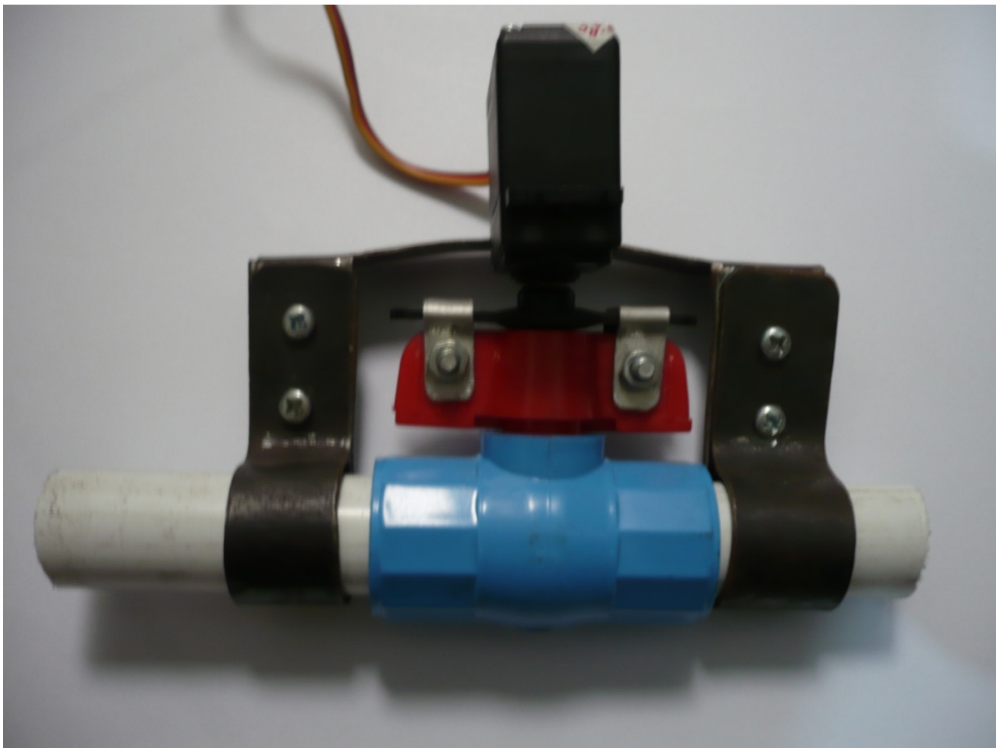
# LAMPIRAN A

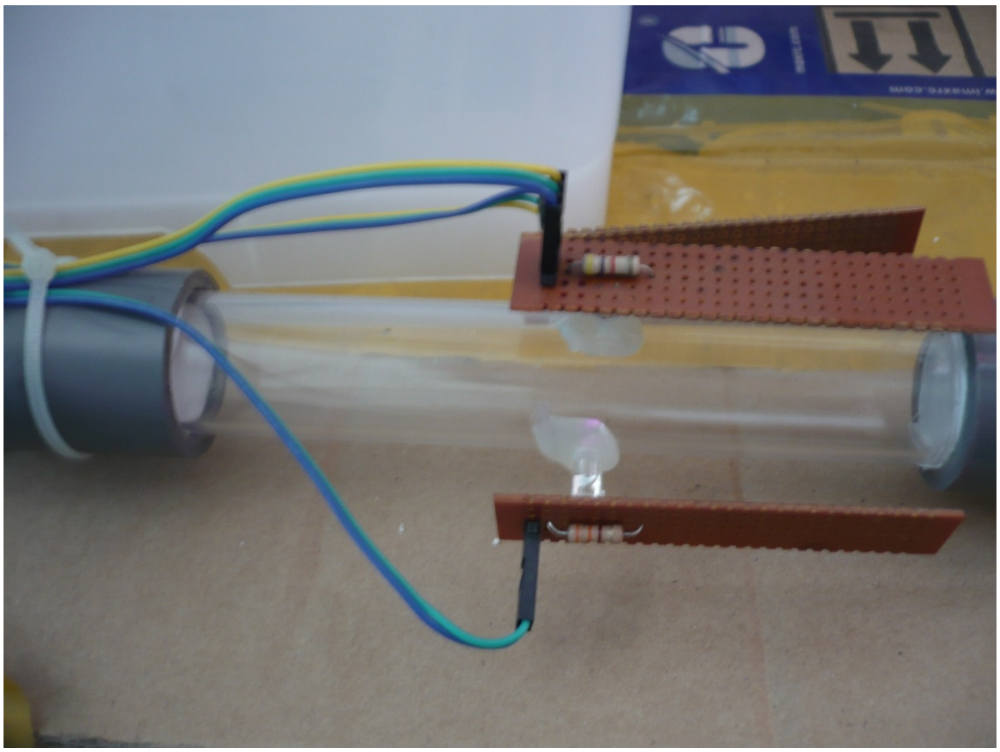
# FOTO SISTEM PENYARINGAN AIR

## SISTEM PENYARINGAN

**VALVE**



**SENSOR KEKERUHAN**



**SAMPLE AIR**

# LAMPIRAN B

# PROGRAM PADA PENGONTROL MIKRO ATMEGA16

```
/***************************************************
This program was produced by the
CodeWizardAVR V1.25.3 Standard
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 15/07/2011
Author  : F4CG
Company : F4CG
Comments:


Chip type           : ATmega16
Program type        : Application
Clock frequency     : 11,059200 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
***************************************************/
```

```
#include <mega16.h>
#include <delay.h>
#include <stdio.h>
#include <math.h>

unsigned int sensor_kekeruhan_air_1_90,
             sensor_kekeruhan_air_1_180,
             sensor_kekeruhan_air_2_90,
             sensor_kekeruhan_air_2_180,
             sensor_kekeruhan_air_3_90,
             sensor_kekeruhan_air_3_180,
             valve;
float tegangan_sensor_kekeruhan_air_1_90,
      tegangan_sensor_kekeruhan_air_1_180,
      tegangan_sensor_kekeruhan_air_2_90,
      tegangan_sensor_kekeruhan_air_2_180,
      tegangan_sensor_kekeruhan_air_3_90,
      tegangan_sensor_kekeruhan_air_3_180;
unsigned char text[32];

// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
```

```
#define ADC_VREF_TYPE 0x40

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
// Declare your global variables here
void buka()
{
    for(a=0;a<=100;a++)
    {
    if (valve==1)     PORTB.1=1; //valve 1
    if (valve==2)     PORTB.0=1; //valve 2
    if (valve==3)     PORTB.3=1; //valve 3
    if (valve==4)     PORTB.2=1; //valve 4
    if (valve==5)     PORTB.4=1; //valve 5
    delay_us(1800);
    PORTB.1=0;
    PORTB.0=0;
    PORTB.3=0;
    PORTB.2=0;
    PORTB.4=0;
    delay_ms(18);
    }
}
```

```
void tutup()
{
for(a=0;a<=100;a++)
      {
      if (valve==1)      PORTB.1=1;
      if (valve==2)      PORTB.0=1;
      if (valve==3)      PORTB.3=1;
      if (valve==4)      PORTB.2=1;
      if (valve==5)      PORTB.4=1;
      delay_us(500);
      PORTB.1=0;
      PORTB.0=0;
      PORTB.3=0;
      PORTB.2=0;
      PORTB.4=0;
      delay_ms(19);
      }
}
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=P State6=P State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0xC0;
DDRA=0x00;
```

```
// Port B initialization
// Func7=In Func6=In Func5=In Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=P State6=P State5=T State4=0 State3=0 State2=0 State1=0 State0=0
PORTB=0xC0;
DDRB=0x1F;

// 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=0x00;
DDRC=0x00;

// Port D 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
PORTD=0x00;
DDRD=0x00;

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

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 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: Timer 2 Stopped
// Mode: Normal top=FFh
// 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;

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

// ADC initialization
// ADC Clock frequency: 691,200 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

// LCD module initialization
lcd_init(16);
while (1)
    {
    //switch
    //A.6=sensor setengah penuh 2
    //A.7=sensor penuh 2
    //B.6=sensor setengah penuh 1
    //B.7=sensor penuh 1
```

```
// Place your code here
sensor_kekeruhan_air_1_90=read_adc(0); //sensor 1 90
delay_ms(10);
sensor_kekeruhan_air_1_180=read_adc(1); //sensor 1 180
delay_ms(10);
sensor_kekeruhan_air_2_90=read_adc(2); //sensor 2 90
delay_ms(10);
sensor_kekeruhan_air_2_180=read_adc(3);//sensor 2 180
delay_ms(10);
sensor_kekeruhan_air_3_90=read_adc(4);//sensor 3 90
delay_ms(10);
sensor_kekeruhan_air_3_180=read_adc(5); //sensor 3 180
delay_ms(10);
lcd_clear();

//ubah nilai adc menjadi tegangan
tegangan_sensor_kekeruhan_air_1_90=(sensor_kekeruhan_air_1_90*5)/1024;
tegangan_sensor_kekeruhan_air_1_180=(sensor_kekeruhan_air_1_180*5)/1024;
tegangan_sensor_kekeruhan_air_2_90=(sensor_kekeruhan_air_2_90*5)/1024;
tegangan_sensor_kekeruhan_air_2_180=(sensor_kekeruhan_air_2_180*5)/1024;
tegangan_sensor_kekeruhan_air_3_90=(sensor_kekeruhan_air_3_90*5)/1024;
tegangan_sensor_kekeruhan_air_3_180=(sensor_kekeruhan_air_3_180*5)/1024;
sprintf(text,"%4d %4d %4d %4d %4d %4d",
        tegangan_sensor_kekeruhan_air_1_90,
        tegangan_sensor_kekeruhan_air_1_180,
        tegangan_sensor_kekeruhan_air_2_90,
        tegangan_sensor_kekeruhan_air_2_180,
        tegangan_sensor_kekeruhan_air_3_90,
        tegangan_sensor_kekeruhan_air_3_180);  //tampilin nilai tegangan di
    lcd     dari tiap sensor....
lcd_puts(text);
```

```
//*************************************************************
*************************************************************
    if(sensor_kekeruhan_air_1_90<=984&&sensor_kekeruhan_air_1_180>=473)
//cek kotor
    {
    valve=1;  //bila kotor tutup valve 1 dan 4 , 2 buka
    tutup();
   valve=4;
    tutup();
    valve=2;
    buka();
    delay_ms(1000);

//selama bersih buka keran terus---------------------------------------------------------
 while(sensor_kekeruhan_air_2_90<=953&&sensor_kekeruhan_air_2_180>=479)
 {
 sensor_kekeruhan_air_2_90=read_adc(2); // sensor kekeruhan air 2 90
 delay_ms(10);
 sensor_kekeruhan_air_2_180=read_adc(3);// sensor kekeruhan air 2 180
 delay_ms(10);
 lcd_clear();
 sprintf(text,"%d%d",sensor_kekeruhan_air_2_90,sensor_kekeruhan_air_2_180);
 lcd_puts(text);
            for(a=0;a<=100;a++)
            {
            PORTB.3=1;
            delay_us(1800);
            PORTB.3=0;
            delay_ms(18);
            }
     valve=3;
      buka();
```

```
/////////////////////////////////////////////////////////////////////////////////////////// posisi level air
    if(PINB.7==0 && PINB.6==1) //valve 2 buka setengah
    {
           for(a=0;a<=100;a++)
           {
           PORTB.0=1;
           delay_us(1300); //buka valve 2 setengah
           PORTB.0=0;
           delay_ms(18);
           }
    }
    if(PINB.7==1 && PINB.6==1)  //valve 2 tutup
    {
           for(a=0;a<=100;a++)
           {
           PORTB.0=1;
           delay_us(500); //tutup  valve 2
           PORTB.0=0;
           delay_ms(19);
           }
    }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    }
//-------------------------------------------------------------------------------------------------
    valve=3;
    tutup();
    valve=1;
    tutup();
    valve=2;
    tutup();
    valve=4;
    buka();//ganti saringan kedua jika sensor 3 kotor
```

```
    //selama bersih buka keran terus.........................................................................

while(sensor_kekeruhan_air_3_90<=973&&sensor_kekeruhan_air_3_180>=465)
{
sensor_kekeruhan_air_3_90=read_adc(4);// sensor kekeruhan air 3 90
delay_ms(10);
sensor_kekeruhan_air_3_180=read_adc(5); // sensor kekeruhan air 3 180
delay_ms(10);
lcd_clear();
sprintf(text,"%d %d  ",sensor_kekeruhan_air_3_90,sensor_kekeruhan_air_3_180);
lcd_puts(text);

        for(a=0;a<=100;a++)
        {
        PORTB.4=1;
        delay_us(1800);
        PORTB.4=0;
        delay_ms(18);
        }
    valve=5;
    buka();
    if(PINA.7==0 && PINA.6==1)  // valve 4 buka setengah
    {
        for(a=0;a<=100;a++)
        {
        PORTB.2=1;
        delay_us(1300);
        PORTB.2=0;
        delay_ms(18);
        }
    }
```

```
    if(PINA.6==1 && PINA.7==1)  //valve 4 tutup
    {
        for(a=0;a<=100;a++)
        {
        PORTB.2=1;
        delay_us(500);
        PORTB.2=0;
        delay_ms(19);
        }
    }

    }
//...................................................................................................................


    lcd_clear();
    sprintf(text,"SEMUA SARINGAN KOTOR");
    lcd_puts(text);
    delay_ms(100000);
    }
//****************************************************************
****************************************************************
```

```
else
{
valve=1;
buka();
valve=2;
tutup();
valve=4;
tutup();
valve=3;
tutup();
valve=5;
tutup();
}

 };
}
```

# LAMPIRAN C
# 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

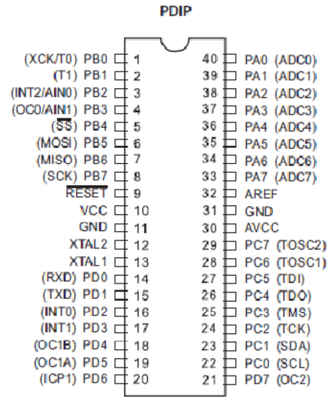
# 8-bit AVR® Microcontroller with 16K Bytes In-System Programmable Flash

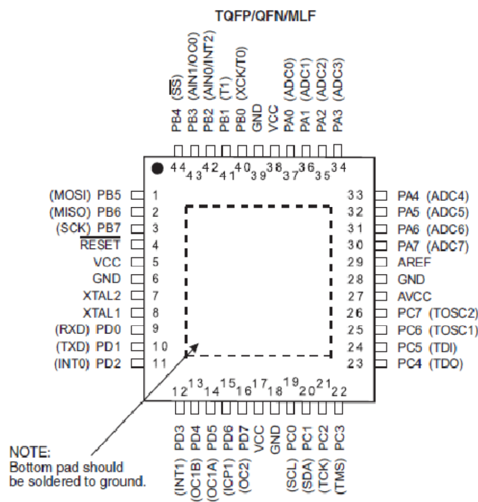
# ATmega16 ATmega16L

Rev. 2466T–AVR–07/10

## Pin Configurations

**Figure 1.** Pinout ATmega16




## Disclaimer

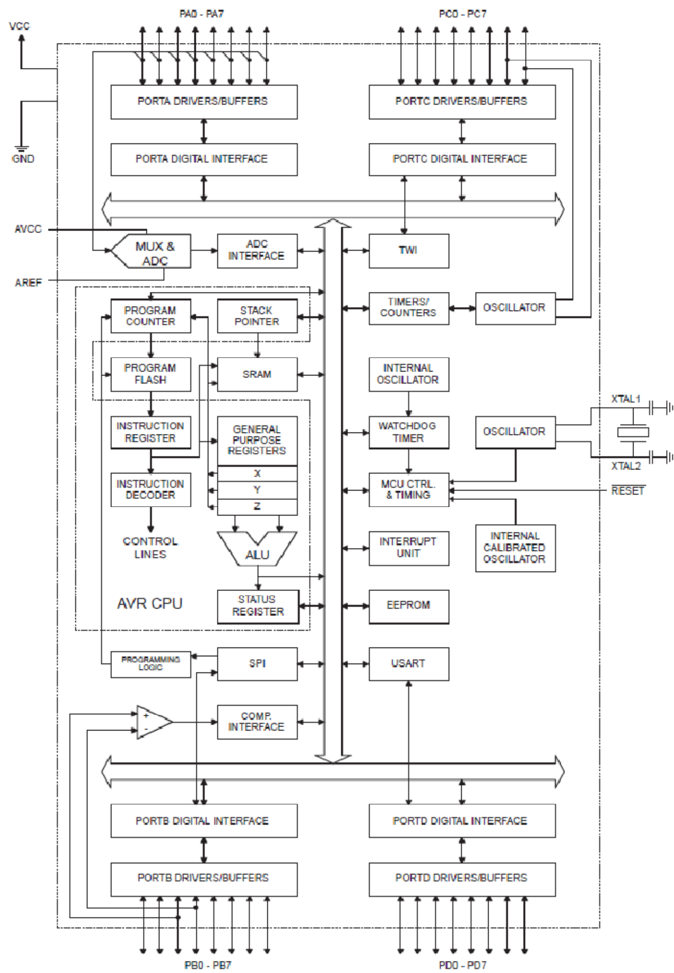
Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

## 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.

# TowerPro MG995 - Standard Servo

### Basic Information

| | |
|---|---|
| Modulation: | Analog |
| Torque: | **4.8V:** 138.9 oz-in (10.00 kg-cm) |
| Speed: | **4.8V:** 0.20 sec/60° |
| Weight: | 1.94 oz (55.0 g) |
| Dimensions: | Length: 1.60 in (40.6 mm)<br>Width: 0.78 in (19.8 mm)<br>Height: 1.69 in (42.9 mm) |
| Motor Type: | Coreless |
| Gear Type: | Metal |
| Rotation/Support: | Dual Bearings |