 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("C");
delay_ms(50);}
   if (PINC.6 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("B");
delay_ms(50);}
   if (PINC.7 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("A");
delay_ms(50);}
   PORTC = 0b11111101;
   delay_ms(30);
   if (PINC.4 == 0) {lcd_gotoxy(15,0);lcd_putsf("p"); delay_ms(50);}
   if (PINC.5 == 0)
{lcd_gotoxy(15,0);nominal[2]=9;flag_nominal++;lcd_putsf("9");
delay_ms(50);}
   if (PINC.6 == 0)
{lcd_gotoxy(15,0);nominal[2]=6;flag_nominal++;lcd_putsf("6");
delay_ms(50);}
   if (PINC.7 == 0)
{lcd_gotoxy(15,0);nominal[2]=3;flag_nominal++;lcd_putsf("3");
delay_ms(50);}
   PORTC = 0b11111011;
   delay_ms(30);
   if (PINC.4 == 0)
{lcd_gotoxy(15,0);nominal[2]=0;flag_nominal++;lcd_putsf("0");
delay_ms(50);}
   if (PINC.5 == 0)
{lcd_gotoxy(15,0);nominal[2]=8;flag_nominal++;lcd_putsf("8");
delay_ms(50);}
   if (PINC.6 == 0)
{lcd_gotoxy(15,0);nominal[2]=5;flag_nominal++;lcd_putsf("5");
delay_ms(50);}
   if (PINC.7 == 0)
{lcd_gotoxy(15,0);nominal[2]=2;flag_nominal++;lcd_putsf("2");
delay_ms(50);}
   PORTC = 0b11110111;
   delay_ms(30);
   if (PINC.4 == 0) {lcd_gotoxy(15,0);lcd_putsf("b"); delay_ms(50);}
   if (PINC.5 == 0)
{lcd_gotoxy(15,0);nominal[2]=7;flag_nominal++;lcd_putsf("7");
delay_ms(50);}
   if (PINC.6 == 0)
{lcd_gotoxy(15,0);nominal[2]=4;flag_nominal++;lcd_putsf("4");
delay_ms(50);}
   if (PINC.7 == 0)
{lcd_gotoxy(15,0);nominal[2]=1;flag_nominal++;lcd_putsf("1");
delay_ms(50);}
   delay_ms(30);
   }
   else if (flag_nominal==4)
   {
   PORTC = 0b11111110;
   delay_ms(30);
   if (PINC.4 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("D");
delay_ms(50);}
   if (PINC.5 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("C");
delay_ms(50);}
   if (PINC.6 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("B");
delay_ms(50);}
   if (PINC.7 == 0) {nominal[0]=0;lcd_gotoxy(15,0);lcd_putsf("A");
delay_ms(50);}
   PORTC = 0b11111101;
```

```
   delay_ms(30);
   if (PINC.4 == 0) {lcd_gotoxy(15,0);lcd_putsf("p"); delay_ms(50);}
   if (PINC.5 == 0)
{lcd_gotoxy(15,0);nominal[3]=9;flag_nominal++;lcd_putsf("9");
delay_ms(50);}
   if (PINC.6 == 0)
{lcd_gotoxy(15,0);nominal[3]=6;flag_nominal++;lcd_putsf("6");
delay_ms(50);}
   if (PINC.7 == 0)
{lcd_gotoxy(15,0);nominal[3]=3;flag_nominal++;lcd_putsf("3");
delay_ms(50);}
   PORTC = 0b11111011;
   delay_ms(30);
   if (PINC.4 == 0)
{lcd_gotoxy(15,0);nominal[3]=0;flag_nominal++;lcd_putsf("0");
delay_ms(50);}
   if (PINC.5 == 0)
{lcd_gotoxy(15,0);nominal[3]=8;flag_nominal++;lcd_putsf("8");
delay_ms(50);}
   if (PINC.6 == 0)
{lcd_gotoxy(15,0);nominal[3]=5;flag_nominal++;lcd_putsf("5");
delay_ms(50);}
   if (PINC.7 == 0)
{lcd_gotoxy(15,0);nominal[3]=2;flag_nominal++;lcd_putsf("2");
delay_ms(50);}
   PORTC = 0b11110111;
   delay_ms(30);
   if (PINC.4 == 0) {lcd_gotoxy(15,0);lcd_putsf("b"); delay_ms(50);}
   if (PINC.5 == 0)
{lcd_gotoxy(15,0);nominal[3]=7;flag_nominal++;lcd_putsf("7");
delay_ms(50);}
   if (PINC.6 == 0)
{lcd_gotoxy(15,0);nominal[3]=4;flag_nominal++;lcd_putsf("4");
delay_ms(50);}
   if (PINC.7 == 0)
{lcd_gotoxy(15,0);nominal[3]=1;flag_nominal++;lcd_putsf("1");
delay_ms(50);}
    delay_ms(30);
    }
    else
    {
    PORTC = 0b11111110;
    delay_ms(30);
    if (PINC.4 == 0)
{flag_volumehuruf=4;lcd_gotoxy(15,0);lcd_putsf("D");
delay_ms(100);}
    if (PINC.5 == 0)
{flag_volumehuruf=3;lcd_gotoxy(15,0);lcd_putsf("C");
delay_ms(100);}
    if (PINC.6 == 0)
{flag_volumehuruf=2;lcd_gotoxy(15,0);lcd_putsf("B");
delay_ms(100);}
    if (PINC.7 == 0)
{flag_volumehuruf=1;lcd_gotoxy(15,0);lcd_putsf("A");
delay_ms(100);}
    PORTC = 0b11111101;
    delay_ms(30);
```

```
   if (PINC.4 == 0) {lcd_gotoxy(15,0);flag_volume=2;lcd_putsf("p");
delay_ms(100);}
   if (PINC.5 == 0) {lcd_gotoxy(15,0);lcd_putsf("9"); delay_ms(100);}
   if (PINC.6 == 0) {lcd_gotoxy(15,0);lcd_putsf("6"); delay_ms(100);}
   if (PINC.7 == 0)
{lcd_gotoxy(15,0);flag_jenisvolume=3;lcd_putsf("3");
delay_ms(100);}
   PORTC = 0b11111011;
   delay_ms(30);
   if (PINC.4 == 0) {lcd_gotoxy(15,0);lcd_putsf("0"); delay_ms(100);}
   if (PINC.5 == 0) {lcd_gotoxy(15,0);lcd_putsf("8"); delay_ms(100);}
   if (PINC.6 == 0) {lcd_gotoxy(15,0);lcd_putsf("5"); delay_ms(100);}
   if (PINC.7 == 0)
{lcd_gotoxy(15,0);flag_jenisvolume=2;lcd_putsf("2");
delay_ms(100);}
   PORTC = 0b11110111;
   delay_ms(30);
   if (PINC.4 == 0) {lcd_gotoxy(15,0);flag_volume=1;lcd_putsf("b");
delay_ms(100);}
   if (PINC.5 == 0) {lcd_gotoxy(15,0);lcd_putsf("7"); delay_ms(100);}
   if (PINC.6 == 0)
{lcd_gotoxy(15,0);flag_jenisvolume=4;lcd_putsf("4");
delay_ms(100);}
   if (PINC.7 == 0)
{lcd_gotoxy(15,0);flag_jenisvolume=1;lcd_putsf("1");
delay_ms(100);}
   delay_ms(30);
   }

   }
```

```
void Reset_Parameter()
   {
   flag_volume=0;flag_nominal=0;flag_jenisvolume=0;
   flag_volumehuruf=0;

nominal[0]=0;nominal[1]=0;nominal[2]=0;nominal[3]=0;nominal[4]=0;
volume=0;
   }
void main(void)
{
// Declare your local variables here
{
// 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=0xFF;
DDRC=0x0F;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In
Func1=Out Func0=In
// State7=P State6=P State5=T State4=T State3=T State2=T State1=0
State0=T
PORTD=0xC2;
DDRD=0x02;

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

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 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: Timer2 Stopped
// Mode: Normal top=0xFF
// 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
// USART disabled
UCSRB=0x00;

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

// ADC initialization
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTB Bit 0
// RD - PORTB Bit 1
// EN - PORTB Bit 2
// D4 - PORTB Bit 4
// D5 - PORTB Bit 5
// D6 - PORTB Bit 6
// D7 - PORTB Bit 7
// Characters/line: 16
lcd_init(16);
// 1 Wire Bus initialization
// 1 Wire Data port: PORTA
// 1 Wire Data bit: 0
// Note: 1 Wire port settings are specified in the
// Project|Configure|C Compiler|Libraries|1 Wire menu.

w1_init();
}
lcd_clear();
for(x=0;x<12;x++)
  {
  lcd_gotoxy(0,0);lcd_putsf("Waiting System");
  lcd_gotoxy(x,1);lcd_putsf(">");
  delay_ms(100);
  }
ds1820_devices=w1_search(0xf0,ds1820_rom_codes); //definisi search
ds1820
lcd_clear(); Valve=0;
awal_set:
```

```
while (1)
    {

 // Place your code here
    Reset_Parameter();
    lcd_clear();
    while(flag_volume!=1 && flag_volumehuruf==0)
    {
      Scan_Keypad();
      Baca_Suhu();
     lcd_gotoxy(0,1);lcd_putsf("*=Set; Char=Start");
    }
    if(flag_volumehuruf!=0)
    {
     switch(flag_volumehuruf)
     {
        case 1:
        {
           lcd_clear();
           while (Switch1==1)
           {
              Baca_Suhu();
              lcd_gotoxy(0,1);lcd_putsf("Tidak ada Gelas");
           }
           lcd_clear();
           sprintf(cetak,"Volume A =
%d",volume_value[1]);lcd_puts(cetak);
           delay_valve=volume_value[1]/100;      //100=nilai debit
valve
           Valve=1;
           counter_volume=0;
           while(counter_volume<delay_valve)
           {
              Baca_Suhu();
sprintf(cetak,"VA=%dml>>C=%dS",volume_value[1],counter_volume)
;
              lcd_gotoxy(0,1);lcd_puts(cetak);
              counter_volume+=1; delay_ms(500);
           }
           counter_volume=0;
           lcd_clear();
           Baca_Suhu();
           lcd_gotoxy(0,1);lcd_putsf("Selesai case 1");
           Valve=0;
        }break;
        case 2:
        {
           lcd_clear();
           while (Switch1==1)
           {
              Baca_Suhu();
              lcd_gotoxy(0,1);lcd_putsf("Tidak ada Gelas");
           }
           lcd_clear();
           lcd_clear();
           sprintf(cetak,"Volume B =
%d",volume_value[2]);lcd_puts(cetak);
           delay_valve=volume_value[2]/100;      //100= nilai debit
valve
```

```
            Valve=1;
            counter_volume=0;
            while(counter_volume<delay_valve)
            {
               Baca_Suhu();

sprintf(cetak,"VA=%dml>>C=%dS",volume_value[2],counter_volume)
;
               lcd_gotoxy(0,1);lcd_puts(cetak);
               counter_volume+=1; delay_ms(500);
            }
            counter_volume=0;
            lcd_clear();
            Baca_Suhu();
            lcd_gotoxy(0,1);lcd_putsf("Selesai");
            Valve=0;
         }break;
         case 3:
         {
            lcd_clear();
            while (Switch1==1)
            {
               Baca_Suhu();
               lcd_gotoxy(0,1);lcd_putsf("Tidak ada Gelas");
            }
            lcd_clear();
            lcd_clear();
            sprintf(cetak,"Volume C =
%d",volume_value[3]);lcd_puts(cetak);
            delay_valve=volume_value[3]/100;   //100= nilai debit valve
```

```
            Valve=1;
            counter_volume=0;
            while(counter_volume<delay_valve)
            {
               Baca_Suhu();

sprintf(cetak,"VA=%dml>>C=%dS",volume_value[3],counter_volume)
;
               lcd_gotoxy(0,1);lcd_puts(cetak);
               counter_volume+=1; delay_ms(500);
            }
            counter_volume=0;
            lcd_clear();
            Baca_Suhu();
            lcd_gotoxy(0,1);lcd_putsf("Selesai");
            Valve=0;
         }break;
         case 4:
         {
            lcd_clear();
            while (Switch1==1)
            {
               Baca_Suhu();
               lcd_gotoxy(0,1);lcd_putsf("Tidak ada Gelas");
            }
            lcd_clear();
            lcd_clear();
            sprintf(cetak,"Volume D =
%d",volume_value[4]);lcd_puts(cetak);
            delay_valve=volume_value[4]/100;  //100= nilai debit valve
```

```
            Valve=1;
            counter_volume=0;
            while(counter_volume<delay_valve)
            {
               Baca_Suhu();

sprintf(cetak,"VA=%dml>>C=%dS",volume_value[4],counter_volume)
;
               lcd_gotoxy(0,1);lcd_puts(cetak);
               counter_volume+=1;  delay_ms(500);
            }
            counter_volume=0;
            lcd_clear();
            Baca_Suhu();
            lcd_gotoxy(0,1);lcd_putsf("Selesai");
            Valve=0;
          }break;
          default:
          break;
          }
       delay_ms(400);
       flag_volumehuruf=0;
       goto awal_set;
       }
     lcd_clear();

     while(flag_jenisvolume==0)
     {
       Scan_Keypad();
       lcd_gotoxy(0,0);lcd_putsf("Press Number ");
        lcd_gotoxy(0,1);lcd_putsf("Select Volume");
       }

       lcd_clear();
       lcd_gotoxy(0,1);sprintf(cetak,"Select
Volume=%d",flag_jenisvolume);lcd_puts(cetak);
       delay_ms(300);
       lcd_clear();
       flag_nominal=1;
       while(flag_volume!=2)
        {   Scan_Keypad();
           lcd_gotoxy(0,1);
            sprintf(cetak,"Volume %d=
%d%d%d%d",flag_jenisvolume,nominal[0],nominal[1],nominal[2],no
minal[3]);lcd_puts(cetak);
        }

       nominal[0]=nominal[0]*1000;
       nominal[1]=nominal[1]*100;
       nominal[2]=nominal[2]*10;
       nominal[3]=nominal[3]*1;
       volume=nominal[0]+nominal[1]+nominal[2]+nominal[3];
       volume_value[flag_jenisvolume]=volume;
       lcd_clear();

       switch(flag_jenisvolume)
       {
          case 1:
          {
               sprintf(cetak,"Suhu=%c%i.%u\xdfC",j,Temp/100,Temp%10);
```

```
            lcd_gotoxy(0,0);lcd_puts(cetak);
            lcd_gotoxy(0,1);
            sprintf(cetak,"Voleme A = %d",volume_value[1]);
lcd_puts(cetak);
        }break;
        case 2:
        {
            sprintf(cetak,"Suhu=%c%i.%u\xdfC",j,Temp/100,Temp%10);
            lcd_gotoxy(0,0);lcd_puts(cetak);
            lcd_gotoxy(0,1);
            sprintf(cetak,"Voleme B = %d",volume_value[2]);
lcd_puts(cetak);
        }break;
        case 3:
        {
            sprintf(cetak,"Suhu=%c%i.%u\xdfC",j,Temp/100,Temp%10);
            lcd_gotoxy(0,0);lcd_puts(cetak);
            lcd_gotoxy(0,1);
            sprintf(cetak,"Voleme C = %d",volume_value[3]);
lcd_puts(cetak);
        }break;
        case 4:
        {
            sprintf(cetak,"Suhu=%c%i.%u\xdfC",j,Temp/100,Temp%10);
            lcd_gotoxy(0,0);lcd_puts(cetak);
            lcd_gotoxy(0,1);
            sprintf(cetak,"Voleme D = %d",volume_value[4]);
lcd_puts(cetak);
        }break;
        default:
            break;
    }
    delay_ms(300);
    lcd_clear();
    lcd_putsf("Setting Done..");
    delay_ms(300);
    flag_volume=0;flag_nominal=0;flag_jenisvolume=0;

nominal[0]=0;nominal[1]=0;nominal[2]=0;nominal[3]=0;nominal[4]=0;
volume=0;
 }
}
```

LAMPIRAN III

Datasheet

Features

- High-performance, Low-power Atmel® AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 16 Kbytes of In-System Self-programmable Flash program memory
  - 512 Bytes EEPROM
  - 1 Kbyte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C[1]
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    - 8 Single-ended Channels
    - 7 Differential Channels in TQFP Package Only
    - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V for ATmega16L
  - 4.5V - 5.5V for ATmega16
- Speed Grades
  - 0 - 8 MHz for ATmega16L
  - 0 - 16 MHz for ATmega16
- Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
  - Active: 1.1 mA
  - Idle Mode: 0.35 mA
  - Power-down Mode: < 1 µA

ATMEL

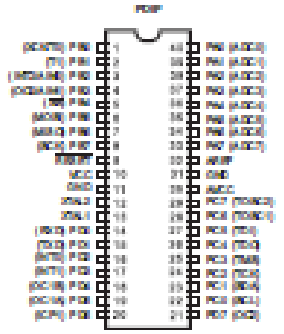
8-bit AVR® Microcontroller with 16K Bytes In-System Programmable Flash

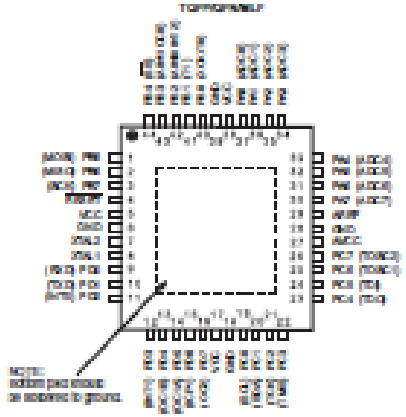
ATmega16
ATmega16L

Rev. 2466T–AVR–07/10

Pin Configurations

Figure 1. Pinout ATmega16





Disclaimer

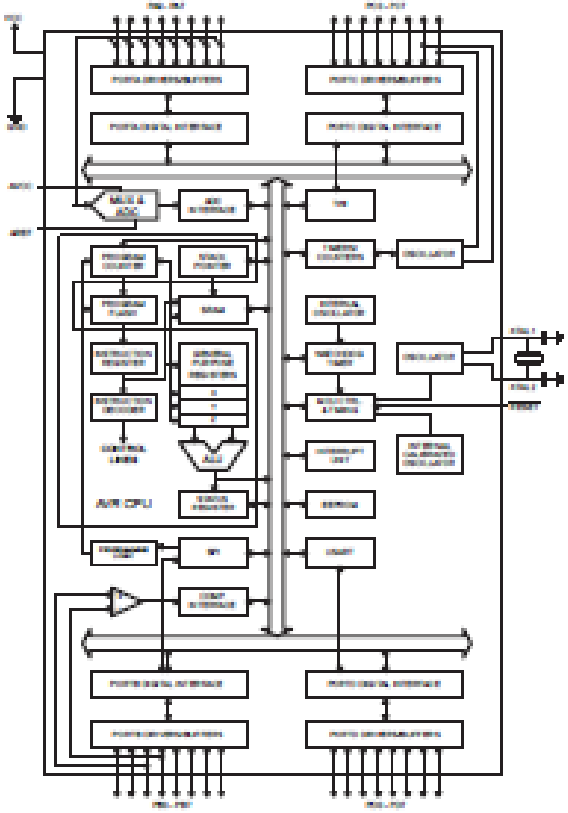
Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Overview

The ATmega16 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega16 provides the following features: 16 Kbytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 512 bytes EEPROM, 1 Kbyte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire Interface, A/D Converter, SRAM, Timer/Counters, SPI port, and Interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega16 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega16 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, In-circuit emulators, and evaluation kits.

Pin Descriptions

VCC Digital supply voltage.

GND Ground.

Port A (PA7..PA0) Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB7..PB0) Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega16 as listed on page 58.

Port C (PC7..PC0) Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.

Port C also serves the functions of the JTAG interface and other special features of the ATmega16 as listed on page 61.

Port D (PD7..PD0) Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega16 as listed on page 63.

$\overline{\text{RESET}}$ Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 38. Shorter pulses are not guaranteed to generate a reset.

XTAL1 Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2 Output from the inverting Oscillator amplifier.

AVCC AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

AREF AREF is the analog reference pin for the A/D Converter.

Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://www.atmel.com/avr.

Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

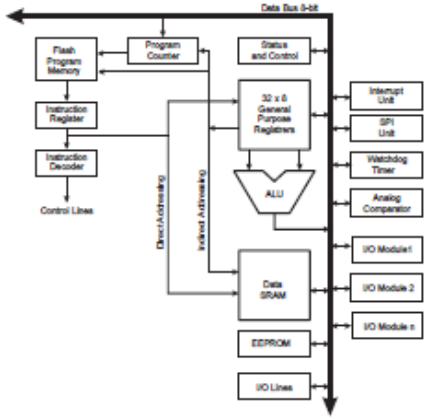
AVR CPU Core

Introduction

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

Architectural Overview

Figure 3. Block Diagram of the AVR MCU Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash Program memory. These added function registers are the 16-bit X-register, Y-register, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16-bit or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The Stack Pointer SP is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the Status Register. All interrupts have a separate interrupt vector in the interrupt vector table. The interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, $20 - $5F.

ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the "Instruction Set" section for a detailed description.

Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

- **Bit 6 – T: Bit Copy Storage**

The Bit Copy Instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

- **Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry is useful in BCD arithmetic. See the "Instruction Set Description" for detailed information.

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

The S-bit is always an exclusive or between the Negative Flag N and the Two's Complement Overflow Flag V. See the "Instruction Set Description" for detailed information.

- **Bit 3 – V: Two's Complement Overflow Flag**

The Two's Complement Overflow Flag V supports two's complement arithmetics. See the "Instruction Set Description" for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

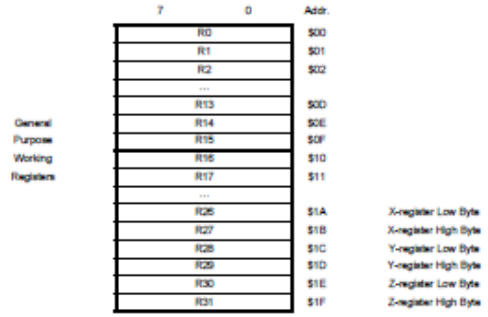
General Purpose Register File

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Figure 4 shows the structure of the 32 general purpose working registers in the CPU.

Figure 4. AVR CPU General Purpose Working Registers



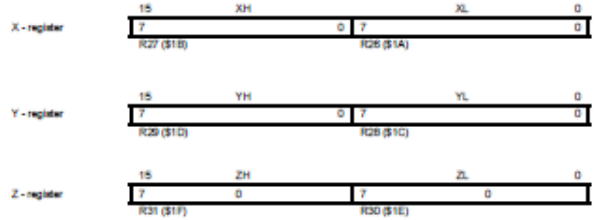
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 4, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-pointer Registers can be set to index any register in the file.

The X-register, Y-register and Z-register

The registers R26..R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the Data Space. The three indirect address registers X, Y, and Z are defined as described in Figure 5.

Figure 5. The X-register, Y-register, and Z-register



In the different addressing modes these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see the Instruction Set Reference for details).

Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer. If software reads the Program Counter from the Stack after a call or an interrupt, unused bits (15:13) should be masked out.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above $60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 6 shows the parallel Instruction fetches and Instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 6. The Parallel Instruction Fetches and Instruction Executions

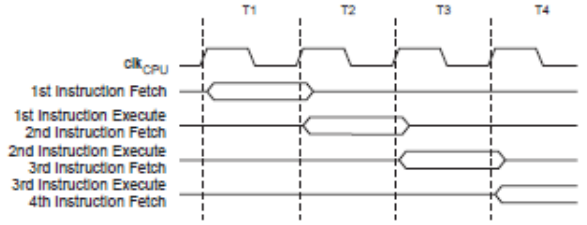
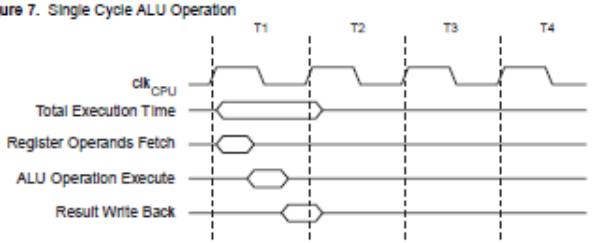


Figure 7 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 7. Single Cycle ALU Operation



Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate reset vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security. See the section "Memory Programming" on page 259 for details.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in "Interrupts" on page 45. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request

AVR ATmega16 Memories

This section describes the different memories in the ATmega16. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega16 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

In-System Reprogrammable Flash Program Memory

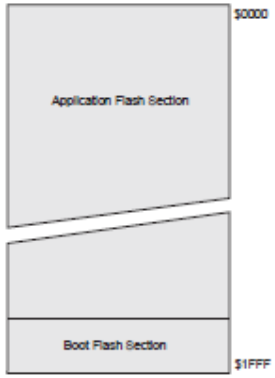
The ATmega16 contains 16 Kbytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 8K x 16. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega16 Program Counter (PC) is 13 bits wide, thus addressing the 8K program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in "Boot Loader Support – Read-While-Write Self-Programming" on page 246. "Memory Programming" on page 259 contains a detailed description on Flash data serial downloading using the SPI pins or the JTAG interface.

Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory instruction Description).

Timing diagrams for instruction fetch and execution are presented in "Instruction Execution Timing" on page 13.

Figure 8. Program Memory Map



SRAM Data Memory

Figure 9 shows how the ATmega16 SRAM Memory is organized.

The lower 1120 Data Memory locations address the Register File, the I/O Memory, and the internal data SRAM. The first 96 locations address the Register File and I/O Memory, and the next 1024 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register File, registers R26 to R31 feature the indirect addressing pointer registers.

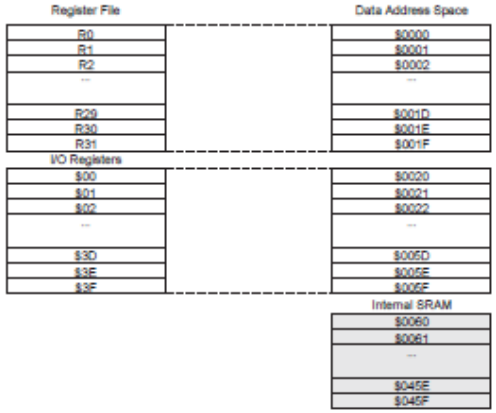
The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y-register or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O Registers, and the 1024 bytes of internal data SRAM in the ATmega16 are all accessible through all these addressing modes. The Register File is described in "General Purpose Register File" on page 11.

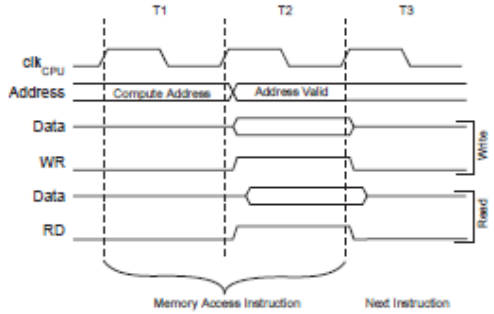
Figure 9. Data Memory Map



Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 10.

Figure 10. On-chip Data SRAM Access Cycles



EEPROM Data Memory

The ATmega16 contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For a detailed description of SPI, JTAG, and Parallel data downloading to the EEPROM, see page 273, page 278, and page 262, respectively.

EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space.

The write access time for the EEPROM is given in Table 1. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on Power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See "Preventing EEPROM Corruption" on page 22 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

The EEPROM Address Register – EEARH and EEARL

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	–	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

- **Bits 15..9 – Res: Reserved Bits**

These bits are reserved bits in the ATmega16 and will always read as zero.

- **Bits 8..0 – EEAR8..0: EEPROM Address**

The EEPROM Address Registers – EEARH and EEARL – specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

The EEPROM Data Register – EEDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – EEDR7.0: EEPROM Data**

For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

The EEPROM Control Register – EECR

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

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega16 and will always read as zero.

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**

Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEWE is cleared.

- **Bit 2 – EEMWE: EEPROM Master Write Enable**

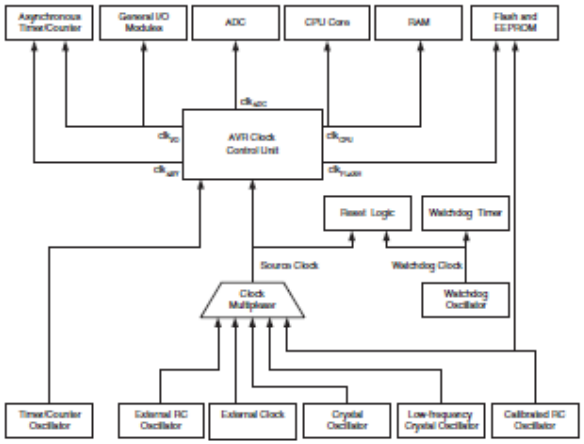
The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set, setting EEWE within four clock cycles will write data to the EEPROM at the selected address if EEMWE is zero, setting EEWE will have no effect. When EEMWE has been written to one by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for an EEPROM write procedure.

System Clock and Clock Options

Clock Systems and their Distribution

Figure 11 presents the principal clock systems in the AVR and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in "Power Management and Sleep Modes" on page 32. The clock systems are detailed Figure 11.

Figure 11. Clock Distribution



CPU Clock – clk_{CPU}

The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the General Purpose Register File, the Status Register and the data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like Timer/Counters, SPI, and USART. The I/O clock is also used by the External Interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted. Also note that address recognition in the TWI module is carried out asynchronously when $clk_{I/O}$ is halted, enabling TWI address reception in all sleep modes.

Flash Clock – clk_{FLASH}

The Flash clock controls operation of the Flash interface. The Flash clock is usually active simultaneously with the CPU clock.

Asynchronous Timer Clock – clk_{ASY}

The Asynchronous Timer clock allows the Asynchronous Timer/Counter to be clocked directly from an external 32 kHz clock crystal. The dedicated clock domain allows using this Timer/Counter as a real-time counter even when the device is in sleep mode.

ADC Clock – clk_{ADC}

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

Clock Sources

The device has the following clock source options, selectable by Flash Fuse bits as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

Table 2. Device Clocking Options Select(1)

Device Clocking Option	CKSEL3..0
External Crystal/Ceramic Resonator	1111 - 1010
External Low-frequency Crystal	1001
External RC Oscillator	1000 - 0101
Calibrated Internal RC Oscillator	0100 - 0001
External Clock	0000

Note: 1. For all fuses "1" means unprogrammed while "0" means programmed.

The various choices for each clocking option is given in the following sections. When the CPU wakes up from Power-down or Power-save, the selected clock source is used to time the start-up, ensuring stable Oscillator operation before instruction execution starts. When the CPU starts from Reset, there is as an additional delay allowing the power to reach a stable level before commencing normal operation. The Watchdog Oscillator is used for timing this real-time part of the start-up time. The number of WDT Oscillator cycles used for each time-out is shown in Table 3. The frequency of the Watchdog Oscillator is voltage dependent as shown in "ATmega16 Typical Characteristics" on page 299.

Table 3. Number of Watchdog Oscillator Cycles

Typ Time-out (V_{CC} = 6.0V)	Typ Time-out (V_{CC} = 3.0V)	Number of Cycles
4.1 ms	4.3 ms	4K (4,096)
65 ms	69 ms	64K (65,536)

Default Clock Source

The device is shipped with CKSEL = "0001" and SUT = "10". The default clock source setting is therefore the 1 MHz Internal RC Oscillator with longest startup time. This default setting ensures that all users can make their desired clock source setting using an In-System or Parallel Programmer.

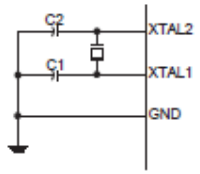
Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in Figure 12. Either a quartz crystal or a ceramic resonator may be used. The CKOPT Fuse selects between two different Oscillator amplifier modes. When CKOPT is programmed, the Oscillator output will oscillate will a full rail-to-rail swing on the output. This mode is suitable when operating in a very noisy environment or when the output from XTAL2 drives a second clock buffer. This mode has a wide frequency range. When CKOPT is unprogrammed, the Oscillator has a smaller output swing. This reduces power consumption considerably. This mode has a limited frequency range and it can not be used to drive other clock buffers.

For resonators, the maximum frequency is 8 MHz with CKOPT unprogrammed and 16 MHz with CKOPT programmed. C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for

choosing capacitors for use with crystals are given in Table 4. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 12. Crystal Oscillator Connections



The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1 as shown in Table 4.

Table 4. Crystal Oscillator Operating Modes

CKOPT	CKSEL3..1	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
1	101[1]	0.4 - 0.9	–
1	110	0.9 - 3.0	12 - 22
1	111	3.0 - 8.0	12 - 22
0	101, 110, 111	1.0 ≤	12 - 22

Note: 1. This option should not be used with crystals, only with ceramic resonators.

The CKSEL0 Fuse together with the SUT1..0 Fuses select the start-up times as shown in Table 5.

Table 5. Start-up Times for the Crystal Oscillator Clock Selection

CKSEL0	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
0	00	258 CK[1]	4.1 ms	Ceramic resonator, fast rising power
0	01	258 CK[1]	65 ms	Ceramic resonator, slowly rising power
0	10	1K CK[2]	–	Ceramic resonator, BOD enabled
0	11	1K CK[2]	4.1 ms	Ceramic resonator, fast rising power
1	00	1K CK[2]	65 ms	Ceramic resonator, slowly rising power
1	01	16K CK	–	Crystal Oscillator, BOD enabled
1	10	16K CK	4.1 ms	Crystal Oscillator, fast rising power
1	11	16K CK	65 ms	Crystal Oscillator, slowly rising power

Notes: 1. These options should only be used when not operating close to the maximum frequency of the device, and only if frequency stability at start-up is not important for the application. These options are not suitable for crystals.
2. These options are intended for use with ceramic resonators and will ensure frequency stability at start-up. They can also be used with crystals when not operating close to the maximum frequency of the device, and if frequency stability at start-up is not important for the application.

System Control and Reset

Resetting the AVR

During Reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be a JMP – absolute jump – instruction to the reset handling routine. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa. The circuit diagram in Figure 15 shows the reset logic. Table 15 defines the electrical parameters of the reset circuitry.

The I/O ports of the AVR are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

After all reset sources have gone inactive, a delay counter is invoked, stretching the internal Reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the CKSEL Fuses. The different selections for the delay period are presented in "Clock Sources" on page 25.

Reset Sources

The ATmega16 has five sources of reset:

- **Power-on Reset.**
  The MCU is reset when the supply voltage is below the Power-on Reset threshold (V_{POT}).
- **External Reset.**
  The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for longer than the minimum pulse length.
- **Watchdog Reset.**
  The MCU is reset when the Watchdog Timer period expires and the Watchdog is enabled.
- **Brown-out Reset.**
  The MCU is reset when the supply voltage V_{CC} is below the Brown-out Reset threshold (V_{BOT}) and the Brown-out Detector is enabled.
- **JTAG AVR Reset.**
  The MCU is reset as long as there is a logic one in the Reset Register, one of the scan chains of the JTAG system. Refer to the section "IEEE 1149.1 (JTAG) Boundary-scan" on page 226 for details.

Figure 15. Reset Logic

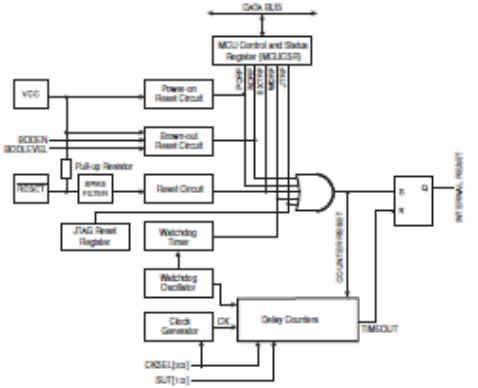


Table 15. Reset Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)			1.4	2.3	V
	Power-on Reset Threshold Voltage (falling)[1]			1.3	2.3	
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		$0.1V_{CC}$		$0.9V_{CC}$	
t_{RST}	Minimum pulse width on $\overline{\text{RESET}}$ Pin				1.5	µs
V_{BOT}	Brown-out Reset Threshold Voltage[2]	BODLEVEL = 1	2.5	2.7	3.2	V
		BODLEVEL = 0	3.6	4.0	4.5	
t_{BOD}	Minimum low voltage period for Brown-out Detection	BODLEVEL = 1		2		µs
		BODLEVEL = 0		2		
V_{HYST}	Brown-out Detector hysteresis			50		mV

Notes: 1. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling).
2. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to $V_{CC} = V_{BOT}$ during the production test. This guarantees that a Brown-out Reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed. The test is performed using BODLEVEL = 1 for ATmega16L and BODLEVEL = 0 for ATmega16. BODLEVEL = 1 is not applicable for ATmega16.

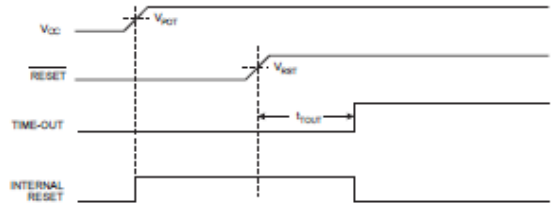
Power-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 15. The POR is activated whenever V_{CC} is below the detection level. The POR circuit can be used to trigger the Start-up Reset, as well as to detect a failure in supply voltage.

A Power-on Reset (POR) circuit ensures that the device is reset from Power-on. Reaching the Power-on Reset threshold voltage invokes the delay counter, which determines how long the device is kept in RESET after V_{CC} rise. The RESET signal is activated again, without any delay, when V_{CC} decreases below the detection level.

Figure 16. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}.

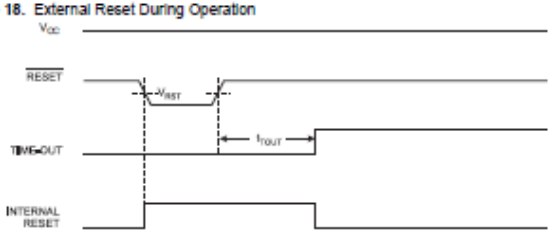
Figure 17. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



External Reset

An External Reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than the minimum pulse width (see Table 15) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the Reset Threshold Voltage – V_{RST} – on its positive edge, the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

Figure 18. External Reset During Operation



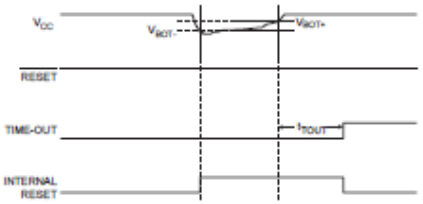
Brown-out Detection

ATmega16 has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the fuse BODLEVEL to be 2.7V (BODLEVEL unprogrammed), or 4.0V (BODLEVEL programmed). The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

The BOD circuit can be enabled/disabled by the fuse BODEN. When the BOD is enabled (BODEN programmed), and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 19), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 19), the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in Table 15.

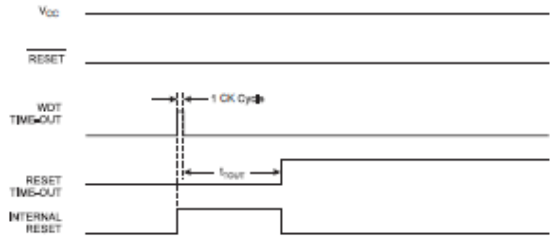
Figure 19. Brown-out Reset During Operation



Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}. Refer to page 42 for details on operation of the Watchdog Timer.

Figure 20. Watchdog Reset During Operation



MCU Control and Status Register – MCUCSR

The MCU Control and Status Register provides information on which reset source caused an MCU Reset.

Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

- **Bit 4 – JTRF: JTAG Reset Flag**

This bit is set if a reset is being caused by a logic one in the JTAG Reset Register selected by the JTAG instruction AVR_RESET. This bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 3 – WDRF: Watchdog Reset Flag**

This bit is set if a Watchdog Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 2 – BORF: Brown-out Reset Flag**

This bit is set if a Brown-out Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 1 – EXTRF: External Reset Flag**

This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 0 – PORF: Power-on Reset Flag**

This bit is set if a Power-on Reset occurs. The bit is reset only by writing a logic zero to the flag.

To make use of the Reset Flags to identify a reset condition, the user should read and then reset the MCUCSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.

Internal Voltage Reference

ATmega16 features an internal bandgap reference. This reference is used for Brown-out Detection, and it can be used as an input to the Analog Comparator or the ADC. The 2.56V reference to the ADC is generated from the internal bandgap reference.

Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in Table 16. To save power, the reference is not always turned on. The reference is on during the following situations:

1. When the BOD is enabled (by programming the BODEN Fuse).
2. When the bandgap reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR).
3. When the ADC is enabled.

Thus, when the BOD is not enabled, after setting the ACBG bit or enabling the ADC, the user must always allow the reference to start up before the output from the Analog Comparator or ADC is used. To reduce power consumption in Power-down mode, the user can avoid the three conditions above to ensure that the reference is turned off before entering Power-down mode.

Table 16. Internal Voltage Reference Characteristics

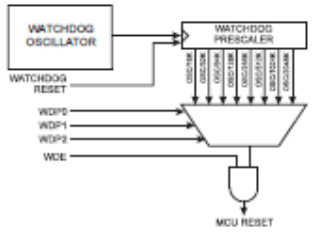
Symbol	Parameter	Min	Typ	Max	Units
V_{BG}	Bandgap reference voltage	1.15	1.23	1.4	V
t_{BG}	Bandgap reference start-up time		40	70	µs
I_{BG}	Bandgap reference current consumption		10		µA

Watchdog Timer

The Watchdog Timer is clocked from a separate On-chip Oscillator which runs at 1 MHz. This is the typical value at V_{CC} = 5V. See characterization data for typical values at other V_{CC} levels. By controlling the Watchdog Timer prescaler, the Watchdog Reset interval can be adjusted as shown in Table 17 on page 43. The WDR – Watchdog Reset – instruction resets the Watchdog Timer. The Watchdog Timer is also reset when it is disabled and when a Chip Reset occurs. Eight different clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog Reset, the ATmega16 resets and executes from the Reset Vector. For timing details on the Watchdog Reset, refer to page 41.

To prevent unintentional disabling of the Watchdog, a special turn-off sequence must be followed when the Watchdog is disabled. Refer to the description of the Watchdog Timer Control Register for details.

Figure 21. Watchdog Timer



Watchdog Timer Control Register – WDTCR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..5 – Res: Reserved Bits**

These bits are reserved bits in the ATmega16 and will always read as zero.

- **Bit 4 – WDTOE: Watchdog Turn-off Enable**

This bit must be set when the WDE bit is written to logic zero. Otherwise, the Watchdog will not be disabled. Once written to one, hardware will clear this bit after four clock cycles. Refer to the description of the WDE bit for a Watchdog disable procedure.

- **Bit 3 – WDE: Watchdog Enable**

When the WDE is written to logic one, the Watchdog Timer is enabled, and if the WDE is written to logic zero, the Watchdog Timer function is disabled. WDE can only be cleared if the WDTOE bit has logic level one. To disable an enabled Watchdog Timer, the following procedure must be followed:

1. In the same operation, write a logic one to WDTOE and WDE. A logic one must be written to WDE even though it is set to one before the disable operation starts.
2. Within the next four clock cycles, write a logic 0 to WDE. This disables the Watchdog.

- **Bits 2..0 – WDP2, WDP1, WDP0: Watchdog Timer Prescaler 2, 1, and 0**

The WDP2, WDP1, and WDP0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 17.

Table 17. Watchdog Timer Prescale Select

WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 3.0V	Typical Time-out at V_{CC} = 5.0V
0	0	0	16K (16,384)	17.1 ms	16.3 ms
0	0	1	32K (32,768)	34.3 ms	32.5 ms
0	1	0	64K (65,536)	68.5 ms	65 ms
0	1	1	128K (131,072)	0.14 s	0.13 s
1	0	0	256K (262,144)	0.27 s	0.26 s
1	0	1	512K (524,288)	0.55 s	0.52 s
1	1	0	1,024K (1,048,576)	1.1 s	1.0 s
1	1	1	2,048K (2,097,152)	2.2 s	2.1 s

The following code example shows one assembly and one C function for turning off the WDT. The example assumes that interrupts are controlled (for example by disabling interrupts globally) so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
WDT_off:
  ; Reset WDT
  WDR
  ; Write logical one to WDTOE and WDE
  in   r16, WDTCR
  ori r16, (1<<WDTOE)|(1<<WDE)
  out  WDTCR, r16
  ; Turn off WDT
  ldi  r16, (0<<WDE)
  out  WDTCR, r16
  ret
```

C Code Example

```
void WDT_off(void)
{
  /* Reset WDT*/
  _WDR();
  /* Write logical one to WDTOE and WDE */
  WDTCR |= (1<<WDTOE) | (1<<WDE);
  /* Turn off WDT */
  WDTCR = 0x00;
}
```

Interrupts

This section describes the specifics of the interrupt handling as performed in ATmega16. For a general explanation of the AVR interrupt handling, refer to "Reset and Interrupt Handling" on page 13.

Interrupt Vectors in ATmega16

Table 18. Reset and Interrupt Vectors

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow
11	$014	SPI, STC	Serial Transfer Complete
12	$016	USART, RXC	USART, Rx Complete
13	$018	USART, UDRE	USART Data Register Empty
14	$01A	USART, TXC	USART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface
19	$024	INT2	External Interrupt Request 2
20	$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	$028	SPM_RDY	Store Program Memory Ready

Notes: 1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Boot Loader Support – Read-While-Write Self-Programming" on page 246.
2. When the IVSEL bit in GICR is set, interrupt vectors will be moved to the start of the Boot Flash section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash section.

Table 19 shows Reset and Interrupt Vectors placement for the various combinations of BOOTRST and IVSEL settings. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa.

Table 19. Reset and Interrupt Vectors Placement(1)

BOOTRST	IVSEL	Reset address	Interrupt Vectors Start Address
1	0	$0000	$0002
1	1	$0000	Boot Reset Address + $0002
0	0	Boot Reset Address	$0002
0	1	Boot Reset Address	Boot Reset Address + $0002

Note: 1. The Boot Reset Address is shown in Table 100 on page 257. For the BOOTRST Fuse "1" means unprogrammed while "0" means programmed.

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega16 is:

```
Address   Labels   Code                 Comments
$000              jmp   RESET          ; Reset Handler
$002              jmp   EXT_INT0       ; IRQ0 Handler
$004              jmp   EXT_INT1       ; IRQ1 Handler
$006              jmp   TIM2_COMP      ; Timer2 Compare Handler
$008              jmp   TIM2_OVF       ; Timer2 Overflow Handler
$00A              jmp   TIM1_CAPT      ; Timer1 Capture Handler
$00C              jmp   TIM1_COMPA     ; Timer1 CompareA Handler
$00E              jmp   TIM1_COMPB     ; Timer1 CompareB Handler
$010              jmp   TIM1_OVF       ; Timer1 Overflow Handler
$012              jmp   TIM0_OVF       ; Timer0 Overflow Handler
$014              jmp   SPI_STC        ; SPI Transfer Complete Handler
$016              jmp   USART_RXC      ; USART RX Complete Handler
$018              jmp   USART_UDRE     ; UDR Empty Handler
$01A              jmp   USART_TXC      ; USART TX Complete Handler
$01C              jmp   ADC            ; ADC Conversion Complete Handler
$01E              jmp   EE_RDY         ; EEPROM Ready Handler
$020              jmp   ANA_COMP       ; Analog Comparator Handler
$022              jmp   TWSI           ; Two-wire Serial Interface Handler
$024              jmp   EXT_INT2       ; IRQ2 Handler
$026              jmp   TIM0_COMP      ; Timer0 Compare Handler
$028              jmp   SPM_RDY        ; Store Program Memory Ready Handler
;
$02A     RESET:   ldi   r16,high(RAMEND) ; Main program start
$02B              out   SPH,r16        ; Set Stack Pointer to top of RAM
$02C              ldi   r16,low(RAMEND)
$02D              out   SPL,r16
$02E              sei                  ; Enable interrupts
$02F              <instr>  xxx
...      ...      ...
```

Moving Interrupts Between Application and Boot Space

The General Interrupt Control Register controls the placement of the Interrupt Vector table.

General Interrupt Control Register – GICR

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 1 – IVSEL: Interrupt Vector Select**

When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash section is determined by the BOOTSZ Fuses. Refer to the section "Boot Loader Support – Read-While-Write Self-Programming" on page 246 for details. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

1. Write the Interrupt Vector Change Enable (IVCE) bit to one.
2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

Interrupts will automatically be disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to IVSEL. If IVSEL is not written, Interrupts remain disabled for four cycles. The I-bit in the Status Register is unaffected by the automatic disabling.

Note: If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and Boot Lock bit BLB12 is programed, interrupts are disabled while executing from the Boot Loader section. Refer to the section "Boot Loader Support – Read-While-Write Self-Programming" on page 246 for details on Boot Lock bits.

- **Bit 0 – IVCE: Interrupt Vector Change Enable**

The IVCE bit must be written to logic one to enable change of the IVSEL bit. IVCE is cleared by hardware four cycles after it is written or when IVSEL is written. Setting the IVCE bit will disable interrupts, as explained in the IVSEL description above. See Code Example below

Two-wire Serial Interface

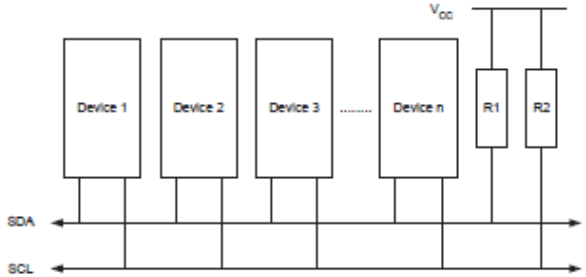
Features

- Simple Yet Powerful and Flexible Communication Interface, Only Two Bus Lines Needed
- Both Master and Slave Operation Supported
- Device Can Operate as Transmitter or Receiver
- 7-bit Address Space allows up to 128 Different Slave Addresses
- Multi-master Arbitration Support
- Up to 400 kHz Data Transfer Speed
- Slew-rate Limited Output Drivers
- Noise Suppression Circuitry Rejects Spikes on Bus Lines
- Fully Programmable Slave Address with General Call Support
- Address Recognition causes Wake-up when AVR is in Sleep Mode

Two-wire Serial Interface Bus Definition

The Two-wire Serial Interface (TWI) is ideally suited for typical microcontroller applications. The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the TWI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWI protocol.

Figure 76. TWI Bus Interconnection



TWI Terminology

The following definitions are frequently encountered in this section.

Table 72. TWI Terminology

Term	Description
Master	The device that initiates and terminates a transmission. The Master also generates the SCL clock.
Slave	The device addressed by a Master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.

Electrical Interconnection

As depicted in Figure 76, both bus lines are connected to the positive supply voltage through pull-up resistors. The bus drivers of all TWI-compliant devices are open-drain or open-collector. This implements a wired-AND function which is essential to the operation of the interface. A low level on a TWI bus line is generated when one or more TWI devices output a zero. A high level is output when all TWI devices tri-state their outputs, allowing the pull-up resistors to pull the line high. Note that all AVR devices connected to the TWI bus must be powered in order to allow any bus operation.

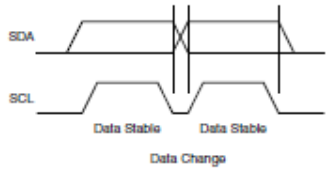
The number of devices that can be connected to the bus is only limited by the bus capacitance limit of 400 pF and the 7-bit Slave address space. A detailed specification of the electrical characteristics of the TWI is given in "Two-wire Serial Interface Characteristics" on page 294. Two different sets of specifications are presented there, one relevant for bus speeds below 100 kHz, and one valid for bus speeds up to 400 kHz.

Data Transfer and Frame Format

Transferring Bits

Each data bit transferred on the TWI bus is accompanied by a pulse on the clock line. The level of the data line must be stable when the clock line is high. The only exception to this rule is for generating start and stop conditions.

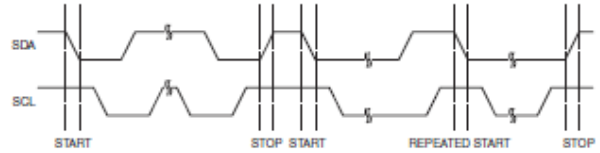
Figure 77. Data Validity



START and STOP Conditions

The Master initiates and terminates a data transmission. The transmission is initiated when the Master issues a START condition on the bus, and it is terminated when the Master issues a STOP condition. Between a START and a STOP condition, the bus is considered busy, and no other Master should try to seize control of the bus. A special case occurs when a new START condition is issued between a START and STOP condition. This is referred to as a REPEATED START condition, and is used when the Master wishes to initiate a new transfer without releasing control of the bus. After a REPEATED START, the bus is considered busy until the next STOP. This is identical to the START behavior, and therefore START is used to describe both START and REPEATED START for the remainder of this datasheet, unless otherwise noted. As depicted below, START and STOP conditions are signalled by changing the level of the SDA line when the SCL line is high.

Figure 78. START, REPEATED START, and STOP Conditions



Address Packet Format

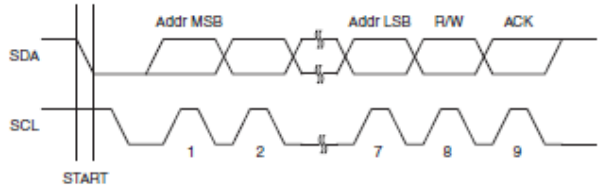
All address packets transmitted on the TWI bus are nine bits long, consisting of seven address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is to be performed, otherwise a write operation should be performed. When a Slave recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle. If the addressed Slave is busy, or for some other reason can not service the Master's request, the SDA line should be left high in the ACK clock cycle. The Master can then transmit a STOP condition, or a REPEATED START condition to initiate a new transmission. An address packet consisting of a Slave address and a READ or a WRITE bit is called SLA+R or SLA+W, respectively.

The MSB of the address byte is transmitted first. Slave addresses can freely be allocated by the designer, but the address 0000 000 is reserved for a general call.

When a general call is issued, all Slaves should respond by pulling the SDA line low in the ACK cycle. A general call is used when a Master wishes to transmit the same message to several Slaves in the system. When the general call address followed by a Write bit is transmitted on the bus, all Slaves set up to acknowledge the general call will pull the SDA line low in the ack cycle. The following data packets will then be received by all the Slaves that acknowledged the general call. Note that transmitting the general call address followed by a Read bit is meaningless, as this would cause contention if several Slaves started transmitting different data.

All addresses of the format 1111 xxx should be reserved for future purposes.

Figure 79. Address Packet Format



2-Way Hot Water and Steam

Direct Acting & Pilot Operated Valves
1/4" - 1 1/2" NPT

General Description:

2-Way Hot Water and Steam valves are specifically designed to withstand harsh application conditions. Many designs include integral stainless steel main and pilot orifices.

Installation
Valves should be mounted vertical and upright.

Standard Materials of Construction
Please refer to page D33.

Compatible Fluids
Ideal for the control of hot water and steam in a variety of applications.

Specialty

Speedy

Electrical Characteristics:

Standard Voltages:
AC –24/60
120/60–110/50
240/60–220/50
DC –12, 24 & 120
For other voltages – consult factory

Coil Classification:
Class F Standard
Class H Standard for media temperatures over 297°F

Note: Magnalatch coils are not available on steam valves.

Agency Approvals:
Standard valves with NEMA 4X or explosion psolenoid enclosures are UL Listed and CSA Certified. For additional details, consult factory.

Maximum Ambient Temperature
150° F

Please refer to Page D33 for details.

Applications:

- Industrial laundry machines
- Industrial dish washing machines
- Institutional cooking and food warming equipment
- Steam tables, steam cookers/kettles
- Sterilizers
- Dry cleaning equipment and steam irons
- Hospital equipment
- Steam presses
- Steam baths
- Autoclaves

Parker

ENGINEERING YOUR SUCCESS.

D

2-Way Hot Water and Steam - Normally Closed - Brass

Port Size NPT	Orifice Size in.	Flow Factor Cv	Operating Pressure Differential (MOPD) PSI Min.	Hot Water	Steam	Watt	Max. Media Temp. °F	Seal***	Pressure Vessel Number**	Reference Coil	Reference Valve
AC TECHNICAL SPECIFICATIONS											
1/4	13/64	0.76	0	100	-	10	210	EPDM	7121KBN2SE00	7	D21
1/4	13/64	0.76	0	-	40	10	285	EPDM	7121KBN2SES0	7	D21
1/4	7/16	2.00	3	150	-	10	210	EPDM	7321KBN2RE00	7	D22
1/4	7/16	2.00	3	-	45	10	293	EPDM	7321KBN2RES0	7	D22
3/8	7/16	2.50	3	150	-	10	210	EPDM	7321KBN3SE00	7	D22
3/8	5/8	3.00	0	150	50	11	300	EPDM	06FS3C2340ACF	4	D10
3/8	5/8	3.00	5	150	-	10	210	EPDM	7321BBN3TE00	7	D20
3/8	5/8	3.00	0	100	-	10	210	EPDM	7221BBN3TE00	7	D14
3/8	19/32	4.40	0	150	-	10	210	EPDM	7221GBN3VE00	7	D23
3/8	5/8	3.00	3	-	125	10	353	PTFE	7321BBN3TTS0*	7	D20
3/8	1/2	3.00	1	-	125	11	353	PTFE	06FS5C2432ACH*	4	D9
3/8	1/2	3.00	1	-	80	11	320	PTFE	06FS5C2432ACF	4	D9
3/8	5/8	3.00	0	-	50	10	297	EPDM	7221BBN3TES0	7	D14
3/8	5/8	3.00	5	-	50	10	297	EPDM	7321BBN3TES0	7	D20
3/8	1/2	3.00	1	-	50	11	300	EPDM	06FS5C2332ACF	4	D9
3/8	19/32	4.40	0	-	45	10	293	EPDM	7221GBN3VES0	7	D23
1/2	7/16	3.00	3	150	-	10	210	EPDM	7321KBN4SE00	7	D22
1/2	5/8	4.00	5	150	-	10	210	EPDM	7321BBN4UE00	7	D20
1/2	5/8	4.00	0	100	-	10	210	EPDM	7221BBN4UE00	7	D14
1/2	19/32	4.40	0	150	-	10	210	EPDM	7221GBN4VE00	7	D23
1/2	5/8	4.00	0	150	50	11	300	EPDM	08FS3C2340ACF	4	D10
1/2	1/2	3.60	1	-	125	11	353	PTFE	08FS5C2432ACH*	4	D9
1/2	1/2	3.60	1	-	80	11	320	PTFE	08FS5C2432ACF	4	D9
1/2	1/2	3.60	1	-	50	11	300	EPDM	08FS5C2332ACF	4	D9
1/2	7/16	3.00	3	-	45	10	293	EPDM	7321KBN4SES0	7	D22
1/2	5/8	4.00	3		125	10	353	PTFE	7321BBN4UTS0*	7	D20
1/2	5/8	4.00	0	-	50	10	297	EPDM	7221BBN4UES0	7	D14
1/2	5/8	4.00	5	-	50	10	297	EPDM	7321BBN4UES0	7	D20
1/2	19/32	4.40	0	-	45	10	293	EPDM	7221GBN4VES0	7	D23

* High pressure steam valves require Class 'H' coils only from referenced coil chart.
** Models xxS5xx have an integral stainless steel main orifice seat.
Models 7221Bxxx & 7221Gxxx are direct lift and will open at zero pressure differential but not at full flow.
*** Valves with EPDM (Ethylene Propylene) elastomers are limited to pressure range shown (45 or 50 psi) AND temperature rating shown (293-300° F). Do not use on higher pressure steam with pressure reducing valve; this may result in superheated steam.

Parker

Parker Hannifin Corporation
Fluid Control Division
1 800 825 8305 (1 800 Valve05)
www.parker.com/fcd

2-Way Hot Water and Steam - Normally Closed - Brass (Continued)

Port Size NPT	Orifice Size in.	Flow Factor Cv	Operating Pressure Differential (MOPD) PSI			Watt	Max. Media Temp. °F	Seal***	Pressure Vessel Number**	Reference	
			Min.	Hot Water	Steam					Coil	Valve
AC TECHNICAL SPECIFICATIONS											
3/4	3/4	5.00	5	150	-	10	210	EPDM	73218BN5VE00	7	D20
3/4	3/4	5.00	0	100	-	10	210	EPDM	72218BN5VE00	7	D14
3/4	19/32	5.50	0	150	-	10	210	EPDM	7221GBN51E00	7	D23
3/4	3/4	5.00	0	150	50	11	300	EPDM	12FS3C2348ACF	4	D11
3/4	5/8	4.50	3	-	125	10	353	PTFE	73218BN5VTS0*	7	D20
3/4	3/4	5.00	0	-	50	10	297	EPDM	72218BN5VES0	7	D14
3/4	3/4	5.00	5	-	50	10	297	EPDM	73218BN5VES0	7	D20
3/4	19/32	5.50	0	-	45	10	293	EPDM	7221GBN51ES0	7	D23
3/4	3/4	7.40	1	-	125	11	353	PTFE	12FS5C2448ACH*	4	D25
3/4	3/4	7.40	1	-	80	11	320	PTFE	12FS5C2448ACF	4	D25
3/4	3/4	7.40	1	-	50	11	300	EPDM	12FS5C2348ACF	4	D25
1	19/32	5.50	0	150	-	10	210	EPDM	7221GBN61E00	7	D23
1	1 1/16	13.50	5	125	-	10	210	EPDM	73218BN64E00	7	D16
1	1	11.70	0	150	-	10	210	EPDM	7221GBN64E00	7	D23
1	1	12.20	1	150	50	11	300	EPDM	16FS5C2364ACF	4	D12
1	19/32	5.50	0	-	45	10	293	EPDM	7221GBN61ES0	7	D23
1	1	11.70	0	-	45	10	293	EPDM	7221GBN64ES0	7	D23
1	1	12.20	1	-	125	11	353	PTFE	16FS5C2464ACH*	4	D12
1	1	12.20	1	-	80	11	320	PTFE	16FS5C2464ACF	4	D12
1	1 1/16	13.50	5	-	125	10	353	PTFE	73218BN64TS0*	7	D17
1	1 1/16	13.50	5	-	50	10	297	EPDM	73218BN64ES0	7	D16
1 1/4	1 1/8	15.00	5	125	-	10	210	EPDM	73218BN75E00	7	D16
1 1/4	1 1/8	15.00	5	150	50	6	300	EPDM	20FS4C2372AAF	1	D13
1 1/4	1 1/8	15.00	5	-	50	10	297	EPDM	73218BN75ES0	7	D16
1 1/4	1 1/8	16.00	5	-	125	10	353	PTFE	73218BN75TS0*	7	D17
1 1/2	1 1/4	22.50	5	125	-	10	210	EPDM	73218BN87E00	7	D19
1 1/2	1 1/2	22.50	5	150	50	6	300	EPDM	24FS4C2380AAF	1	D13
1 1/2	1 1/4	22.50	5	-	50	10	297	EPDM	73218BN87ES0	7	D19
1 1/2	1 1/4	22.50	5	-	125	10	353	PTFE	73218BN87TS0*	7	D19

* High pressure steam valves require Class 'H' coils only from referenced coil chart.
** Models xxS5xx have an integral stainless steel main orifice seat.
Models 7221Bxxx & 7221Gxxx are direct lift and will open at zero pressure differential but not at full flow.
*** Valves with EPDM (Ethylene Propylene) elastomers are limited to pressure range shown (45 or 50 psi) AND temperature rating shown (293-300° F). Do not use on higher pressure steam with pressure reducing valve; this may result in superheated steam.

Parker

Specialty

2-Way Hot Water and Steam - Normally Closed - Brass (Continued)

Port Size NPT	Orifice Size in.	Flow Factor Cv	Operating Pressure Differential (MOPD) PSI			Watt	Max. Media Temp. °F	Seal***	Pressure Vessel Number**	Reference	
			Min.	Hot Water	Steam					Coil	Valve
DC TECHNICAL SPECIFICATIONS											
1/4	13/64	0.76	0	40	-	10	210	EPDM	7121KBN2SE00	7	D21
1/4	13/64	0.76	0	100	-	22	210	EPDM	7121KBN2SE00	8	D21
1/4	7/16	2.00	3	150	-	22	210	EPDM	7321KBN2RE00	8	D22
1/4	7/16	2.00	3	60	-	10	210	EPDM	7321KBN2RE00	7	D22
1/4	13/64	0.76	0	-	40	22	285	EPDM	7121KBN2SES0	8	D21
1/4	7/16	2.00	3	-	45	22	293	EPDM	7321KBN2RES0	8	D22
3/8	7/16	2.50	3	150	-	22	210	EPDM	7321KBN3SE00	8	D22
3/8	7/16	2.50	3	60	-	10	210	EPDM	7321KBN3SE00	7	D22
3/8	5/8	3.00	0	40	-	22	210	EPDM	72218BN3TE00	8	D14
3/8	5/8	3.00	0	100	-	11.5	150	EPDM	06F22C2340A3F	6	D10
3/8	5/8	3.00	5	150	-	10	210	EPDM	73218BN3TE00	7	D20
3/8	5/8	3.00	5	40	-	11.5	150	EPDM	06F23C2340A3F	6	D10
3/8	19/32	4.40	0	100	-	22	210	EPDM	7221GBN3VE00	8	D23
3/8	19/32	4.40	0	-	45	22	293	EPDM	7221GBN3VES0	8	D23
1/2	7/16	3.00	3	150	-	22	210	EPDM	7321KBN4SE00	8	D22
1/2	7/16	3.00	3	60	-	10	210	EPDM	7321KBN4SE00	7	D22
1/2	5/8	4.00	5	150	-	10	210	EPDM	73218BN4UE00	7	D20
1/2	5/8	4.00	0	100	-	11.5	150	EPDM	06F22C2340A3F	6	D10
1/2	5/8	4.00	5	40	-	11.5	150	EPDM	06F23C2340A3F	6	D10
1/2	5/8	4.00	0	40	-	22	210	EPDM	72218BN4UE00	8	D14
1/2	19/32	4.40	0	100	-	22	210	EPDM	7221GBN4VE00	8	D23
1/2	7/16	3.00	3	-	45	22	293	EPDM	7321KBN4SES0	8	D22
1/2	19/32	4.40	0	-	45	22	293	EPDM	7221GBN4VES0	8	D23
3/4	3/4	5.00	5	150	-	10	210	EPDM	73218BN5VE00	7	D20
3/4	3/4	5.00	0	100	-	11.5	150	EPDM	12F22C2348A3F	6	D11
3/4	3/4	5.00	5	40	-	11.5	150	EPDM	12F23C2348A3F	6	D11
3/4	3/4	5.00	0	40	-	22	210	EPDM	72218BN5VE00	8	D14
3/4	19/32	5.50	0	100	-	22	210	EPDM	7221GBN51E00	8	D23
3/4	19/32	5.50	0	-	45	22	293	EPDM	7221GBN51ES0	8	D23

* High pressure steam valves require Class 'H' coils only from referenced coil chart.
** Models xxS5xx have an integral stainless steel main orifice seat.
Models 7221Bxxx & 7221Gxxx are direct lift and will open at zero pressure differential but not at full flow.
*** Valves with EPDM (Ethylene Propylene) elastomers are limited to pressure range shown (45 or 50 psi) AND temperature rating shown (293-300° F). Do not use on higher pressure steam with pressure reducing valve; this may result in superheated steam.

2-Way Hot Water and Steam - Normally Closed - Brass (Continued)

Port Size NPT	Orifice Size in.	Flow Factor Cv	Operating Pressure Differential (MOPD) PSI			Watt	Max. Media Temp. °F	Seal***	Pressure Vessel Number**	Reference	
			Min.	Hot Water	Steam					Coil	Valve
DC TECHNICAL SPECIFICATIONS											
1	19/32	5.50	0	100	-	22	210	EPDM	7221GBN61E00	8	D23
1	1	11.70	0	100	-	22	210	EPDM	7221GBN64E00	8	D23
1	1 1/16	13.50	5	125	-	10	210	EPDM	73218BN64E00	7	D16
1	19/32	5.50	0	-	45	22	293	EPDM	7221GBN61ES0	8	D23
1	1	11.70	0	-	45	22	293	EPDM	7221GBN64ES0	8	D23
1 1/4	1 1/8	15.00	5	125	-	10	210	EPDM	73218BN75E00	7	D16
1 1/2	1 1/4	22.50	5	125	-	10	210	EPDM	73218BN87E00	7	D19

2-Way Hot Water and Steam - Normally Closed - Stainless Steel

Port Size NPT	Orifice Size in.	Flow Factor Cv	Operating Pressure Differential (MOPD) PSI			Watt	Max. Media Temp. °F	Seal***	Pressure Vessel Number**	Reference	
			Min.	Hot Water	Steam					Coil	Valve
AC TECHNICAL SPECIFICATIONS											
1/4	5/32	0.52	0	-	110	11	344	PTFE	04FS0C3410ACH*	4	D24
3/8	5/8	3.00	0	100	-	10	210	EPDM	72218RN3TE00	7	D14
1/2	5/8	4.00	0	100	-	10	210	EPDM	72218RN4UE00	7	D14
1/2	5/8	4.00	0	-	50	10	297	EPDM	72218RN4UES0	7	D14
3/4	3/4	5.00	0	100	-	10	210	EPDM	72218RN5VE00	7	D14
3/4	3/4	5.00	0	-	50	10	297	EPDM	72218RN5VES0	7	D14
DC TECHNICAL SPECIFICATIONS											
3/8	5/8	3.00	0	40	-	22	210	EPDM	72218RN3TE00	8	D14
1/2	5/8	4.00	0	40	-	22	210	EPDM	72218RN4UE00	8	D14
3/4	3/4	5.00	0	40	-	22	210	EPDM	72218RN5VE00	8	D14

* High pressure steam valves require Class 'H' coils only from referenced coil chart.
** Models xxS5xx have an integral stainless steel main orifice seat.
Models 72218xxx & 7221Gxxx are direct lift and will open at zero pressure differential but not at full flow.
*** Valves with EPDM (Ethylene Propylene) elastomers are limited to pressure range shown (45 or 50 psi) AND temperature rating shown (293-300° F). Do not use on higher pressure steam with pressure reducing valve; this may result in superheated steam.

Specialty Appods

2-Way Hot Water and Steam - Normally Open - Brass

Port Size NPT	Orifice Size in.	Flow Factor Cv	Operating Pressure Differential (MOPD) PSI			Watt	Max. Media Temp. °F	Seal***	Pressure Vessel Number**	Reference	
			Min.	Hot Water	Steam					Coil	Valve
AC TECHNICAL SPECIFICATIONS											
3/8	5/8	3.0	0	125	-	22	210	EPDM	72228BN3TE00	8	D18
3/8	5/8	3.0	5	-	125	10	353	PTFE	73228BN3TTS0	7	D15
3/8	5/8	3.0	0	-	50	10	297	EPDM	72228BN3TES0	7	D18
1/2	5/8	4.0	0	125	-	22	210	EPDM	72228BN4UE00	8	D18
1/2	1/2	3.6	1	-	125	11	353	PTFE	08FS5O2432ACH*	4	D9
1/2	5/8	4.0	5	-	125	10	353	PTFE	73228BN4UTS0*	7	D15
1/2	5/8	4.0	0	-	50	10	297	EPDM	72228BN4UES0	7	D18
3/4	3/4	5.0	0	125	-	22	210	EPDM	72228BN5VE00	8	D18
3/4	3/4	5.0	0	-	50	10	297	EPDM	72228BN5VES0	7	D18
3/4	3/4	7.5	5	-	125	10	353	PTFE	73228BN52TS0*	7	D15
3/4	3/4	7.4	1	-	125	11	353	PTFE	12FS5O2448ACH*	4	D25
1	1 1/16	13.5	5	-	125	10	353	PTFE	73228BN64TS0*	7	D17
1 1/4	1 1/8	16.0	5	-	125	10	353	PTFE	73228BN75TS0*	7	D17
1 1/2	1 1/4	22.5	5	-	125	10	353	PTFE	73228BN87TS0*	7	D19
DC TECHNICAL SPECIFICATIONS											
3/8	5/8	3.0	0	125	-	22	210	EPDM	72228BN3TE00	8	D18
1/2	5/8	4.0	0	125	-	22	210	EPDM	72228BN4UE00	8	D18
3/4	3/4	5.0	0	125	-	22	210	EPDM	72228BN5VE00	8	D18

* High pressure steam valves require Class 'H' coils only from referenced coil chart.
** Models xxS5xx have an integral stainless steel main orifice seat.
Models 72218xxx & 7221Gxxx are direct lift and will open at zero pressure differential but not at full flow.
*** Valves with EPDM (Ethylene Propylene) elastomers are limited to pressure range shown (45 or 50 psi) AND temperature rating shown (293-300° F). Do not use on higher pressure steam with pressure reducing valve; this may result in superheated steam.

2-Way Hot Water and Steam - Normally Open - Stainless Steel

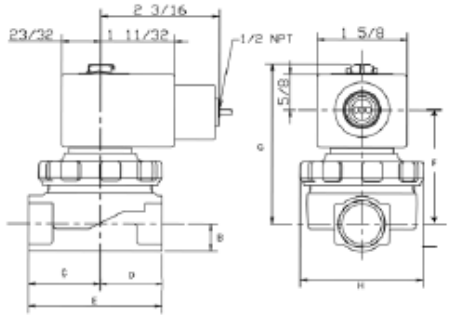
Port Size NPT	Orifice Size in.	Flow Factor Cv	Operating Pressure Differential (MOPD) PSI			Watt	Max. Media Temp. °F	Seal	Pressure Vessel Number*	Reference	
			Min.	Hot Water	Steam					Coil	Valve
AC TECHNICAL SPECIFICATIONS											
3/8	5/8	3.0	0	125	-	22	210	EPDM	72226RN3TE00	8	D18
1/2	5/8	4.0	0	125	-	22	210	EPDM	72226RN4UE00	8	D18
3/4	3/4	5.0	0	125	-	22	210	EPDM	72226RN5VE00	8	D18
DC TECHNICAL SPECIFICATIONS											
3/8	5/8	3.0	0	125	-	22	210	EPDM	72226RN3TE00	8	D18
1/2	5/8	4.0	0	125	-	22	210	EPDM	72226RN4UE00	8	D18
3/4	3/4	5.0	0	125	-	22	210	EPDM	72226RN5VE00	8	D18

*Models 72228xxx is a direct lift and will open at zero pressure differential but not at full flow.

Valve Reference D9

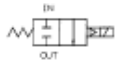
2-Way Normally Closed
06FS5Cxx, 08FS5Cxx

2-Way Normally Open
08FS5Oxx
Port Identification:
In-In/Out-Out

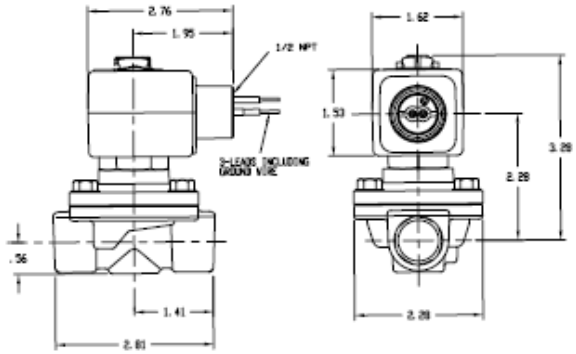


Valve	Dimensions						
	B	C	D	E	F	G	H
06FS5C2xxx 08FS5C2xxx	1/2	15/16	1 1/8	2 7/16	2 5/16	2 29/32	2 1/4
08FS5O2xxx	1/2	15/16	1 1/8	2 7/16	2 7/32	2 29/32	2 1/4

Valve Reference D10



2-Way Normally Closed
Port Identification:
In-In/Out-Out

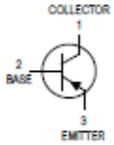


MOTOROLA
SEMICONDUCTOR TECHNICAL DATA

Order this document by BC556/D

Amplifier Transistors
PNP Silicon

BC556,B
BC557A,B,C
BC558B



CASE 29–04, STYLE 17
TO–92 (TO–226AA)

MAXIMUM RATINGS

Rating	Symbol	BC 556	BC 557	BC 558	Unit
Collector–Emitter Voltage	V_{CEO}	–65	–45	–30	Vdc
Collector–Base Voltage	V_{CBO}	–80	–50	–30	Vdc
Emitter–Base Voltage	V_{EBO}		–5.0		Vdc
Collector Current — Continuous	I_C		–100		mAdc
Total Device Dissipation @ T_A = 25°C Derate above 25°C	P_D		625 5.0		mW mW/°C
Total Device Dissipation @ T_C = 25°C Derate above 25°C	P_D		1.5 12		Watt mW/°C
Operating and Storage Junction Temperature Range	T_J, T_{stg}		–55 to +150		°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Max	Unit
Thermal Resistance, Junction to Ambient	$R_{\theta JA}$	200	°C/W
Thermal Resistance, Junction to Case	$R_{\theta JC}$	83.3	°C/W

ELECTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted)

Characteristic		Symbol	Min	Typ	Max	Unit
OFF CHARACTERISTICS						
Collector–Emitter Breakdown Voltage (I_C = –2.0 mAdc, I_B = 0)	BC556	$V_{(BR)CEO}$	–65	—	—	V
	BC557		–45	—	—	
	BC558		–30	—	—	
Collector–Base Breakdown Voltage (I_C = –100 µAdc)	BC556	$V_{(BR)CBO}$	–80	—	—	V
	BC557		–50	—	—	
	BC558		–30	—	—	
Emitter–Base Breakdown Voltage (I_E = –100 µAdc, I_C = 0)	BC556	$V_{(BR)EBO}$	–5.0	—	—	V
	BC557		–5.0	—	—	
	BC558		–5.0	—	—	
Collector–Emitter Leakage Current (V_{CES} = –40 V)	BC556	I_{CES}	—	–2.0	–100	nA
(V_{CES} = –20 V)	BC557		—	–2.0	–100	
	BC558		—	–2.0	–100	
(V_{CES} = –20 V, T_A = 125°C)	BC556		—	—	–4.0	µA
	BC557		—	—	–4.0	
	BC558		—	—	–4.0	

BC556,B BC557A,B,C BC558B

ELECTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted) (Continued)

Characteristic		Symbol	Min	Typ	Max	Unit
ON CHARACTERISTICS						
DC Current Gain (I_C = –10 µAdc, V_{CE} = –5.0 V)	BC557A	h_{FE}	—	90	—	—
	BC556B/557B/558B		—	150	—	
	BC557C		—	270	—	
(I_C = –2.0 mAdc, V_{CE} = –5.0 V)	BC556		120	—	500	
	BC557		120	—	800	
	BC558		120	—	800	
	BC557A		120	170	220	
	BC556B/557B/558B		180	290	460	
	BC557C		420	500	800	
(I_C = –100 mAdc, V_{CE} = –5.0 V)	BC557A		—	120	—	
	BC556B/557B/558B		—	180	—	
	BC557C		—	300	—	
Collector–Emitter Saturation Voltage (I_C = –10 mAdc, I_B = –0.5 mAdc)		$V_{CE(sat)}$	—	–0.075	–0.3	V
(I_C = –10 mAdc, I_B = see Note 1)			—	–0.3	–0.6	
(I_C = –100 mAdc, I_B = –5.0 mAdc)			—	–0.25	–0.65	
Base–Emitter Saturation Voltage (I_C = –10 mAdc, I_B = –0.5 mAdc)		$V_{BE(sat)}$	—	–0.7	—	V
(I_C = –100 mAdc, I_B = –5.0 mAdc)			—	–1.0	—	
Base–Emitter On Voltage (I_C = –2.0 mAdc, V_{CE} = –5.0 Vdc)		$V_{BE(on)}$	–0.55	–0.62	–0.7	V
(I_C = –10 mAdc, V_{CE} = –5.0 Vdc)			—	–0.7	–0.82	
SMALL–SIGNAL CHARACTERISTICS						
Current–Gain — Bandwidth Product (I_C = –10 mA, V_{CE} = –5.0 V, f = 100 MHz)	BC556	f_T	—	280	—	MHz
	BC557		—	320	—	
	BC558		—	360	—	
Output Capacitance (V_{CB} = –10 V, I_C = 0, f = 1.0 MHz)		C_{ob}	—	3.0	6.0	pF
Noise Figure (I_C = –0.2 mAdc, V_{CE} = –5.0 V,	BC556	NF	—	2.0	10	dB
R_S = 2.0 kΩ, f = 1.0 kHz, Δf = 200 Hz)	BC557		—	2.0	10	
	BC558		—	2.0	10	
Small–Signal Current Gain (I_C = –2.0 mAdc, V_{CE} = –5.0 V, f = 1.0 kHz)	BC556	h_{fe}	125	—	500	—
	BC557/558		125	—	900	
	BC557A		125	220	260	
	BC556B/557B/558B		240	330	500	
	BC557C		450	600	900	

Note 1: I_C = –10 mAdc on the constant base current characteristics, which yields the point I_C = –11 mAdc, V_{CE} = –1.0 V.

SONGLE RELAY

RELAY ISO9002	SRD

1. MAIN FEATURES

- ☐ Switching capacity available by 10A in spite of small size design for highdensity P.C. board mounting technique.
- ☐ UL,CUL,TUV recognized.
- ☐ Selection of plastic material for high temperature and better chemical solution performance.
  - ☐ Sealed types available.
- ☐ Simple relay magnetic circuit to meet low cost of mass production.

2. APPLICATIONS

☐ Domestic appliance, office machine, audio, equipment, automobile, etc.
(Remote control TV receiver, monitor display, audio equipment high rushing current use application.)

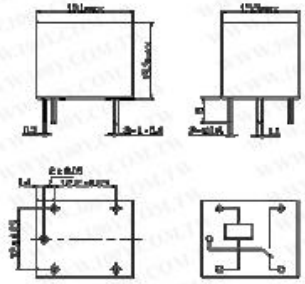
3. ORDERING INFORMATION

SRD	XX VDC	S	L	C
Model of relay	Nominal coil voltage	Structure	Coil	Contact form
SRD	03|05|06|09|12|24|48VDC	S:Sealed type	L:0.36W	A:1 form A
		F:Flux free type	D:0.45W	B:1 form B C:1 form C

4. RATING

CCC	FILE NUMBER:CQC03001003729	7A/240VDC
CCC	FILE NUMBER:CQC03001003731	10A/250VDC
UL /CUL	FILE NUMBER: E167996	10A/125VAC 28VDC
TUV	FILE NUMBER: R50056114	10A/250VAC 30VDC

5. DIMENSION(unit:mm) DRILLING(unit:mm) WIRING DIAGRAM



6. COIL DATA CHART (AT20 ° C)

Coil Sensitivity	Coil Voltage Code	Nominal Voltage (VDC)	Nominal Current (mA)	Coil Resistance (Ω) ☐ 10%	Power Consumption (W)	Pull-In Voltage (VDC)	Drop-Out Voltage (VDC)	Max-Allowable Voltage (VDC)
SRD (High Sensitivity)	03	03	120	25	abt. 0.36W	75%Max.	10% Min.	130%
	05	05	71.4	70				
	06	06	60	100				
	09	09	40	225				
	12	12	30	400				
	24	24	15	1600				
	48	48	7.5	6400				
SRD (Standard)	03	03	150	20	abt. 0.45W	75% Max.	10% Min.	110%
	05	05	89.3	55				
	06	06	75	80				
	09	09	50	180				
	12	12	37.5	320				
	24	24	18.7	1280				
	48	48	10	4500	abt. 0.51W			

7. CONTACT RATING

Item \ Type	SRD FORM C	SRD FORM A
Contact Capacity	7A 30VDC	10A 30VDC
Resistive Load (cosΦ=1)	10A 125VAC 10A 250VAC	10A 240VAC
Inductive Load (cosΦ=0.4 L/R=7msec)	3A 120VAC 3A 28VDC	5A 120VAC 5A 28VDC
Max. Allowable Voltage	250VAC/110VDC	250VAC/110VDC
Max. Allowable Power Force	2500VA/210W	1200VA/300W
Contact Material	AgCdO	AgCdO

8. PERFORMANCE (at Initial value)

Item \ Type	SRD
Contact Resistance	100mΩ Max.
Operation Time	10msec Max.
Release Time	5msec Max.
Dielectric Strength	
Between coil & contact	1500VAC 50/60HZ (1 minute)
Between contacts	1000VAC 50/60HZ (1 minute)
Insulation Resistance	100 MΩ Min. (500VDC)
Max. ON/OFF Switching	
Mechanically	300 operation/min
Electrically	30 operation/min
Ambient Temperature	-25℃ to +70 C
Operating Humidity	45 to 85% RH
Vibration	
Endurance	10 to 55Hz Double Amplitude 1.5mm
Error Operation	10 to 55Hz Double Amplitude 1.5mm
Shock	
Endurance	100G Min.
Error Operation	100G Min.
Life Expectancy	
Mechanically	10^7 operations. Min. (no load)
Electrically	10^5 operations. Min. (at rated coil voltage)
Weight	abt. 10grs.

9.REFERENCE DATA