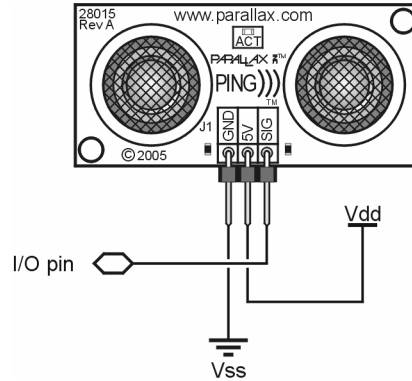




## Connection to a Microcontroller

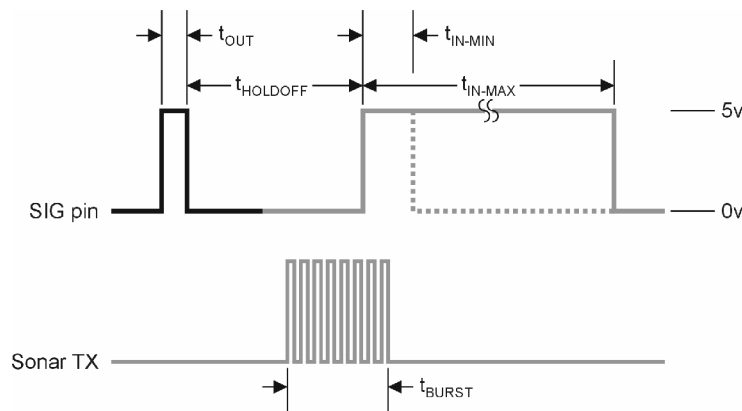
The PING))) sensor has a male 3-pin header used to supply power (5 vdc), ground, and signal. The header allows the sensor to be plugged into a solderless breadboard, or to be located remotely through the use of a standard servo extender cable. Standard connections are shown in the diagram below:



## Theory of Operation

The Ping sensor detects objects by emitting a short ultrasonic burst and then "listening" for the echo. Under control of a host microcontroller (trigger pulse), the sensor emits a short 40 kHz (ultrasonic) burst. This burst travels through the air at about 1130 feet per second, hits an object and then bounces back to the sensor. The PING))) sensor provides an output pulse to the host that will terminate when the echo is detected, hence the width of this pulse corresponds to the distance to the target.

### PING))) Sensor Timing



— HOST	$t_{OUT}$	2 $\mu$ S (min), 5 $\mu$ S typical
— PING	$t_{HOLDOFF}$	350 $\mu$ S
	$t_{BURST}$	200 $\mu$ S @ 40 kHz
	$t_{IN-MIN}$	115 $\mu$ S
	$t_{IN-MAX}$	18.5 mS

## Program Example: BASIC Stamp 2 Microcontroller

The following program demonstrates the use of the PING))) sensor with the BASIC Stamp 2 microcontroller. Any model of BASIC Stamp 2 module will work with this program as conditional compilation techniques are used to make adjustments based on the module that is connected.

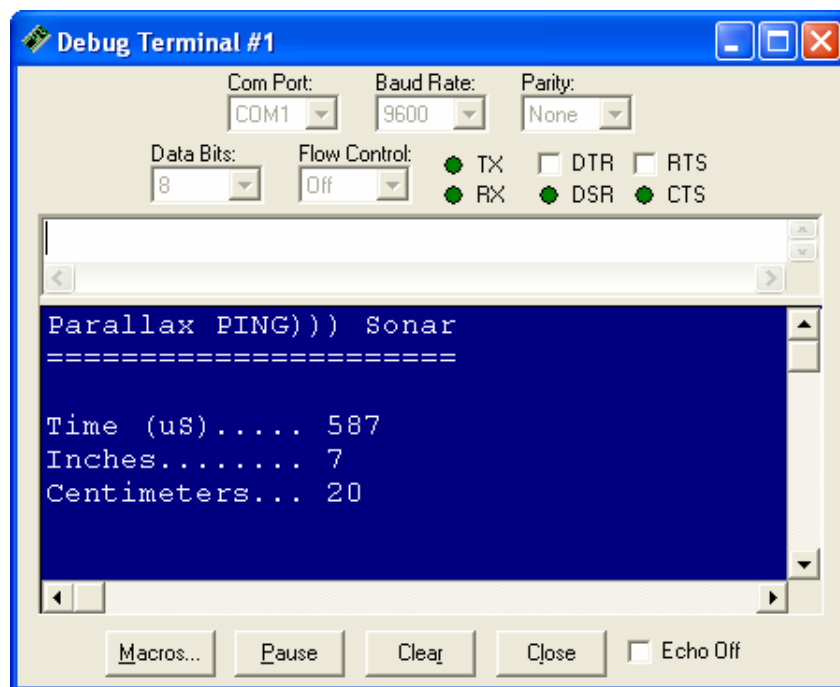
The heart of the program is the **Get\_Sonar** subroutine. This routine starts by making the output bit of the selected IO pin zero – this will cause the successive **PULSOUT** to be low-high-low as required for triggering the PING))) sensor. After the trigger pulse falls the sensor will wait about 200 microseconds before transmitting the ultrasonic burst. This allows the BS2 to load and prepare the next instruction. That instruction, **PULSIN**, is used to measure the high-going pulse that corresponds to the distance to the target object.

The raw return value from **PULSIN** must be scaled due to resolution differences between the various members of the BS2 family. After the raw value is converted to microseconds, it is divided by two in order to remove the "return trip" of the echo pulse. The value now held in *rawDist* is the distance to the target in microseconds.

Conversion from microseconds to inches (or centimeters) is now a simple matter of math. The generally-accepted value for the speed-of-sound is 1130 feet per second. This works out to 13,560 inches per second or one inch in 73.746 microseconds. The question becomes, how do we divide our pulse measurement value by the floating-point number 73.746?

Another way to divide by 73.746 is to multiply by 0.01356. For new BASIC Stamp users this may seem a dilemma but in fact there is a special operator, **\*\***, that allows us to do just that. The **\*\*** operator has the affect of multiplying a value by units of 1/65,536. To find the parameter for **\*\*** then, we simply multiply 0.01356 by 65,536; the result is 888.668 (we'll round up to 889).

Conversion to centimeters uses the same process and the result of the program is shown below:



```
Debug Terminal #1
Com Port: COM1
Baud Rate: 9600
Parity: None
Data Bits: 8
Flow Control: Off
TX
RX
DTR
DSR
RTS
CTS

Parallax PING))) Sonar
=====

Time (uS)..... 587
Inches..... 7
Centimeters... 20

Macros... Pause Clear Close Echo Off
```

```

' =====
'
' File..... Ping_Demo.BS2
' Purpose.... Demo Code for Parallax Ping Sonar Sensor
' Author..... Parallax, Inc.
' E-mail..... support@parallax.com
' Started....
' Updated.... 03 FEB 2005
'
'   {$STAMP BS2}
'   {$PBASIC 2.5}
'
' =====

' -----[ Program Description ]-----
'
' This program demonstrates the use of the Parallax PING))) sensor and then
' converting the raw measurement to English (inches) and Metric (cm) units.
'
' Sonar Math:
'
' At sea level sound travels through air at 1130 feet per second.  This
' equates to 1 inch in 73.746 uS, or 1 cm in 29.034 uS).
'
' Since the PING))) sensor measures the time required for the sound wave to
' travel from the sensor and back.  The result -- after conversion to
' microseconds for the BASIC Stamp module in use -- is divided by two to
' remove the return portion of the echo pulse.  The final raw result is
' the duration from the front of the sensor to the target in microseconds.

' -----[ I/O Definitions ]-----

Ping          PIN      15

' -----[ Constants ]-----

#SELECT $STAMP
#CASE BS2, BS2E
  Trigger      CON      5           ' trigger pulse = 10 uS
  Scale        CON      $200       ' raw x 2.00 = uS
#CASE BS2SX, BS2P
  Trigger      CON      13
  Scale        CON      $0CD       ' raw x 0.80 = uS
#CASE BS2PE
  Trigger      CON      5
  Scale        CON      $1E1       ' raw x 1.88 = uS
#ENDSELECT

RawToIn       CON      889         ' 1 / 73.746 (with **)
RawToCm       CON      2257       ' 1 / 29.034 (with **)

IsHigh        CON      1           ' for PULSOUT
IsLow         CON      0

```

```

' -----[ Variables ]-----
rawDist      VAR      Word      ' raw measurement
inches       VAR      Word
cm           VAR      Word

' -----[ Initialization ]-----

Reset:
  DEBUG CLS,
    "Parallax PING)) Sonar", CR,      ' setup report screen
    "=====", CR,
    CR,
    "Time (uS).....", CR,
    "Inches.....", CR,
    "Centimeters..."

' -----[ Program Code ]-----

Main:
  DO
    GOSUB Get_Sonar      ' get sensor value
    inches = rawDist ** RawToIn  ' convert to inches
    cm = rawDist ** RawToCm      ' convert to centimeters

    DEBUG CRSRXY, 15, 3,      ' update report screen
      DEC rawDist, CLREOL,
      CRSRXY, 15, 4,
      DEC inches, CLREOL,
      CRSRXY, 15, 5,
      DEC cm, CLREOL

    PAUSE 100
  LOOP
END

' -----[ Subroutines ]-----

' This subroutine triggers the Ping sonar sensor and measures
' the echo pulse.  The raw value from the sensor is converted to
' microseconds based on the Stamp module in use.  This value is
' divided by two to remove the return trip -- the result value is
' the distance from the sensor to the target in microseconds.

Get_Sonar:
  Ping = IsLow      ' make trigger 0-1-0
  PULSOUT Ping, Trigger  ' activate sensor
  PULSIN Ping, IsHigh, rawDist  ' measure echo pulse
  rawDist = rawDist */ Scale  ' convert to uS
  rawDist = rawDist / 2      ' remove return trip
  RETURN

```

## Program Example: BASIC Stamp 1 Microcontroller

```
' =====
'
' File..... Ping_Demo.BS1
' Purpose.... Demo Code for Parallax Ping Sonar Sensor
' Author..... Parallax, Inc.
' E-mail..... support@parallax.com
' Started....
' Updated.... 03 FEB 2005
'
'   {$STAMP BS1}
'   {$PBASIC 1.0}
'
' =====

' -----[ Program Description ]-----
'
' This program demonstrates the use of the Parallax PING))) sensor and then
' converting the raw measurement to English (inches) and Metric (cm) units.
'
' Sonar Math:
'
' At sea level sound travels through air at 1130 feet per second.  This
' equates to 1 inch in 73.746 uS, or 1 cm in 29.034 uS).
'
' Since the PING))) sensor measures the time required for the sound wave to
' travel from the sensor and back.  The result -- after conversion to
' microseconds for the BASIC Stamp module in use -- is divided by two to
' remove the return portion of the echo pulse.  The final raw result is
' the duration from the front of the sensor to the target in microseconds.

' -----[ I/O Definitions ]-----

SYMBOL  Ping                = 7

' -----[ Constants ]-----

SYMBOL  Trigger              = 1                ' 10 uS trigger pulse
SYMBOL  Scale                = 10              ' raw x 10.00 = uS

SYMBOL  RawToIn              = 889            ' 1 / 73.746 (with **)
SYMBOL  RawToCm              = 2257          ' 1 / 29.034 (with **)

SYMBOL  IsHigh               = 1                ' for PULSOUT
SYMBOL  IsLow                = 0

' -----[ Variables ]-----

SYMBOL  rawDist              = W1                ' raw measurement
SYMBOL  inches               = W2
SYMBOL  cm                   = W3
```

```

' -----[ Program Code ]-----
Main:
  GOSUB Get_Sonar           ' get sensor value
  inches = rawDist ** RawToIn  ' convert to inches
  cm = rawDist ** RawToCm     ' convert to centimeters

  DEBUG CLS                ' report
  DEBUG "Time (uS)..... ", #rawDist, CR
  DEBUG "Inches..... ", #inches, CR
  DEBUG "Centimeters... ", #cm

  PAUSE 500
  GOTO Main

END

' -----[ Subroutines ]-----

' This subroutine triggers the Ping sonar sensor and measures
' the echo pulse. The raw value from the sensor is converted to
' microseconds based on the Stamp module in use. This value is
' divided by two to remove the return trip -- the result value is
' the distance from the sensor to the target in microseconds.

Get_Sonar:
  LOW Ping                 ' make trigger 0-1-0
  PULSOUT Ping, Trigger   ' activate sensor
  PULSIN Ping, IsHigh, rawDist ' measure echo pulse
  rawDist = rawDist * Scale ' convert to uS
  rawDist = rawDist / 2    ' remove return trip
  RETURN

```

## Program Example: Javelin Stamp Microcontroller

This class file implements several methods for using the PING))) sensor:

```
package stamp.peripheral.sensor;

import stamp.core.*;

/**
 * This class provides an interface to the Parallax PING))) ultrasonic
 * range finder module.
 * <p>
 * <i>Usage:</i><br>
 * <code>
 *   Ping range = new Ping(CPU.pin0);           // trigger and echo on P0
 * </code>
 * <p>
 * Detailed documentation for the PING))) Sensor can be found at: <br>
 * http://www.parallax.com/detail.asp?product\_id=28015
 * <p>
 *
 * @author Jon Williams, Parallax Inc. (jwilliams@parallax.com)
 * @version 1.0 03 FEB 2005
 */
public final class Ping {

    private int ioPin;

    /**
     * Creates Ping range finder object
     *
     * @param ioPin Ping trigger and echo return pin
     */
    public Ping (int ioPin) {
        this.ioPin = ioPin;
    }

    /**
     * Returns raw distance value from the PING))) sensor.
     *
     * @return Raw distance value from PING)))
     */
    public int getRaw() {

        int echoRaw = 0;

        CPU.writePin(ioPin, false);           // setup for high-going pulse
        CPU.pulseOut(1, ioPin);              // send trigger pulse
        echoRaw = CPU.pulseIn(2171, ioPin, true); // measure echo return

        // return echo pulse if in range; zero if out-of-range
        return (echoRaw < 2131) ? echoRaw : 0;
    }
}
```



```

/*
 * The PING)) returns a pulse width of 73.746 uS per inch. Since the
 * Javelin pulseIn() round-trip echo time is in 8.68 uS units, this is the
 * same as a one-way trip in 4.34 uS units. Dividing 73.746 by 4.34 we
 * get a time-per-inch conversion factor of 16.9922 (x 0.058851).
 *
 * Values to derive conversion factors are selected to prevent roll-over
 * past the 15-bit positive values of Javelin Stamp integers.
 */

/**
 * @return PING)) distance value in inches
 */
public int getIn() {
    return (getRaw() * 3 / 51);           // raw * 0.058824
}

/**
 * @return PING)) distance value in tenths of inches
 */
public int getIn10() {
    return (getRaw() * 3 / 5);          // raw / 1.6667
}

/*
 * The PING)) returns a pulse width of 29.033 uS per centimeter. As the
 * Javelin pulseIn() round-trip echo time is in 8.68 uS units, this is the
 * same as a one-way trip in 4.34 uS units. Dividing 29.033 by 4.34 we
 * get a time-per-centimeter conversion factor of 6.6896.
 *
 * Values to derive conversion factors are selected to prevent roll-over
 * past the 15-bit positive values of Javelin Stamp integers.
 */

/**
 * @return PING)) distance value in centimeters
 */
public int getCm() {
    return (getRaw() * 3 / 20);         // raw / 6.6667
}

/**
 * @return PING)) distance value in millimeters
 */
public int getMm() {
    return (getRaw() * 3 / 2);         // raw / 0.6667
}
}

```

This simple demo illustrates the use of the PING))) ultrasonic range finder class with the Javelin Stamp:

```
import stamp.core.*;
import stamp.peripheral.sensor.Ping;

public class testPing {

    public static final char HOME = 0x01;

    public static void main() {

        Ping range = new Ping(CPU.pin0);
        StringBuffer msg = new StringBuffer();

        int distance;

        while (true) {
            // measure distance to target in inches
            distance = range.getIn();

            // create and display measurement message
            msg.clear();
            msg.append(HOME);
            msg.append(distance);
            msg.append(" \"  \" \n");
            System.out.print(msg.toString());

            // wait 0.5 seconds between readings
            CPU.delay(5000);
        }
    }
}
```