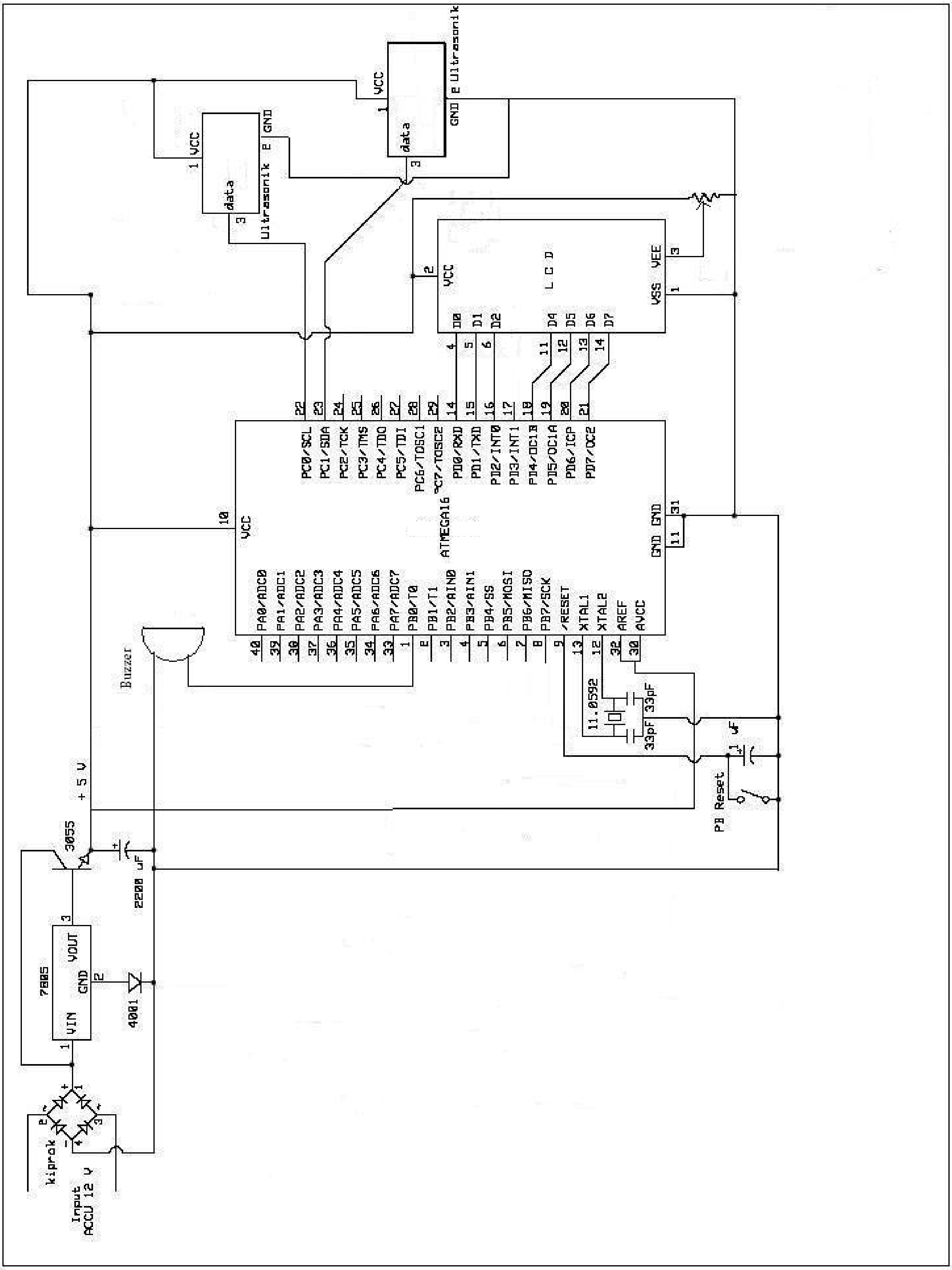
LAMPIRAN

A



Ultrasonik
VCC
data
GND
L C D
VCC
VSS
VEE
ATMEGA16
Buzzer
+ 5 V
3055
2200 uF
7805
VIN
GND
VOUT
4001
kiprok
Input
ACCU 12 V
PB Reset
11.0592
33pF

LAMPIRAN

B

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

Project : TA - Sensor Parkir
Version : 1.0
Date    : 10/05/2010
Author  : Hendri / 0727004
Company : HP
Comments: TUGAS AKHIR


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>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x18 ;PORTB
```

```
#endasm
#include <lcd.h>
#include <delay.h>
#include <stdio.h>

#define PULSE1 PORTA.0
#define PULSE2 PORTA.1
#define ECHO1 PINA.0
#define ECHO2 PINA.1
#define ARAH1 DDRA.0
#define ARAH2 DDRA.1
#define OUT 1
#define INP 0

// Declare your global variables here
unsigned int count1=0;
unsigned int count2=0;
float jarak1;
float jarak2;
unsigned char kata1[16];
unsigned char kata2[16];

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=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTC=0xFF;
DDRC=0xFF;

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTD=0x00;
DDRD=0xFF;

// 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;

// LCD module initialization
lcd_init(16);

while (1)
    {
    // Place your code here
    // port as output
```

```
count1=0;

ARAH1=OUT;
// pulse 2us
PULSE1=1;
delay_ms(5);
PULSE1=0;

// port as input
ARAH1=INP;
//with pull-up
PULSE1=1;

while (ECHO1==0) {};

while (ECHO1==1)
{
count1++;
}

jarak1=(count1*1/65.536);
sprintf(kata1,"Jarak=%3.2f cm",jarak1);
lcd_clear();
lcd_gotoxy(0,0);
lcd_puts(kata1);
if(jarak1<=30)
{
PORTD.7=1;
delay_ms(100);
```

```
PORTD.7=0;
delay_ms(100);
}

// port as output
count2=0;

ARAH2=OUT;
// pulse 2us
PULSE2=1;
delay_ms(5);
PULSE2=0;

// port as input
ARAH2=INP;
//with pull-up
PULSE2=1;

while (ECHO2==0) {};

while (ECHO2==1)
{
count2++;
}

jarak2=(count2*1/65.536);
sprintf(kata2,"Jarak=%3.2f cm",jarak2);
lcd_clear();
lcd_gotoxy(0,1);
```

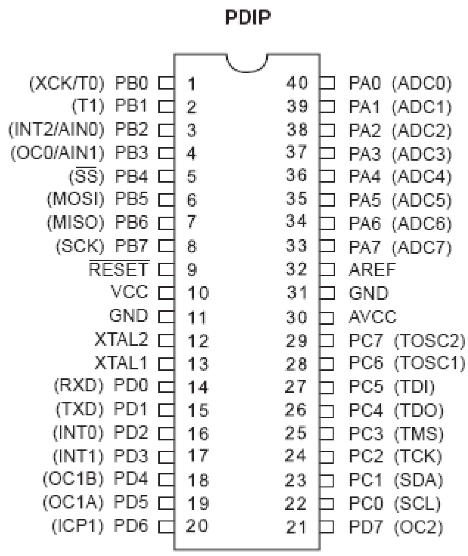
```
    lcd_puts(kata2);
    if(jarak2<=30)
    {
    PORTD.7=1;
    delay_ms(100);
    PORTD.7=0;
    delay_ms(100);
    }
    };
}
```

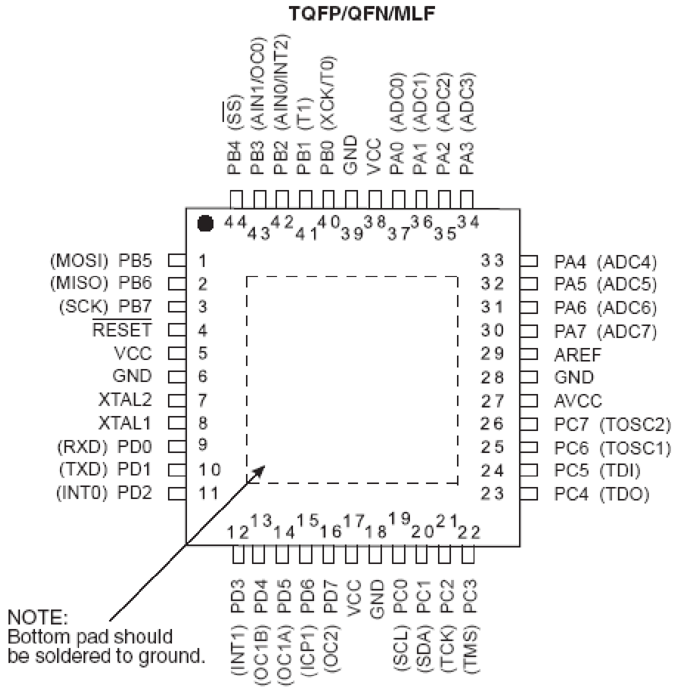
LAMPIRAN

C

Pin Configurations

Figure 1. Pinout ATmega16





Disclaimer

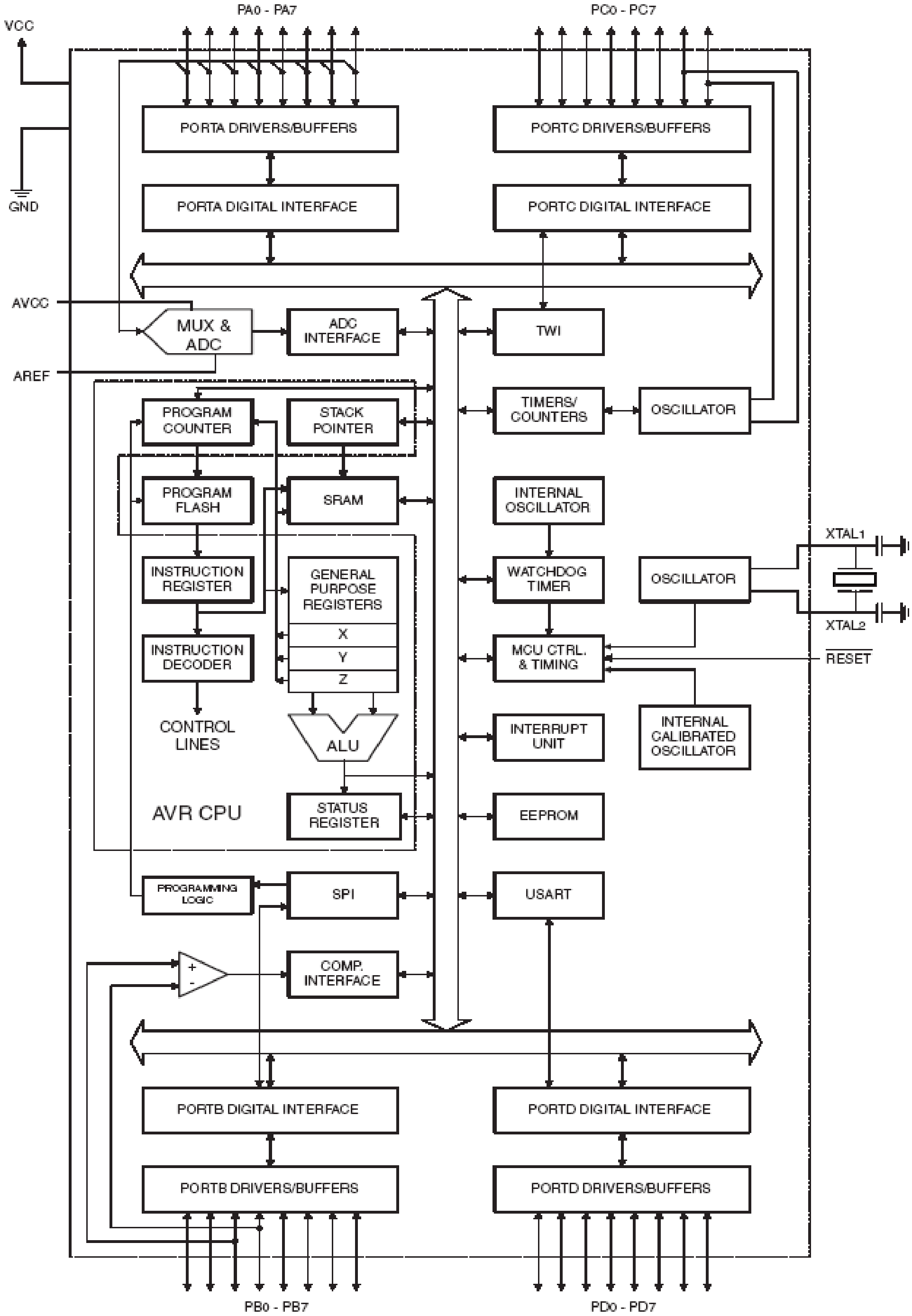
Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

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



Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
8	2.7 - 5.5V	ATmega16L-8AC ATmega16L-8PC ATmega16L-8MC	44A 40P6 44M1	Commercial (0°C to 70°C)
		ATmega16L-8AI ATmega16L-8AU[1] ATmega16L-8PI ATmega16L-8PU[1] ATmega16L-8MI ATmega16L-8MU[1]	44A 44A 40P6 40P6 44M1 44M1	Industrial (-40°C to 85°C)
16	4.5 - 5.5V	ATmega16-16AC ATmega16-16PC ATmega16-16MC	44A 40P6 44M1	Commercial (0°C to 70°C)
		ATmega16-16AI ATmega16-16AU[1] ATmega16-16PI ATmega16-16PU[1] ATmega16-16MI ATmega16-16MU[1]	44A 44A 40P6 40P6 44M1 44M1	Industrial (-40°C to 85°C)

Note: 1. Pb-free packaging alternative, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

Package Type	
44A	44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
40P6	40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)
44M1	44-pad, 7 x 7 x 1.0 mm body, lead pitch 0.50 mm, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)

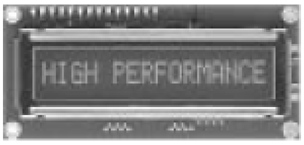
LAMPIRAN

D

Dot Matrix Liquid Crystal Display Modules

CHARACTER TYPE

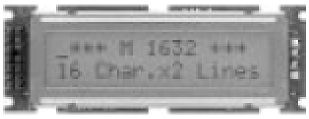
- **FEATURES :**
  - Slim, light weight and low power consumption
  - High contrast and wide viewing angle
  - Built-in controller for easy interfacing
  - LCD modules with built-in EL or LED backlight



M1641

L1642

L1614



M1632

L1652

L2012

- **SPECIFICATIONS :**

(grey) : Standard products (white) : Products of optional specification

		M1641	M1632	L1642	L1652	L1614	L2012
Character Format (character x line)		16 x 1	16 x 2	16 x 2	16 x 2	16 x 4	20 x 2
Model		M1641	M1632	L1642	L1652	L1614	L2012
Reflective		M16410AS	M16320AS	L164200J000S	L165200J200S	L161400J000S	L201200J000S
EL backlight		M16419DWS	M16329DWS	L164221J000S	L165221J200S	L161421J000S	L201221J000S
LED backlight		M16417DYS	M16327DYS	L1642B1J000S	L1652B1J200S	L1614B1J000S	L2012B1J000S
Reflective (wide temp)		M16410CS	M16320CS	L164200L000S	L165200L200S	L161400L000S	L201200L000S
LED backlight (wide temp)		M16417JYS	M16327JYS	L1642B1L000S	L1652B1L200S	L1614B1L000S	L2012B1L000S
Character font		5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor
Module size (HxVxT) mm	Reflective	80,0 x 36,0 x 11,3	85,0 x 30,0 x 10,1	80,0 x 36,0 x 11,3	122,0 x 44,0 x 11,3	87,0 x 60,0 x 11,6	116,0 x 37,0 x 11,3
	EL backlight	80,0 x 36,0 x 11,3	85,0 x 30,0 x 10,1	80,0 x 36,0 x 11,3	122,0 x 44,0 x 11,3	87,0 x 60,0 x 11,6	116,0 x 37,0 x 11,3
	LED backlight	80,0 x 36,0 x 15,8	80,0 x 30,0 x 15,8	80,0 x 36,0 x 15,8	122,0 x 44,0 x 15,8	87,0 x 60,0 x 15,8	116,0 x 37,0 x 15,8
Viewing area (HxV) mm		64,5 x 13,8	62,0 x 16,0	64,5 x 13,8	99,0 x 24,0	61,8 x 25,2	83,0 x 18,6
Character size (HxV) mm *1		3,07 x 5,73	2,78 x 4,27	2,95 x 3,80	4,84 x 8,06	2,95 x 4,15	3,20 x 4,85
Dot size (HxV) mm		0,55 x 0,75	0,50 x 0,55	0,50 x 0,55	0,92 x 1,10	0,55 x 0,55	0,60 x 0,65
Power supply voltage (VDD-VSS) V		+ 5 V	+ 5 V	+ 5 V	+ 5 V	+ 5 V	+ 5 V
Current consumption (mA,typ)	IDD	1,5	2,0	1,6	2,0	2,7	2,0
	ILC *4	0,2	0,2	0,3	0,4	1,1	0,4
Driving method (duty)		1/16	1/16	1/16	1/16	1/16	1/16
Built-in LSI		KS0066 or equivalent	KS0066 MSM5839 or equivalent	KS0066 MSM5839 or equivalent	KS0066 MSM5839 or equivalent	KS0066 KS0063 or equivalent	KS0066 KS0063 or equivalent
Operating temperature (°C)	normal temp.	0 to + 50	0 to + 50	0 to + 50	0 to + 50	0 to + 50	0 to + 50
	wide temp. *2	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70
Storage temperature (°C)	normal temp.	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60
	wide temp.	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80
Weight (g, typ.)	Reflective	25	25	25	50	50	40
	EL backlight	30	30	30	55	55	45
	LED backlight	35	40	35	65	65	60
Inverters for EL	Model	5S	5S	5S	5C	5A	5A
	Power supply (V)	+ 5.0	+ 5,0	+ 5.0	+ 5.0	+ 5.0	+ 5.0
	current consumption (mA) *3	10	10	10	35	45	45
LED backlight	Forward current consumption (mA)	100	112	100	240	200	154
	Forward input voltage (V,typ.)	+ 4,1	+ 4,1	+ 4,1	+ 4,1	+ 4,1	+ 4,1

*1 : Excluding cursor

H : Horizontal V : Vertical T : Thickness (max)

*2 : With external temperature compensation

*3 : Including EL backlight

*4 : Based on normal temperature range

Since our policy is one of continues improvements we reserve the right to change the specifications for the products in the catalogue without notice.

L2022

L2432

L4042

L2014

M4024

• SPECIFICATIONS :

(grey) : Standard products (white) : Products of optional specification

Character Format (character x line)		20 x 2	20 x 4	24 x 2	40 x 2	40 x 4
Model		L2022	L2014	L2432	L4042	M4024
Reflective		-	L201400J000S	L243200J000S	L404200J000S	M40240AS
EL backlight		-	L201421J000S	L243221J000S	L404221J000S	M40249DWS
LED backlight		-	L2014B1J000S	L2432B1J000S	L4042B1J000S	M40247DYS
Reflective (wide temp)		L202200P000S	L201400L000S	L243200L000S	L404200L000S	M40240CS
LED backlight (wide temp)		L2022B1P000S	L2014B1L000S	L2432B1L000S	L4042B1L000S	M40247JYS
Character font		5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor	5x7 dots + cursor
Module size (HxVxT) mm	Reflective	180,0 x 40,0 x 10,5	98,0 x 60,0 x 11,6	118,0 x 36,0 x 11,3	182,0 x 33,5 x 11,3	190,0 x 54,0 x 10,1
	EL backlight	180,0 x 40,0 x 10,5	98,0 x 60,0 x 11,6	118,0 x 36,0 x 11,3	182,0 x 33,5 x 11,3	190,0 x 54,0 x 10,1
	LED backlight	180,0 x 40,0 x 14,8	98,0 x 60,0 x 15,8	118,0 x 36,0 x 15,8	182,0 x 33,5 x 16,3	190,0 x 54,0 x 16,3
Viewing area (HxV) mm		149,0 x 23,0	76,0 x 25,2	94,5 x 17,8	154,4 x 15,8	147,0 x 29,5
Character size (HxV) mm *1		6,00 x 9,66	2,95 x 4,15	3,20 x 4,85	3,20 x 4,85	2,78 x 4,27
Dot size (HxV) mm		1,12 x 1,12	0,55 x 0,55	0,60 x 0,65	0,60 x 0,65	0,50 x 0,55
Power supply voltage (VDD-VSS) V		+ 5 V	+ 5 V	+ 5 V	+ 5 V	+ 5 V
Current consumption (mA,typ)	IDD	4,2	2,9	2,5	3,0	8,0
	ILC *4	2,6	1,2	0,5	1,0	3,0
Driving method (duty)		1/16	1/16	1/16	1/16	1/16
Built-in LSI		KS0066 KS0063 or equivalent	KS0066 MSM5839 or equivalent	KS0066 KS0063 or equivalent	KS0066 KS0063 or equivalent	KS0066 MSM5839 or equivalent
Operating temperature (°C)	normal temp.	-	0 to + 50	0 to + 50	0 to + 50	0 to + 50
	wide temp. *2	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70	- 20 to + 70
Storage temperature (°C)	normal temp.	-	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60
	wide temp.	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80	- 30 to + 80
Weight (g, typ.)	Reflective	80	55	40	70	90
	EL backlight	-	60	45	75	105
	LED backlight	110	70	60	95	140
Inverters for EL	Model	-	5A	5A	5C	5D
	Power supply (V)	+ 5.0	+ 5.0	+ 5.0	+ 5.0	+ 5.0
	current consumption (mA) *3	-	45	45	25	80
LED backlight	Forward current consumption (mA)	320	240	150	260	480
	Forward input voltage (V,typ.)	+ 4,1	+ 4,1	+ 4,1	+ 4,1	+ 4,1

*1 : Excluding cursor H : Horizontal V : Vertical T : Thickness (max)

*2 : With external temperature compensation

*3 : Including EL backlight

*4 : Based on normal temperature range

Dot Matrix Liquid Crystal Display Modules

GRAPHIC TYPE

• FEATURES :

- •Wide viewing angle and high contrast
- •Full dot configuration fits any application
- •Slim, light weight and low power consumption
- •Available in STN and FSTN

• SPECIFICATIONS :

Dot format (HxV,dot)			97 x 32	128 x 32	128 x 64	128 x 64
Model			Y97031	G1213	G1216	G1226
STN type (Gray mode)	Reflective	built-in RAM	-	-	-	-
	Reflective wide temp.	built-in RAM	-	G121300N000S	G121600N000S	-
	LED backlight	built-in RAM	-	-	-	G1226B1J000S
	LED backlight wide temp	built-in RAM	-	G1213B1N000S	G1216B1N000S	-
FSTN type (B&W mode)	Transmissive	-	-	-	-	-
	with CFL backlight	built-in controller	-	-	-	-
	Transflective	built-in RAM	Y97031LF60W	-	-	-
Module size (H x V x T) mm	Reflective (no backlight)		47,5 x 65,4 x 2,1	75,0 x 41,5 x 6,8	75,0 x 52,7 x 6,8	-
	LED backlight		-	75,0 x 41,5 x 8,9	75,0 x 52,7 x 8,9	93,0 x 70,0 x 11,4
	CFL backlight		-	-	-	-
Viewing area (HxV) mm			43,5 x 23,9	60,0 x 21,3	60,0 x 32,5	70,7 x 38,8
Dot size (H x V) mm			0,35 x 0,46	0,40 x 0,48	0,40 x 0,40	0,44 x 0,44
Dot pitch (H x V) mm			0,39 x 0,52	0,43 x 0,51	0,43 x 0,43	0,48 x 0,48
Power supply voltage (V)		(VDD - VSS)	+ 5,0	+ 5,0	+ 5,0	+ 5,0
		(VLC - VSS)	-	- 8,0	- 8,1	-8,2
Current consumption (mA, typ.)		IDD	0,10	2,0	2,0	3,0
		IDD (built-in controller)	-	-	-	-
		ILC	-	1,8	1,8	2,0
Driving method (duty)			1/33	1/64	1/64	1/64
Built-in LSI		Driver	SED1530 or equivalent	HD61202 HD61203 or equivalent	HD61202 HD61203 or equivalent	KS0107 KS0108 or equivalent
		Controller	-	-	-	-
Operating temperature range (°C)			- 20 to + 70	- 20 to + 70	- 20 to + 70	0 to + 50
Storage temperature range (°C)			- 30 to + 80	- 30 to + 80	- 30 to + 80	- 20 to + 60
Weight (g, typ.)	Reflective (Transflective no backlight)		10	23	35	-
	LED backlight		-	35	45	72
	CFL backlight		-	-	-	-
LED backlight	Forward current consumption (mA)		-	40	90	125
	Forward input voltage (V, typ.)		-	3,8	4,1	4,1
Inverter for CFL	Mode		-	-	-	-
	Power supply voltage (V)		-	-	-	-
	Current consumption (mA, typ.)		-	-	-	-

*1 : built-in DC/DC converter (single power source)

*2 : Use with external temperature compensation circuit

Since our policy is one of continues improvements we reserve the right to change the specifications of the products in the catalogue without notice.

Dot format (HxV,dot)			240 x 64	240 x 128	320 x 200	320 x 240	640 x 200
Model			G2446	G242C	G321D	G324E	G649D
STN type	Reflective	built-in RAM	-	-	-	-	-
(Gray mode)	Reflective wide temp.	built-in RAM	-	-	-	-	-
	LED backlight	built-in RAM	-	-	-	-	-
	LED backlight wide temp	built-in RAM	-	-	-	-	-
FSTN type	Transmissive	-	G2446X5R1A0S	G242CX5R1ACS	G321DX5R1A0S	G324EX5R1A0S	G649DX5R010S
(B&W mode)	with CFL backlight	built-in controller	G2446X5R1ACS	G242CX5R1A0S	G321DX5R1ACS	G324EX5R1ACS	-
	Transflective	built-in RAM	-	-	-	-	-
Module size	Reflective (no backlight)			-	-	-	-
(H x V x T)	LED backlight			-	-	-	-
mm	CFL backlight		191,0 x 79,0 x 15,1	180,0 x 110,0 x 15,1	166,0 x 134,0 x 15,1	166,0 x 134,0 x 15,1	260,0 x 122,0 x 15,7
Viewing area (HxV) mm			134,0 x 41,0	134,0 x 76,0	128,0 x 110,0	128,0 x 110,0	216,0 x 83,0
Dot size (H x V) mm			0,49 x 0,49	0,47 x 0,47	0,34 x 0,48	0,32 x 0,39	0,30 x 0,36
Dot pitch (H x V) mm			0,53 x 0,53	0,51 x 0,51	0,38 x 0,52	0,36 x 0,43	0,33 x 0,39
Power supply voltage (V)		(VDD - VSS)	+ 5,0	+ 5,0	+ 5,0	+ 5,0	+ 5,0
		(VLC - VSS)	*1	*1	-24,0	-24,0	-24,0
Current consumption		IDD	12	30	8	7,5	11
		IDD (built-in controller)	15	40	23	23	-
(mA, typ.)		ILC	-	-	6	6,5	9
Driving method (duty)			1/64	1/128	1/200	1/240	1/200
Built-in LSI		Driver	MSM5298 MSM5299 or equivalent	KS0103 KS0104 or equivalent	MSM5298 MSM5299 or equivalent	HD66204 HD66205 or equivalent	MSM5298 MSM5299 or equivalent
		Controller	SED1330FB	SED1330FB	SED1330FB	SED1330FB	-
Operating temperature range (°C)			0 to + 50	0 to + 50	0 to + 50	0 to + 50	0 to + 50
Storage temperature range (°C)			- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60	- 20 to + 60
Weight	Reflective (Transflective no backlight)		-	-	-	-	-
(g, typ.)	LED backlight		-	-	-	-	-
	CFL backlight		200	280	350	350	420
LED backlight	Forward current consumption (mA)		-	-	-	-	-
	Forward input voltage (V, typ.)		-	-	-	-	-
Inverter for CFL	Mode		4800210	4800210	4800210	4800210	4800120
	Power supply voltage (V)		+ 5,0	+ 5,0	+ 5,0	+ 5,0	+ 12,0
	Current consumption (mA, typ.)		250	350	365	365	390

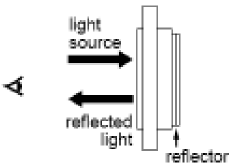
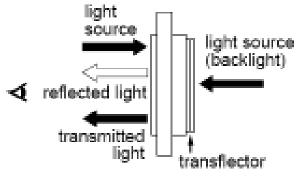
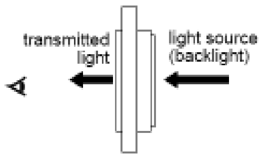
*1 : built-in DC/DC converter (single power source)

*2 : Use with external temperature compensation

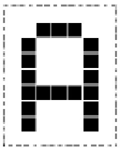
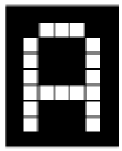
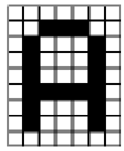
Since our policy is one of continous improvemets, we reserve the right to change the specifications of the products in the catalogue without notice.

Liquid Crystal Display Modules

■ REFLECTIVE/TRANSFLECTIVE/TRANSMISSIVE LCD

❶ Reflective LCD	❷ Transflective LCD	❸ Transmissive LCD
Reflector bonded to the rear polarizer reflects the incoming ambient light. Low power consumption because no backlight is required. 	Transflector bonded to the rear polarizer reflects light from the front as well as enabling lights to pass through the back. Used with backlight off in bright light and with it on in low light to reduce power consumption. 	Without reflector or transflector bonded to the rear polarizer. Backlight required. Most common is transmissive negative image. 

■ POSITIVE/NEGATIVE MODE

Positive type		Negative type		Negative type (inverse image) (when data is inverted)	

■ TN TYPE/STN TYPE/FSTN TYPE

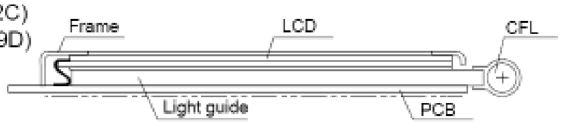
TN	(Background/dot color) Gray/Black	TN(Twisted Nematic) type is most conventional and economical. It is used for static drive LCD and low-duty drive LCD (watch,calculator, etc.)
STN	Yellowgreen/Dark blue Gray/Dark blue White/Blue	STN (Super Twisted Nematic) type has a higher twist angle, and thus provides clear visibility and wider viewing angle. This is suitable especially for high-duty drive LCD.
FSTN	White/Black	FSTN (Film Super Twisted Nematic) type utilizes RCF (Retardation Control Film) to remove the coloring of STN LCD. Thus FSTN type provides easy-to-read black-and-white display.

■ STRUCTURE AND FEATURE OF LCD MODULE WITH BACKLIGHT

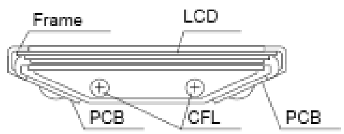
CFL (Cold Cathode Fluorescent Lamp) backlight

Features: high brightness, long service life, inverter required

- Edge backlight type
  (G2446,G242C)
  (G321D,G649D)

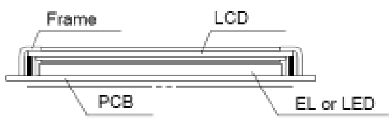


- Backlight type



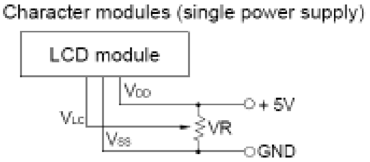
EL (Electroluminescent Lamp) backlight
LED (Light Emitting Diode) backlight

Features: EL: thin, inverter required
LED: long service life, low voltage driving, no inverter required

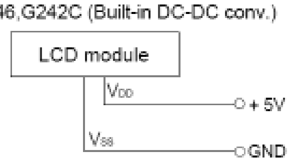


■ POWER SUPPLY

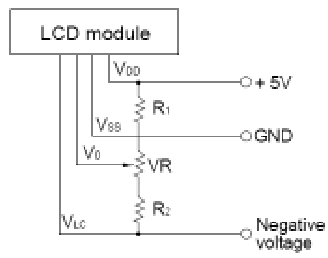
- Character modules (single power supply)



- G2446,G242C (Built-in DC-DC conv.)

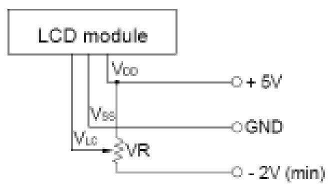


- G321D, G324E and G649D

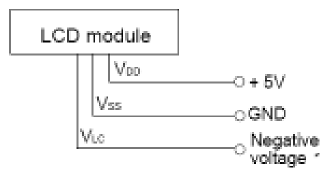


Note 1:Contrast can be adjusted by VR.
Note 2:For module with backlight, power supply for backlight is necessary.

- Character Modules(Dual power supply)



- Y1206 and G1226



*Negative voltage should be variable for contrast adjustment.

LAMPIRAN
E

PARALLAX
599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003
General: info@parallax.com
Technical: support@parallax.com
Web Site: www.parallax.com
Educational: www.stampsinclass.com

PING)))™ Ultrasonic Distance Sensor (#28015)

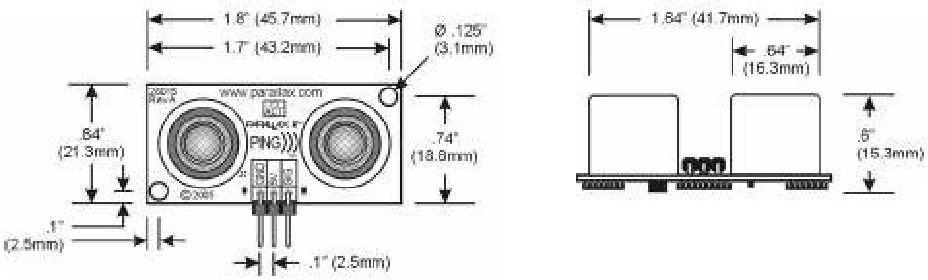
The Parallax PING))) ultrasonic distance sensor provides precise, non-contact distance measurements from about 2 cm (0.8 inches) to 3 meters (3.3 yards). It is very easy to connect to BASIC Stamp® or Javelin Stamp microcontrollers, requiring only one I/O pin.

The PING))) sensor works by transmitting an ultrasonic (well above human hearing range) burst and providing an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width the distance to target can easily be calculated.

Features

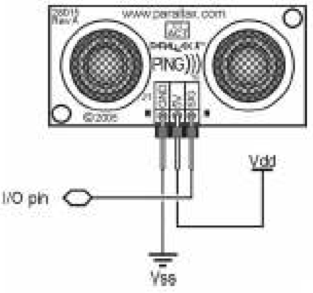
- Supply Voltage – 5 VDC
- Supply Current – 30 mA typ; 35 mA max
- Range – 2 cm to 3 m (0.8 in to 3.3 yrds)
- Input Trigger – positive TTL pulse, 2 uS min, 5 µs typ.
- Echo Pulse – positive TTL pulse, 115 uS to 18.5 ms
- Echo Hold-off – 750 µs from fall of Trigger pulse
- Burst Frequency – 40 kHz for 200 µs
- Burst Indicator LED shows sensor activity
- Delay before next measurement – 200 µs
- Size – 22 mm H x 46 mm W x 16 mm D (0.84 in x 1.8 in x 0.6 in)

Dimensions



Pin Definitions

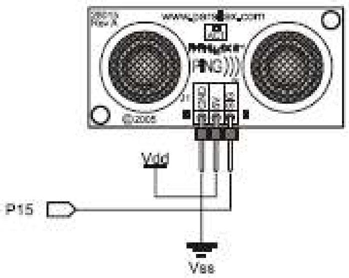
GND	Ground (Vss)
5 V	5 VDC (Vdd)
SIG	Signal (I/O pin)

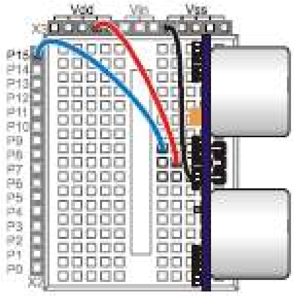


The PING))) sensor has a male 3-pin header used to supply power (5 VDC), ground, and signal. The header allows the sensor to be plugged into a solderless breadboard, or to be located remotely through the use of a standard servo extender cable (Parallax part #805-00002). Standard connections are show in the diagram to the right.

Quick-Start Circuit

This circuit allows you to quickly connect your PING))) sensor to a BASIC Stamp® 2 via the Board of Education® breadboard area. The PING))) module's GND pin connects to Vss, the 5 V pin connects to Vdd, and the SIG pin connects to I/O pin P15. This circuit will work with the example program Ping_Demo.BS2 listed on page 7.

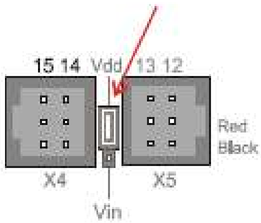




Servo Cable and Port Cautions

If you want to connect your PING))) sensor to a Board of Education using a servo extension cable, follow these steps:

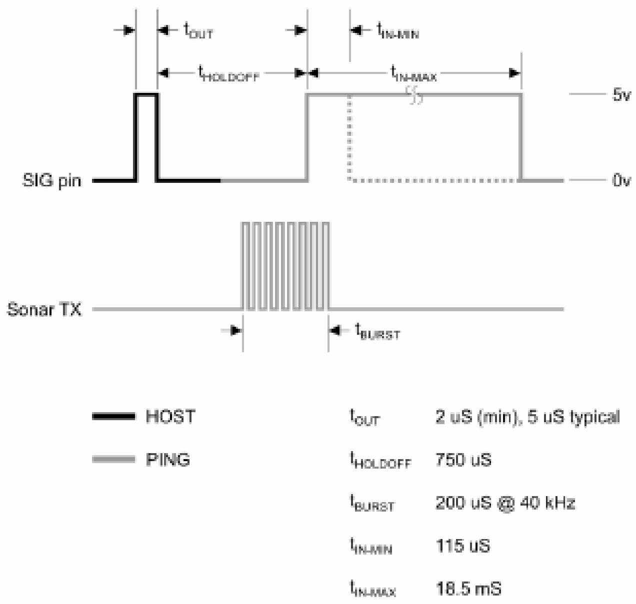
1. When plugging the cable onto the PING))) sensor, connect Black to GND, Red to 5 V, and White to SIG.
2. Check to see if your Board of Education servo ports have a jumper, as shown at right.
3. If your Board of Education servo ports have a jumper, set it to Vdd as shown.
4. If your Board of Education servo ports do not have a jumper, do not use them with the PING))) sensor. These ports only provide Vin, not Vdd, and this may damage your PING))) sensor. Go to the next step.
5. Connect the servo cable directly to the breadboard with a 3-pin header. Then, use jumper wires to connect Black to Vss, Red to Vdd, and White to I/O pin P15.



Board of Education Servo Port Jumper, Set to Vdd

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.

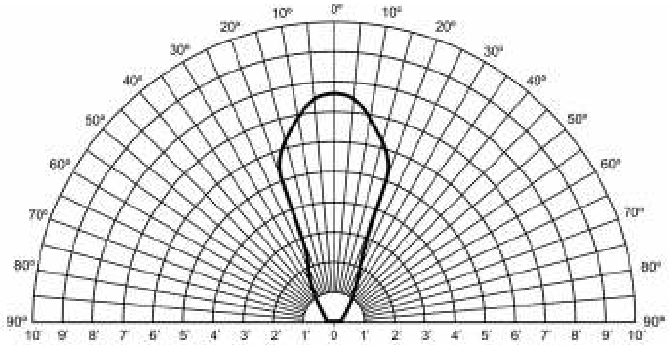


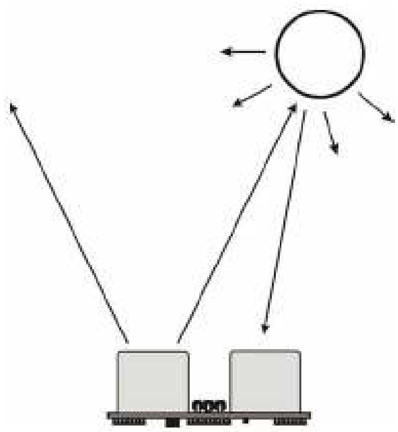
Test Data

The test data on the following pages is based on the PING))) sensor, tested in the Parallax lab, while connected to a BASIC Stamp microcontroller module. The test surface was a linoleum floor, so the sensor was elevated to minimize floor reflections in the data. All tests were conducted at room temperature, indoors, in a protected environment. The target was always centered at the same elevation as the PING))) sensor.

Test 1

Sensor Elevation: 40 in. (101.6 cm)
Target: 3.5 in. (8.9 cm) diameter cylinder, 4 ft. (121.9 cm) tall – vertical orientation

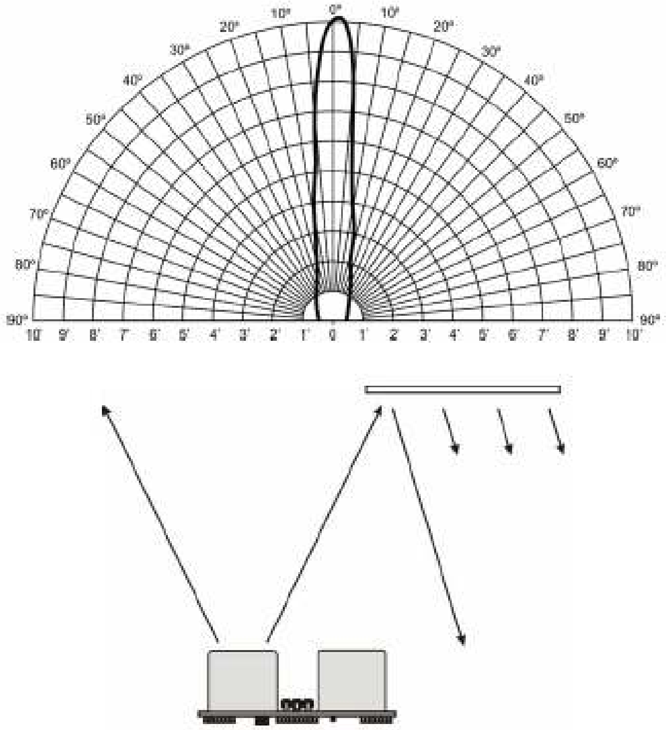




Test 2

Sensor Elevation: 40 in. (101.6 cm)
Target: 12 in. x 12 in. (30.5 cm x 30.5 cm) cardboard, mounted on 1 in. (2.5 cm) pole

- target positioned parallel to backplane of sensor



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.

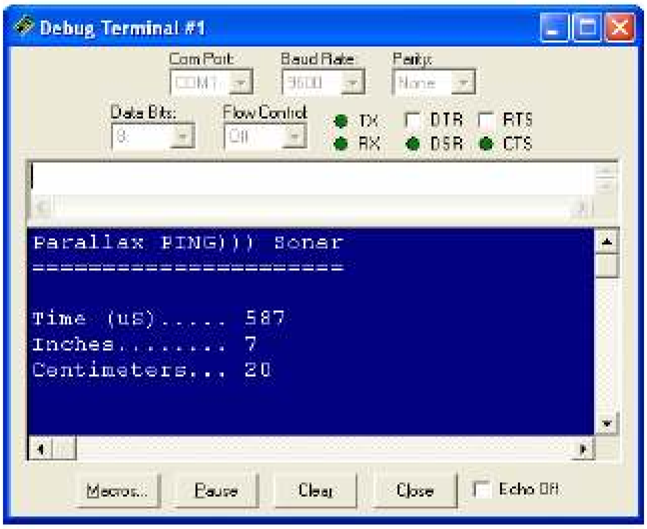
The heart of the program is the **Get_Sonar** subroutine. This routine starts by making the output bit of the selected IO pin zero – this will cause the successive **PULSOUT** to be low-high-low as required for triggering the PING))) sensor. After the trigger pulse falls the sensor will wait about 200 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:



```
' =========================================================================
'
'   File....... Ping_Demo.BS2
'   Purpose.... Demo Code for Parallax PING))) Sonar Sensor
'   Author..... Parallax, Inc.
'   E-mail..... support@parallax.com
'   Started....
'   Updated.... 08 JUN 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, BS2PX
    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
```

```
' =========================================================================
'
'   File....... Ping_Demo.BS1
'   Purpose.... Demo Code for Parallax PING))) Sonar Sensor
'   Author..... Parallax, Inc.
'   E-mail..... support@parallax.com
'   Started....
'   Updated.... 06 JUN 2006
'
'   {$STAMP BS1}
'   {$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>
 *
 * @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);
    }
  }
}
```