I2CXL-MaxSonar®- WR/WRC™ Series

High Performance Sonar Rangefinder
MB7040, MB7047

CE   RoHS COMPLIANT

The I2CXL-MaxSonar-WR ultrasonic sensor features an I2C interface. The sensors have high acoustic power output along with real-time auto calibration for changing conditions (voltage and acoustic or electrical noise) that ensure users receive the most reliable (in air) ranging data for every reading taken. The I2CXL-MaxSonar-WR low power 3V – 5.5V operation provides very short to long-range detection and ranging, in a tiny and compact form factor. The I2CXL-MaxSonar-WR will typically detect objects from 0-cm to 765-cm (3-cm to 625-cm for the WRC) and provide sonar range information from 20-cm out to 765-cm with 1-cm resolution. Objects from 0-cm (3-cm for the WRC) to 20-cm typically range as 20-cm.

Features

- I2C bus communication allows rapid control of multiple sensors with only two wires
- High acoustic power output
- Real-time auto calibration and noise rejection for every ranging cycle
- Calibrated beam patterns
- Continuously variable gain
- Object presence information up to the front face of the sensor (3-cm for the WRC). Range information starting at min. distance.
- 3V to 5.5V supply with very low average current draw
- Readings can occur up to every 25mS (40Hz rate)[3] for up-close objects. 15Hz rate for full range.
- Triggered operation provides a new range reading as desired
- Ultrasonic signal frequency of 42KHz
- Status pin available to determine sensor state
- Power-up address reset pin available
- Physical dimensions match other XL-MaxSonar-WR/WRC products
- -40°C to +65°C operation

Low Power Requirement

- Wide, low supply voltage requirements eases battery powered design
- Low current draw reduces current drain for battery operation
- Fast first reading after power-up eases battery requirements

Benefits

- Acoustic and electric noise resistance
- Reliable and stable range data
- Low cost
- Quality controlled beam characteristics
- Very low power rangefinder, excellent for multiple sensor or battery based systems
- Ranging is triggered externally
- Fast measurement cycle
- No power up calibration required
- Perfect for when objects may be directly in front of the sensor during power up
- Easy mounting

Applications and Uses

- Multi-sensor arrays
- Proximity zone detection
- People detection
- Robot ranging sensor
- Autonomous navigation
- Educational and hobby robotics
- Environments with acoustic and electrical noise
- Distance measuring
- Long range object detection
- Security systems
- Motion detection
- Collision avoidance
- Bin level measurement

Notes:

[1] Users are encouraged to evaluate the sensors performance in their application
[2] By design
[3] Recommended time between readings of 100ms (10Hz Rate)

Close Range Operation

Applications requiring 100% reading-to-reading reliability should not use MaxSonar sensors at a distance closer than 20cm. Although most users find MaxSonar sensors to work reliably from 0 to 20cm for detecting objects in many applications, MaxBotix® Inc. does not guarantee operational reliability for objects closer than the minimum reported distance. Because of ultrasonic physics, these sensors are unable to achieve 100% reliability at close distances.

Warning: Personal Safety Applications

We do not recommend or endorse this product be used as a component in any personal safety applications. This product is not designed, intended or authorized for such use. These sensors and controls do not include the self-checking redundant circuitry needed for such use. Such unauthorized use may create a failure of the MaxBotix® Inc. product which may result in personal injury or death. MaxBotix® Inc. will not be held liable for unauthorized use of this component.

About Ultrasonic Sensors

Our ultrasonic sensors are in air, non-contact object detection and ranging sensors that detect objects within an area. These sensors are not affected by the color or other visual characteristics of the detected object. Ultrasonic sensors use high frequency sound to detect and localize objects in a variety of environments. Ultrasonic sensors measure the time of flight for sound that has been transmitted to and reflected back from nearby objects. Based upon the time of flight, the sensor determines the range to a target.

Pin Out

GND: Return for the DC power supply. GND (& V+) must be ripple and noise free for best operation.
V+: Operates on 3VDC to 5.5VDC. The average current draw for 3.3V operation is 2.7mA (50mA peak) and for 5V operation is 4.4mA (100mA peak) respectively. Peak current is used during sonar pulse transmit.
Pin 5-SCL (I2C Clock): This is the clock line for I2C communications. These sensors support I2C clock frequencies up to 400kHz provided clock stretching is supported by the master device. Without clock stretching the devices can run at speeds up to 50kHz.
Pin 4-SDA (I2C Data): This is the data line for I2C communications. These sensors operate as I2C slave devices.
Pin 3-Not Used: This pin is not used.
Pin 2-Address Announce / Status: While the sensor is performing a range reading, this pin is set high and I2C communications are ignored. During non-ranging operation, this pin is held low and the sensor is listening for incoming I2C communication. Optionally, users may poll this pin to determine if the sensor has finished its ranging cycle and is ready to report the latest range information.
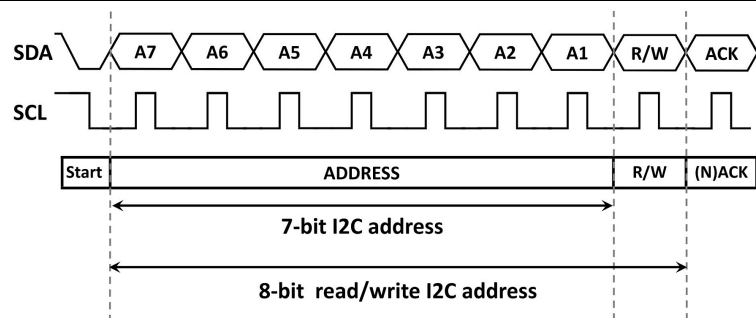During power-up this pin will provide a pulse width representation of the sensors current address with a length of ~100 microseconds per digit. (The default address of 224 will announce with a pulse of 22,400 microseconds in length)
Pin 1-Temporary Default: This pin is internally pulled high. On power up, the state of this pin is checked; if left high or disconnected, the sensor will use the address stored memory for I2C communications. If pulled low, the sensor will use its default address for the current power cycle.

Default Address

The representation of the sensor address will be different depending on the addressing scheme your master device uses. The chart below shows the default address for the I2CXL-MaxSonar-WR/WRC sensors under different addressing implementations. Elsewhere in this datasheet a 8-bit read/write addressing scheme is assumed.

| Addressing Scheme	Default Address (decimal)	Default Address (binary)	Notes
7-bit addressing	112	1110 000X	7-bit addressing handles the address shifting and R/W bit for the user
8-bit addressing	224	1110 000X	8-bit addressing inserts the R/W bit and only allows for even number addresses
8-bit read/write addressing	Write: 224 Read: 225	1110 0000 1110 0001	8-bit R/W addressing schemes require the user to set the R/W bit directly.

MaxBotix® Inc.
Copyright 2005 - 2012 MaxBotix Incorporated
Patent 7,679,996

MaxBotix Inc., products are engineered and assembled in the USA

Page 2
Web: www.maxbotix.com
PD11502e

I2C-MaxSonar-WR Commands

Command	Sequence of Events	Value Used (decimal)	Value Used (binary)	Notes
Take a range reading	**1.** Initiate a write at the sensor address **2.** Write the range command byte	224 (default) **81**	1110 0000 0101 0001	Commands the sensor to take a single range reading and save to distance found for the next range request. It is best to allow 100ms between readings to allow for proper acoustic dissipation.
Report the last range value	**1.** Initiate a read at the sensor address **2a.** Read the two bytes from the sensor starting with the range high byte. **2b.** Read the range low byte.	225 (default) *(Sent by sensor)* *(Sent by sensor)*	1110 0001 Range High Byte Values are MSB to LSB	The sensor will report the distance value in cm obtained from its last range reading. Users requiring real-time information should command a range reading ~80ms before reading the sensor. After power-up if no range command is sent the sensor respond with two part info bytes.
Change the sensor address	**1.** Initiate a write at the sensor address **2a.** Write three bytes to the sensor starting with the addr_unlock_1 command **2b.** Write the addr_unlock_2 command **2c.** Write the new sensor address	224 (default) **170** **165** *(User Value)*	1110 0000 1010 1010 1010 0101 #### ###0	The sensor will only accept even address values. If an odd numbered address is sent the sensor will be set to the next lowest even number. If the sensor is told to change to one of the invalid addresses below the sensor will ignore this command and stay at its current address. **Invalid Address Values:** 0, 80, 164, 170

Range Cycle Interrupt

If the sensor receives a request to report the last range value while it is taking a range reading the range reading will be interrupted and a NACK will be sent. If you desire to allow the full range cycle to complete before reading monitor the status pin for the completion of a range cycle or wait for the full 100ms for the range . If the sensor is interrupted and has already found a target the sensor will report the range to the target. If the sensor has not yet found a target when it is interrupted the sensor will send the previous range value. If no range values have been found the distance sent will alternate between 0cm and 255cm.

MaxBotix® Inc.
Copyright 2005 - 2012 MaxBotix Incorporated
Patent 7,679,996

MaxBotix Inc., products are engineered and assembled in the USA

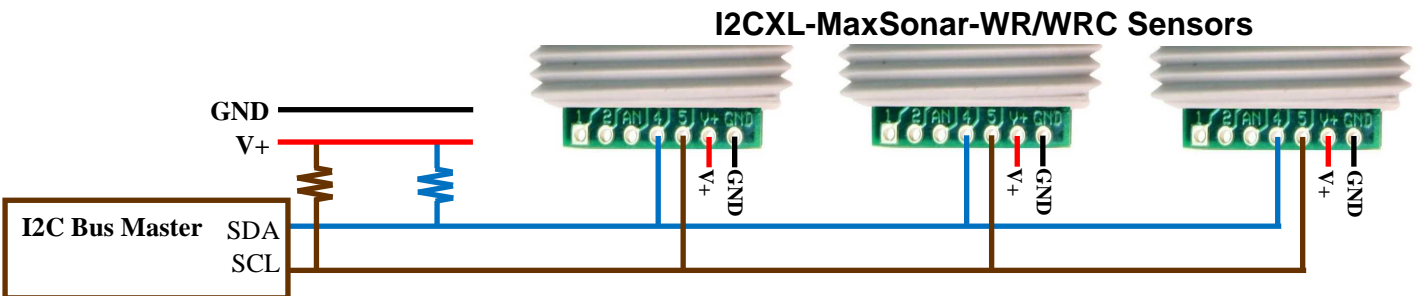Page 3
Web: www.maxbotix.com
PD11502e

Single Sensor Wiring Diagram

The I2C bus is a two wire interface that consists of a clock line and data line where each requires a pull-up resistor attached to V+. Only one pull-up resistor is required each for the SCL and SDA lines per bus – not per sensor.

I2CXL-MaxSonar-WR/WRC Sensor
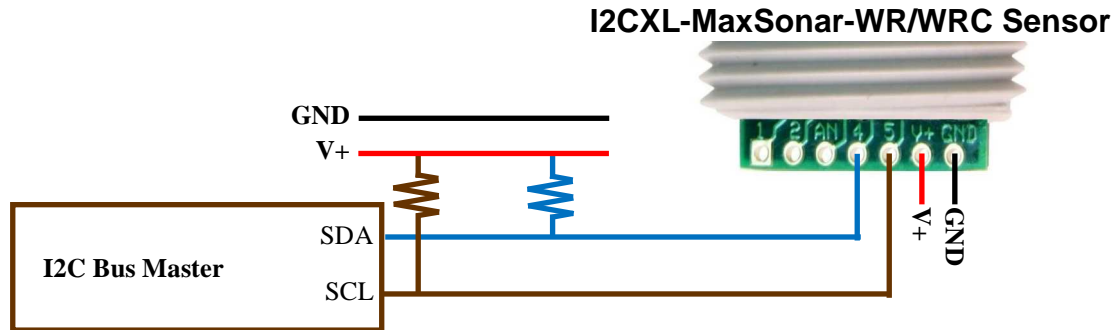
The I2C specification recommends a resistance value of 4.7 kΩ for 20-100kHz interfaces with good low inductance routing. However, these specifications are for communication between chips on a singe PCB. If you have longer cable lengths it is best to use lower value resistor, such as 1kΩ, and also to use properly shielded cables. Often I2C bus problems can be fixed by doing one of the following: by using properly shielded cable or by decreasing the value of the pull-up resistors. The I2CXL-MaxSonar-WR/WRC series is capable of sinking more current than the I2C specification requires (15mA versus 3mA) so a much lower resistance value can be used. The voltage applied to the I2C lines should be the same voltage that is applied to V+ of the sensor.

Multiple Sensor Wiring Diagram

I2CXL-MaxSonar-WR/WRC Sensors

MaxBotix® Inc.
Copyright 2005 - 2012 MaxBotix Incorporated
Patent 7,679,996

MaxBotix Inc., products are engineered and assembled in the USA

Page 4
Web: www.maxbotix.com
PD11502e

Range "0" Location

The I2CXL-MaxSonar-WR sensors reports the range to distant targets starting from the front of the sensor as shown in the diagrams below.

The I2CXL-MaxSonar-WR/WRC will report the range to the closest detectable object. Target detection has been characterized in the sensor beam patterns.

Range Zero

The range is measured from the front of the transducer to the target.

Range Zero

The range is measured from the front of the transducer to the target.

Range Zero

The range is measured from the front of the transducer to the target.

Range Zero

The range is measured from the front of the transducer to the target.

MaxBotix® Inc.

Copyright 2005 - 2012 MaxBotix Incorporated
Patent 7,679,996

MaxBotix Inc., products are engineered and assembled in the USA

Page 5
Web: www.maxbotix.com
PD11502e

Sensor Minimum Distance

The sensor minimum reported distance is 20-cm . However, the I2CXL-MaxSonar-WR will range and report targets to the front sensor face. Large targets closer than 20-cm will typically range as 20-cm. For the I2CXL-MaxSonar-WR alternative housings, objects between 3-cm and 20-cm will typically range as 20-cm.

Real-time Noise Rejection

While the I2CXL-MaxSonar-WR/WRC is designed to operate in the presence of noise, best operation is obtained when noise strength is low and desired signal strength is high. The user is encouraged to mount the sensor in such a way that minimizes outside acoustic noise pickup. In addition, keep the DC power to the sensor free of noise. This will let the sensor deal with noise issues outside of the users direct control (in general, the sensor will still function well even if these things are ignored). Users are encouraged to test the sensor in their application to verify usability.

For every ranging cycle, individual filtering for that specific cycle is applied. In general, noise from regularly occurring periodic noise sources such as motors, fans, vibration, etc., will not falsely be detected as an object. This holds true even if the periodic noise increases or decreases (such as might occur in engine throttling or an increase/decrease of wind movement over the sensor). Even so, it is possible for sharp non-periodic noise sources to cause false target detection. In addition, *(because of dynamic range and signal to noise physics,) as the noise level increases, at first only small targets might be missed, but if noise increases to very high levels, it is likely that even large targets will be missed.

The I2CXL-MaxSonar-WR series has additional resistance against periodic noise and small target rejection capabilities over the standard XL-MaxSonar-WR series.

*In high noise environments, if needed, use 5V power to keep acoustic signal power high.

Target Size Compensation

The I2CXL-MaxSonar-WR/WRC sensors do not apply target size compensation to detected objects. This means that larger size targets may report as a closer distance than the actual distance or smaller size targets may report as a further distance than the actual distance.

Real-Time Auto Calibration

Each time before the I2CXL-MaxSonar-WR/WRC sensor takes a range reading it calibrates itself. The sensor then uses this data to range objects. If the humidity or applied voltage changes during sensor operation, the sensor will continue to function normally. **The sensor does not apply compensation for the speed of sound change due to temperature to any range readings**. (20C operation is assumed when calculating the speed of sound, other MaxBotix Inc., products are available with these features)

About Package Types

The I2CXL-MaxSonar-WR sensor line is available in a variety of packages for applications with specific mounting requirements. The full horn package provides peak accuracy and sensitivity in this sensor line. It is recommended that testing is completed to ensure that the selected sensor will operate as desired in your application.

Package Types Currently Available
Full Horn — 3/4" NPT straight; back mounted thread (best performance)
Compact — 3/4" NPT straight; back mounted thread
Ultra-Compact — PCB with 4 mounting holes
1"NPS — External thread over full sensor body (1"NPS)
1" BSPP — External thread over full sensor body (1"BSPP)
30mm1.5 — External thread over full sensor body (30mm1.5)
All package types have exposed PCB on user end for easy connection. Users desiring a fully enclosed assembly may purchase the "Shielded Cable Option" along with their sensor.

MaxBotix® Inc.
Copyright 2005 - 2012 MaxBotix Incorporated
Patent 7,679,996

MaxBotix Inc., products are engineered and assembled in the USA

Page 6
Web: www.maxbotix.com
PD11502e

Performance Changes when Selecting a Non-Full Horn Package

When selecting an I2CXL-MaxSonar-WR without the full horn the sensor will experience the following performance changes:
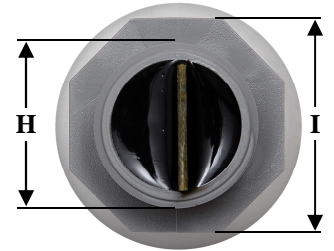
- The sensor will have a wider beam shape for the first meter.
- The sensor may have a dead zone from 0-6cm.
- The sensor may be less accurate by an additional +/- 0.5%.
- The sensor may have worse performance to small or soft targets.
- The sensor may experience decreased noise immunity when ranging to small, soft, angled, or distant targets.

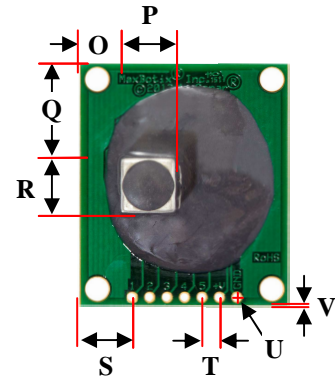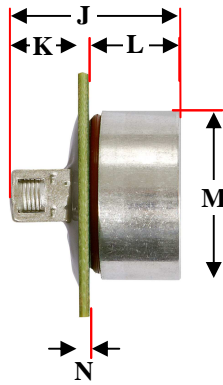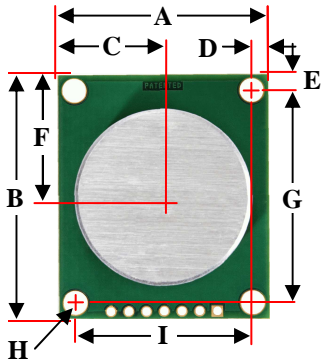# Mechanical Dimensions

Full Horn
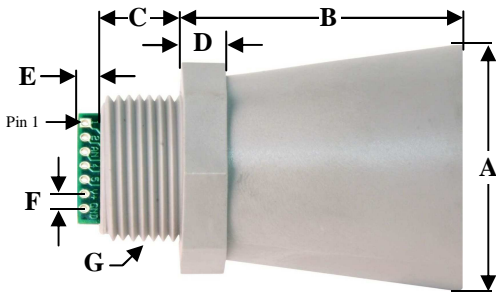
A	1.72" dia.	43.8 mm dia.
B	2.00"	50.7 mm
C	0.58"	14.4 mm
D	0.31"	7.9 mm
E	0.23"	5.8 mm
F	0.10"	2.54 mm
G	3/4"-14 NPS	
H	1.032" dia.	26.2 mm dia.
I	1.37"	34.8 mm
	Weight	50 grams

Values Are Nominal

Ultra-Compact

All values are nominal			F	17.78 mm	0.70"	L	13.4 mm	0.53"	R	7.80 mm	0.31"
A	30.48 mm	1.20"	G	30.48 mm	1.20"	M	25.0 mm	0.98"	S	7.62 mm	0.30"
B	35.56 mm	1.40"	H	3.18 mm	0.13"	N	1.57 mm	0.62"	T	2.54 mm	0.10"
C	12.24 mm	0.60"	I	25.4 mm	1.00"	O	6.2 mm	0.24"	U	1.07 mm	0.04"
D	2.54 mm	0.10"	J	25.27 mm	1.00"	P	8.0 mm	0.31"	V	1.27 mm	0.05"
E	2.54 mm	0.10"	K	10.4 mm	0.41"	Q	13.9 mm	0.55"	**Weight 15.1 gram.**		

Mechanical Dimensions Continued
Compact Housing
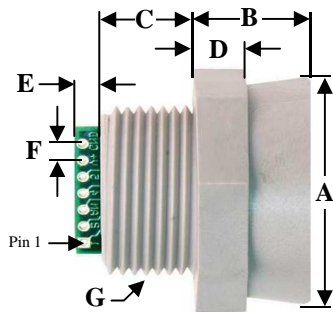
A	1.37" dia.	34.7 mm dia.
B	0.70"	17.9 mm
C	0.57"	14.4 mm
D	0.31"	7.9 mm
E	0.23"	5.8 mm
F	0.10"	2.54 mm
G	3/4"-14 NPS	
H	1.032" dia.	26.2 mm dia.
I	1.37"	34.8 mm
Weight	32 grams	
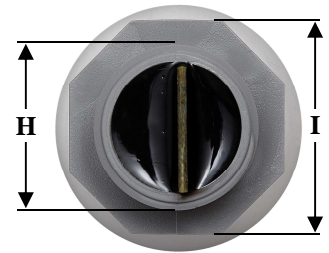
Values Are Nominal

1" NPS Pipe Threading
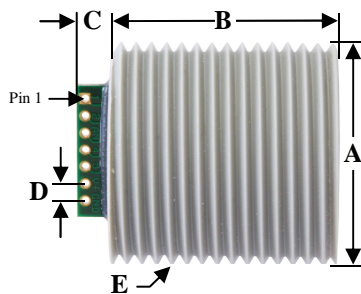
A	1.29" dia.	38.2 mm dia.
B	1.30"	33.1 mm
C	0.20"	5.2 mm
D	0.10"	2.54 mm
E	1"— NPS	
F	0.78"	19.8 mm
Weight	35 grams	
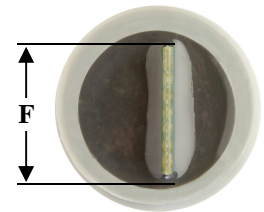
Values Are Nominal

1" BSPP Pipe Threading
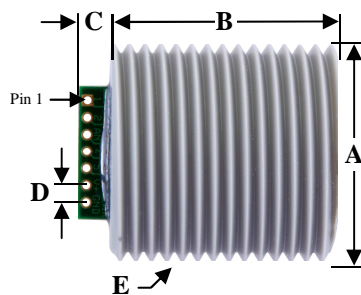
A	1.29" dia.	38.2 mm dia.
B	1.30"	33.1 mm
C	0.20"	5.2 mm
D	0.10"	2.54 mm
E	1"— BSPP	
F	0.78"	19.8 mm
Weight	34 grams	
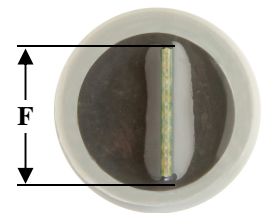
Values Are Nominal

30mm1.5 Pipe Threading
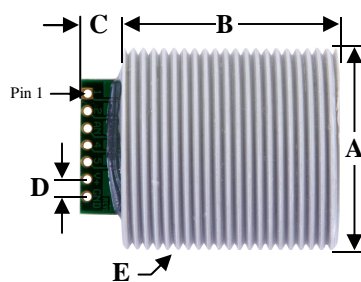
A	1.17" dia.	29.7 mm dia.
B	1.30"	33.1 mm
C	0.20"	5.2 mm
D	0.10"	2.54 mm
E	30mm 1.5	
F	0.78"	19.8 mm
Weight	31 grams	
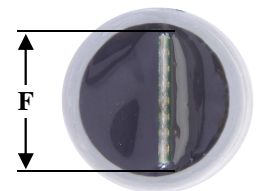
Values Are Nominal

Power-Up Timing

The I2CXL-MaxSonar-WR/WRC starts operating within milliseconds of application of power. The major timing of power-up events for the I2CXL-MaxSonar-WR/WRC can be seen in the diagram below.

Power-up Timing Diagram

Pin 6-V+	Vcc — **Power applied** — Gnd
Pin 2-Address Announce / Status	Vcc — **Pulse Width between 200uS and 25.4ms** — Gnd
Pin 3-Temporary Default	Vcc — **Pulled high unless grounded by external source** Note: Checked for external grounding here — Gnd

Sensor ready for I2C

0ms   ~70ms   ~90ms   ~115ms   ~135ms
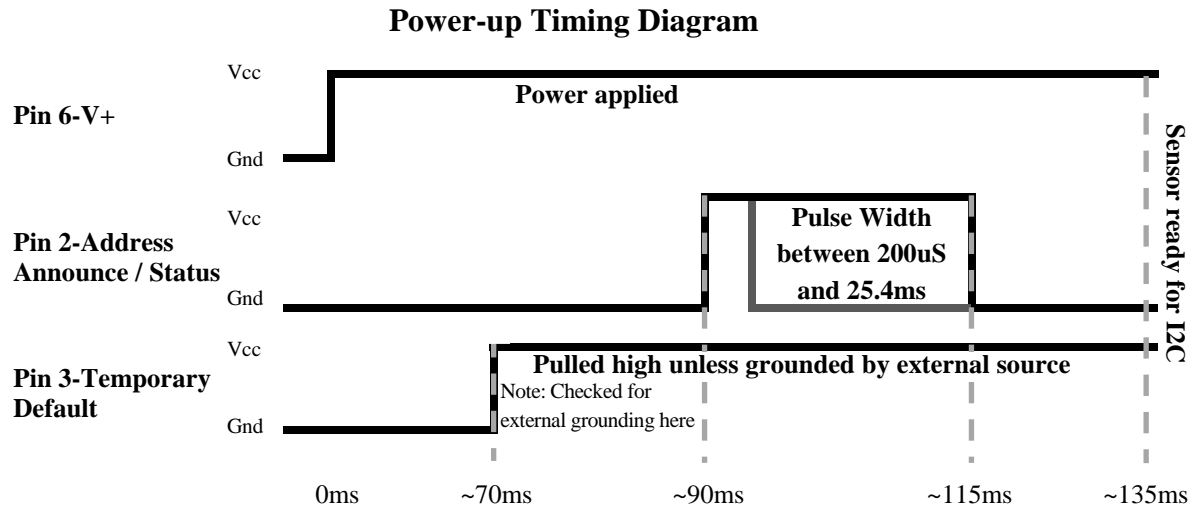
After the sensor is commanded to take a range reading it sends an ultrasonic pulse, waits between ~15ms to ~70ms to detect a target, and determines the range Then the sensor will resume I2C communications. If the sensor is addressed while in the middle of a range reading and the sensor has not yet determined a target to report, the sensor will stop the ranging process. If no range has been found the communication will be responded with a NACK (not acknowledge).

In environments that reflect acoustic noise well, sampling faster than 10Hz could cause the sensor to pick up signals from previous ultrasonic pulses and report false data. It is possible, however, to take range readings at a significantly faster rate in certain environments.

When changing the part address, ensure that power to the sensor is not disrupted or memory corruption may occur. If the memory becomes corrupted, the part should automatically use the default shipped address on power up. It is recommended to avoid changing the address often, as it could cause premature memory failure due to repeated erase/write cycles.

_____

Background Information Regarding our Beam Patterns

Each I2CXL-MaxSonar-WR/WRC sensor has a calibrated beam pattern. Each sensor is matched to provide the approximate detection pattern shown in this datasheet. This allows end users to select the part number that matches their given sensing application. Each part number has a consistent field of detection so additional units of the same part number will have similar beam patterns. The beam plots are provided to help identify an estimated detection zone for an application based on the acoustic properties of a target versus the plotted beam patterns.

Each beam pattern is a 2D representation of the detection area of the sensor. The beam pattern is actually shaped like a 3D cone (having the same detection pattern both vertically and horizontally). Detection patterns for dowels are used to show the beam pattern of each sensor.   Dowels are long cylindered targets of a given diameter. The dowels provide consistent target detection characteristics for a given size target which allows easy comparison of one MaxSonar sensor to another MaxSonar sensor.

For each part number, the four patterns (A, B, C, and D) represent the detection zone for a given target size. Each beam pattern shown is determined by the sensor's part number and target size.

The actual beam angle changes over the full range. Use the beam pattern for a specific target at any given distance to calculate the beam angle for that target at the specific distance. Generally, smaller targets are detected over a narrower beam angle and a shorter distance. Larger targets are detected over a wider beam angle and a longer range.

Compared to the XL-MaxSonar-WR line, the I2CXL-MaxSonar-WR lines offers slightly tighter beam patterns compared to the same model number (i.e. I2CXL-MaxSonar-WR versus XL-MaxSonar-WR1).

People Sensing: For users that desire to detect people, the detection area to the 1-inch diameter dowel, in general, represents the area that the sensor will reliably detect people.

MaxBotix® Inc.
Copyright 2005 - 2012 MaxBotix Incorporated
Patent 7,679,996

MaxBotix Inc., products are engineered and assembled in the USA

Page 9
Web: www.maxbotix.com
PD11502e

MB7040: I2CXL-MaxSonar®-WR™

The I2CXL-MaxSonar-WR is the recommended weather-resistant sensor. This sensor is ideal for environments with multiple sensors where the I2C interface simplifies wiring and hook-up. It features a long, narrow beam pattern with high sensitivity.

MB7040
I2CXL-MaxSonar®-WR™ Beam Pattern

Sample results for measured beam pattern are shown on a 30-cm grid. The detection pattern is shown for dowels of varying diameters that are placed in front of the sensor.

A 6.1-mm (0.25-inch) diameter dowel
B 2.54-cm (1-inch) diameter dowel
C 8.89-cm (3.5-inch) diameter dowel

D 11-inch wide board moved left to right with the board parallel to the front sensor face. This shows the sensor's range capability.
Note: For people detection the pattern typically falls between charts A and B.

- - **Partial Detection**
— **5.0 V**
● **3.3 V**

D
— 1050 cm (~34 ft.)
— 900 cm (~30 ft.)
— 750 cm (~25 ft.)
— 600 cm (~20 ft.)
— 450 cm (~15 ft.)
— 300 cm (~10 ft.)
— 150 cm (~5 ft.)
— 30 cm (~1 ft.)

A   **B**   **C**

Beam Characteristics are Approximate
Beam Patterns drawn to a 1:95 scale for easy comparison to our other products.

MB7040 Features and Benefits

- Low power consumption
- Narrow, controlled beam
- Easy to use interface
- Detects smaller objects
- Can be powered by many different types of power sources

MB7040 Applications and Uses

- People detection
- Security
- Motion detection
- Useable with battery power
- Autonomous navigation
- Educational and hobby robotics
- Collision avoidance

MaxBotix® Inc.
Copyright 2005 - 2012 MaxBotix Incorporated
Patent 7,679,996

MaxBotix Inc., products are engineered and assembled in the USA

Page 10
Web: www.maxbotix.com
PD11502e

MB704X I2CXL-MaxSonar-WRC Beam Pattern and Uses

The XL-MaxSonar-WR product line is available in alternative housings that include an Ultra-Compact form factor, WRC form factor, 1" NPS pipe threading, 1" BSPP pipe threading, and 30mm 1.5 pipe threading

D

MB7040-WRC, MB7040-1" NPS,   MB7040-1" BSPP, MB7040-UC,   MB7040-30mm1.5

Sample results for measured beam pattern are shown on a 30-cm grid. The detection pattern is shown for dowels of varying diameters that are placed in front of the sensor.
A 6.1-mm (0.25-inch) diameter dowel
B 2.54-cm (1-inch) diameter dowel
C 8.89-cm (3.5-inch) diameter dowel
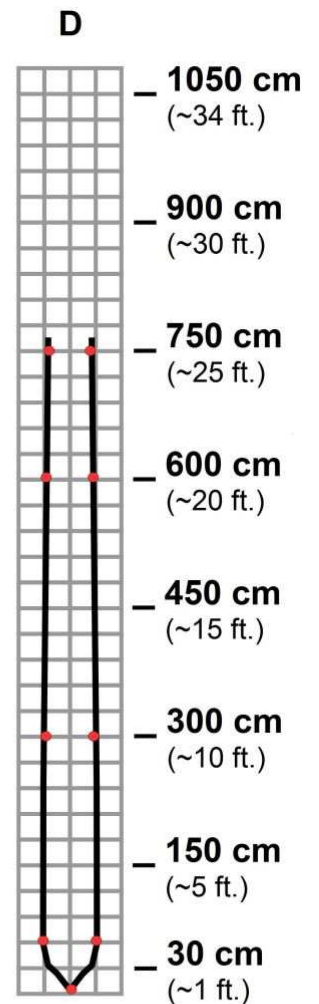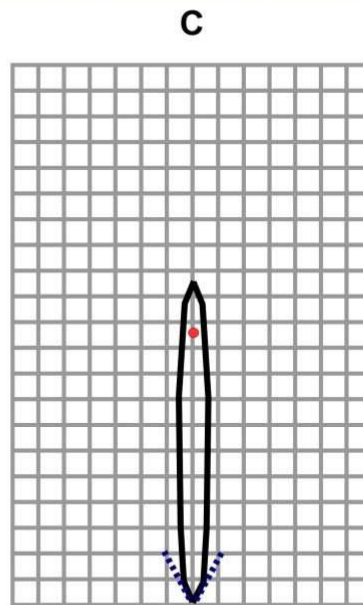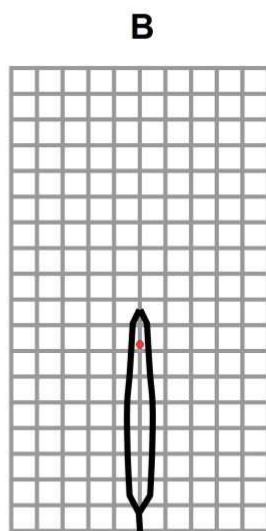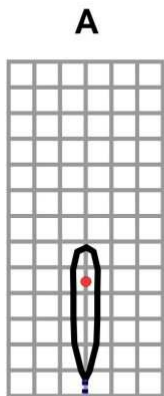**D** 11-inch wide board moved left to right with the board parallel to the front sensor face. This shows the sensor's range capability.
Note: For people detection the pattern typically falls between charts A and B.

C

— 5.0 V
● 3.3 V

A

B

1050 cm (~34 ft.)

900 cm (~30 ft.)

750 cm (~25 ft.)

600 cm (~20 ft.)

450 cm (~15 ft.)

300 cm (~10 ft.)

150 cm (~5 ft.)

30 cm (~1 ft.)
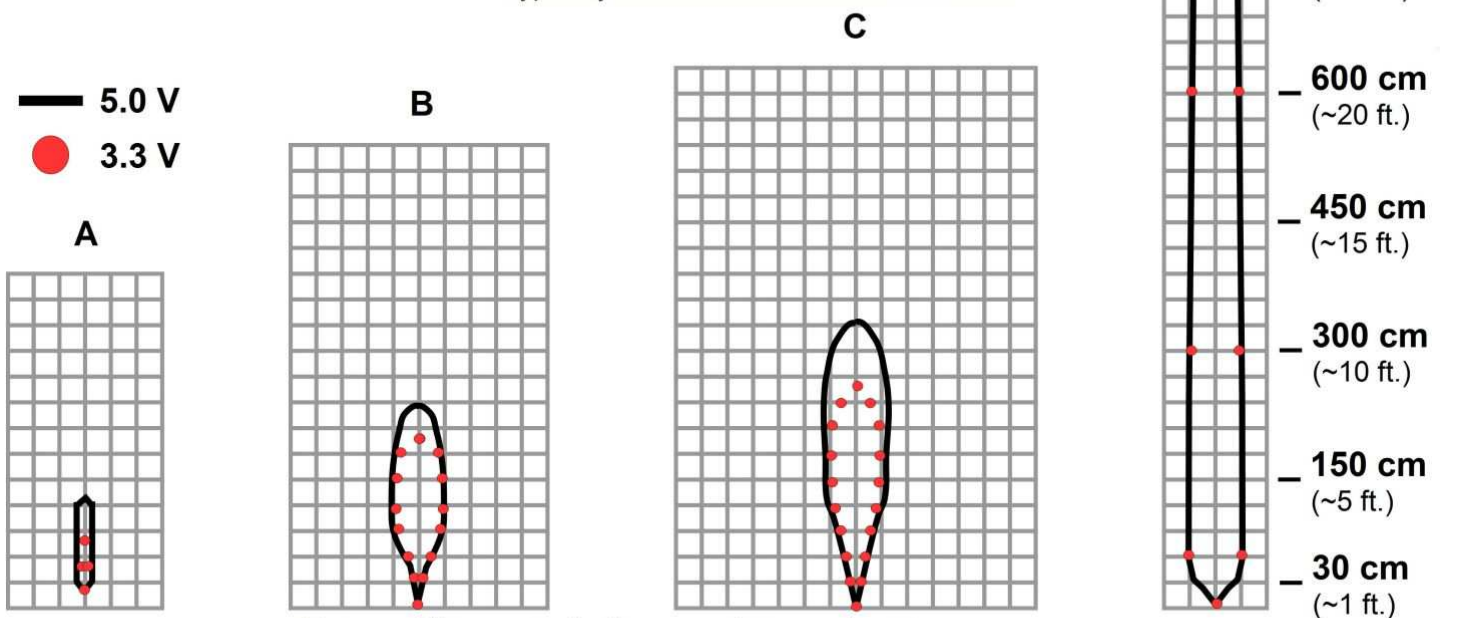
Beam Characteristics are Approximate
Beam Pattern drawn to a 1:95 scale for easy comparison to our other products.

MB704X Features and Benefits

- Low power consumption
- Easy to use interface
- Can be powered by many different types of power sources
- Can detect people up to approximately 6 feet
- Can be flush mounted in an application
- Same resolution as the full horn

- Mounting is available for both metric and imperial standards

MB704X Applications and Uses

- Security
- Motion detection
- Useable with battery power
- Autonomous navigation
- Educational and hobby robotics
- Collision avoidance

I2C Code Examples

Arduino Uno (as of Arduino 1.0.6)

```
/*                    Arduino I2C for a MaxSonar                      */
//////////////////////////////////////////////////////////////////////
//  Arduino I2C for a MaxSonar by Carl Myhre is licensed under a      //
//  Creative Commons Attribution-ShareAlike 4.0 International License. //
//  Original Author:  Carl Myhre, 10-02-2014, Revision: 1.0           //
//  Modifications by:                                                 //
//                                                                    //
//  Revision History: 1.0 -- 10-02-2014 -- Created initial code build //
//                                                                    //
//  The original I2C libraries were created by Peter Fleury           //
//     http://homepage.hispeed.ch/peterfleury/avr-software.html       //
//                                                                    //
//  These libraries were adapted by Bernhard Nebel for use on Arduino //
//     https://github.com/felias-fogg/SoftI2CMaster                   //
//                                                                    //
//  Special Thanks to MaxBotix Inc. for sponsoring this project!      //
//     http://www.maxbotix.com -- High Performance Ultrasonic Sensors //
//                                                                    //
//  For more information on installing the I2C libraries for Arduino  //
//     visit http://playground.arduino.cc/Main/SoftwareI2CLibrary     //
//////////////////////////////////////////////////////////////////////


//Hints on installing this code:
// 1. You will need to install the <SoftI2CMaster.h> library before using this code.
//      On Windows, the files are placed in C:\Program Files (x86)\Arduino\libraries\SoftI2CMaster\
// 2. As of 10-02-14 the Arduino library page (reference above) has the wrong name for the include file
//      it lists <SoftI2C.h> instead of <SoftI2CMaster.h> -- use the one that matches your installation.
// 3. Make sure to load the library into the Arduino compiler.
//      To do this go to: SKETCH >> IMPORT LIBRARY... >> ADD LIBRARY...
//      Then navigate to C:\Program Files (x86)\Arduino\libraries\SoftI2CMaster\SoftI2CMaster.h
// 4. Be sure to set the SCL and SDA pins so that they match the pins you are using.
// 5. I have included 3 working "code examples" which differ from the 3 "functions" I included.
//      The functions are all that should be required to quickly use the I2C library to talk to a MaxSonar.
//      The three code examples show how I would implement each of the common tasks you may wish to do.
// 6. The included functions are as follows:
//      A. start_sensor(addr)
//      B. read_sensor(addr)
//      C. change_address(oldaddr,newaddr)
// 7. The included code examples are as follows:
//      A. read_the_sensor_example()
//      B. address_polling_example()
//      C. default_address_change_example()
```

MaxBotix Inc., products are engineered and assembled in the USA

I2C Code Examples

Arduino Uno (as of Arduino 1.0.6)

```
/*
Below, I define the SCL and SDA pins by their ATMEGA pins I have included links to common mappings below.
    UNO:  http://arduino.cc/en/Hacking/PinMapping168
    NANO: (matches UNO but has fewer pins)
    MEGA 2560: http://arduino.cc/en/Hacking/PinMapping2560
The current data matches the setup for the Arduino Uno -- they may need to be changed if the hardware changes.
You can also switch the I2C interface
to any of the tristate pins that you want (not just the SDA or SCL pins).
*/
#define SCL_PIN 5            //Default SDA is Pin5 PORTC for the UNO -- you can set this to any tristate pin
#define SCL_PORT PORTC
#define SDA_PIN 4            //Default SCL is Pin4 PORTC for the UNO -- you can set this to any tristate pin
#define SDA_PORT PORTC
#define I2C_TIMEOUT 100      //Define a timeout of 100 ms -- do not wait for clock stretching longer than this time


/*
I have included a couple of extra useful settings for easy reference.
//#define I2C_CPUFREQ (F_CPU/8)//Useful if you plan on doing any clock switching
#define I2C_FASTMODE 1       //Run in fast mode (400 kHz)
#define I2C_SLOWMODE 1       //If you do not define the mode it will run at 100kHz with this define set to 1 it
                             //will run at 25kHz
*/
#include <SoftI2CMaster.h>   //You will need to install this library

void setup(){
  // Initialize both the serial and I2C bus
  Serial.begin(9600);
  i2c_init();

  // (OPTIONAL) Check each address for a sensor
  address_polling_example();

  /*
    Note that I placed the address change example in setup() for a good reason.
    Changing the sensor address causes an EEPROM write, there should only be ~1,000,000+
      of these writes to the sensor microcontroller over its product lifetime.
    Changing the address is fine, but doing it every second for the next 4 years may
      cause reliability issues.
  */
  // (OPTIONAL) Run an address change example
  default_address_change_example();

  // Your code here
}

void loop()
{
  // (OPTIONAL) Read a sensor at the default address
  read_the_sensor_example();

  // Your code here

}
```

MaxBotix Inc., products are engineered and assembled in the USA

I2C Code Examples

Arduino Uno (as of Arduino 1.0.6)

```
///////////////////////////////////////////////
// Function: Start a range reading on the sensor //
///////////////////////////////////////////////
//Uses the I2C library to start a sensor at the given address
//Collects and reports an error bit where: 1 = there was an error or 0 = there was no error.
//INPUTS: byte bit8address = the address of the sensor that we want to command a range reading
//OUPUTS: bit  errorlevel = reports if the function was successful in taking a range reading: 1 = the function
//      had an error, 0 = the function was successful
boolean start_sensor(byte bit8address){
  boolean errorlevel = 0;
  bit8address = bit8address & B11111110;              //Do a bitwise 'and' operation to force the last bit to be
                                                      //zero -- we are writing to the address.
  errorlevel = !i2c_start(bit8address) | errorlevel;  //Run i2c_start(address) while doing so, collect any errors
                                                      //where 1 = there was an error.
  errorlevel = !i2c_write(81) | errorlevel;           //Send the 'take range reading' command. (notice how the
                                                      //library has error = 0 so I had to use "!" (not) to invert
                                                      //the error

  i2c_stop();
  return errorlevel;
}




///////////////////////////////////////////////////////////////////////
// Function: Read the range from the sensor at the specified address //
///////////////////////////////////////////////////////////////////////
//Uses the I2C library to read a sensor at the given address
//Collects errors and reports an invalid range of "0" if there was a problem.
//INPUTS: byte  bit8address = the address of the sensor to read from
//OUPUTS: int   range = the distance in cm that the sensor reported; if "0" there was a communication error
int read_sensor(byte bit8address){
  boolean errorlevel = 0;
  int range = 0;
  byte range_highbyte = 0;
  byte range_lowbyte = 0;
  bit8address = bit8address | B00000001;  //Do a bitwise 'or' operation to force the last bit to be 'one' -- we are
                                          //reading from the address.
  errorlevel = !i2c_start(bit8address) | errorlevel;
  range_highbyte = i2c_read(0);           //Read a byte and send an ACK (acknowledge)
  range_lowbyte  = i2c_read(1);           //Read a byte and send a NACK to terminate the transmission
  i2c_stop();
  range = (range_highbyte * 256) + range_lowbyte;  //compile the range integer from the two bytes received.
  if(errorlevel){
    return 0;
  }
  else{
    return range;
  }
}
```

I2C Code Examples

Arduino Uno (as of Arduino 1.0.6)

```
/////////////////////////////////////////////////
// Function: Start a range reading on the sensor //
/////////////////////////////////////////////////
//Uses the I2C library to start a sensor at the given address
//Collects and reports an error bit where: 1 = there was an error or 0 = there was no error.
//INPUTS: byte bit8address = the address of the sensor that we want to command a range reading
//OUPUTS: bit  errorlevel = reports if the function was successful in taking a range reading: 1 = the function
//      had an error, 0 = the function was successful
boolean start_sensor(byte bit8address){
  boolean errorlevel = 0;
  bit8address = bit8address & B11111110;              //Do a bitwise 'and' operation to force the last bit to be
                                                      //zero -- we are writing to the address.
  errorlevel = !i2c_start(bit8address) | errorlevel;  //Run i2c_start(address) while doing so, collect any errors
                                                      //where 1 = there was an error.
  errorlevel = !i2c_write(81) | errorlevel;           //Send the 'take range reading' command. (notice how the
                                                      //library has error = 0 so I had to use "!" (not) to invert
                                                      //the error

  i2c_stop();
  return errorlevel;
}




/////////////////////////////////////////////////////////////////////
// Function: Read the range from the sensor at the specified address //
/////////////////////////////////////////////////////////////////////
//Uses the I2C library to read a sensor at the given address
//Collects errors and reports an invalid range of "0" if there was a problem.
//INPUTS: byte  bit8address = the address of the sensor to read from
//OUPUTS: int   range = the distance in cm that the sensor reported; if "0" there was a communication error
int read_sensor(byte bit8address){
  boolean errorlevel = 0;
  int range = 0;
  byte range_highbyte = 0;
  byte range_lowbyte = 0;
  bit8address = bit8address | B00000001;  //Do a bitwise 'or' operation to force the last bit to be 'one' -- we are
                                          //reading from the address.
  errorlevel = !i2c_start(bit8address) | errorlevel;
  range_highbyte = i2c_read(0);          //Read a byte and send an ACK (acknowledge)
  range_lowbyte  = i2c_read(1);          //Read a byte and send a NACK to terminate the transmission
  i2c_stop();
  range = (range_highbyte * 256) + range_lowbyte;  //compile the range integer from the two bytes received.
  if(errorlevel){
    return 0;
  }
  else{
    return range;
  }
}
```

MaxBotix® Inc.
MaxBotix Inc., products are engineered and assembled in the USA
Page 15
Web: www.maxbotix.com
PD11502e

I2C Code Examples

Arduino Uno (as of Arduino 1.0.6)

```
/////////////////////////////////////////////////////////
// Code Example: Poll all possible addresses to find a sensor //
/////////////////////////////////////////////////////////
void address_polling_example(){
  boolean error = 0;  //Create a bit to check for catch errors as needed.
  int range = 0;
  Serial.println("Polling addresses...");

  //Walk through all possible addresses and check for a device that can receive the range command and will
  //    return two bytes.
  for (byte i=2; i!=0; i+=2){   //start at 2 and count up by 2 until wrapping to 0. Checks all addresses (2-254)
                                //except 0 (which cannot be used by a device)

    error = 0;
    error = start_sensor(i);    //Start the sensor and collect any error codes.
    if (!error){                //If you had an error starting the sensor there is little point in reading it.
      delay(100);
      range = read_sensor(i);   //reading the sensor will return an integer value -- if this value is 0 there was
                                //an error
      Serial.println(i);
      if (range != 0){
        Serial.print("Device found at:");Serial.print(i);Serial.print(" Reported value of:");Serial.println(range);
      }
    }
    else{
      Serial.print("Couldn't start:");Serial.println(i);
    }
  }

  Serial.println("Address polling complete.");
}




/////////////////////////////////////////////
// Code Example: Change the default address //
/////////////////////////////////////////////
void default_address_change_example(){
  boolean error = 0;  //Create a bit to check for catch errors as needed.
  int range;

  Serial.println("Take a reading at the default address");

  //Take a range reading at the default address of 224
  error = start_sensor(224);     //Start the sensor and collect any error codes.
  if (!error){                   //If you had an error starting the sensor there is little point in reading it.
    delay(100);
    range = read_sensor(224);    //reading the sensor will return an integer value -- if this value is 0 there was
                                 //an error
    Serial.print("R:");Serial.println(range);
  }

   Serial.println("Change the sensor at the default address to 222");
```

MaxBotix® Inc.
Copyright 2005 - 2012 MaxBotix Incorporated
Patent 7,679,996

MaxBotix Inc., products are engineered and assembled in the USA

Page 16
Web: www.maxbotix.com
PD11502e

I2C Code Examples

Arduino Uno (as of Arduino 1.0.6)

```
  //Change the address from 224 to 222
  error = 0;
  error = change_address(224,222);  //Change the address -- I don't do anything with the error handler at this point
                                     //but you can if you want.
  delay(200);    //Wait 125ms for the sensor to save the new address and reset

   Serial.println("Take a reading at the new address");

  //Take a range reading at the new address of 222
  error = 0;
  error = start_sensor(222);     //Same as above but at the new address
  if (!error){
    delay(100);
    range = read_sensor(222);
    Serial.print("N:");Serial.println(range);
  }

   Serial.println("Change the sensor back to the default address");

  //Change the address from 222 to 224
  error = 0;
  error = change_address(222,224);
  delay(200);    //Wait 125ms for the sensor to save the new address and reset
}
```

Have the right sensor for your application?

Select from this product list for Protected and Non-Protected Environments.

Protected Environments

1 mm Resolution
HRLV-MaxSonar-EZ

1 in Resolution
LV-MaxSonar-EZ
LV-ProxSonar-EZ

1 cm Resolution
XL-MaxSonar-EZ
XL-MaxSonar-AE
XL-MaxSonar-EZL
XL-MaxSonar-AEL

1 mm Resolution
HRUSB-MaxSonar-EZ
1 in Resolution
USB-ProxSonar-EZ

Non-Protected Environments

1 mm Resolution
HRXL-MaxSonar-WR
HRXL-MaxSonar-WRS
HRXL-MaxSonar-WRT
HRXL-MaxSonar-WRM
HRXL-MaxSonar-WRMT
HRXL-MaxSonar-WRL
HRXL-MaxSonar-WRLT
HRXL-MaxSonar-WRLS
HRXL-MaxSonar-WRLST
SCXL-MaxSonar-WR
SCXL-MaxSonar-WRS
SCXL-MaxSonar-WRT
SCXL-MaxSonar-WRM
SCXL-MaxSonar-WRMT
SCXL-MaxSonar-WRL
SCXL-MaxSonar-WRLT
SCXL-MaxSonar-WRLS
SCXL-MaxSonar-WRLST
4-20HR-MaxSonar-WR

1 cm Resolution
XL-MaxSonar-WR
XL-MaxSonar-WRL
XL-MaxSonar-WRA
XL-MaxSonar-WRLA
I2CXL-MaxSonar-WR

1 mm Resolution
HRXL-MaxSonar-WRC
HRXL-MaxSonar-WRCT

1 cm Resolution
XL-MaxSonar-WRC
XL-MaxSonar-WRCA
I2CXL-MaxSonar-WRC

1 cm Resolution
UCXL-MaxSonar-WR
UCXL-MaxSonar-WRC
I2C-UCXL-MaxSonar-WR

F-Option. Available for WR models except UCXL. For additional protection when necessary in hazardous chemical environments.

Accessories — More information is online.

MB7954 — Shielded Cable

The MaxSonar Connection Wire is used to reduce interference caused by electrical noise on the lines. This cable is a great solution to use when running the sensors at a long distance or in an area with a lot of EMI and electrical noise.

MB7950 — XL-MaxSonar-WR Mounting Hardware

The MB7950 Mounting Hardware is selected for use with our outdoor ultrasonic sensors. The mounting hardware includes a steel lock nut and two O-ring (Buna-N and Neoprene) each optimal for different applications.

MB7955 / MB7956 / MB7957 / MB7958 / MB7972 — HR-MaxTemp

The HR-MaxTemp is an optional accessory for the HR-MaxSonar. The HR-MaxTemp connects to the HR-MaxSonar for automatic temperature compensation without self heating.

MB7961 — Power Supply Filter

The power supply filter is recommended for applications with unclean power or electrical noise.
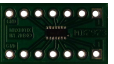
MB7962 / MB7963 / MB7964 / MB7965 — Micro-B USB Connection Cable

The MB7962, MB7963, MB7964 and MB7965 Micro-B USB cables are USB 2.0 compliant and backwards compatible with USB 1.0 standards. Varying lengths.
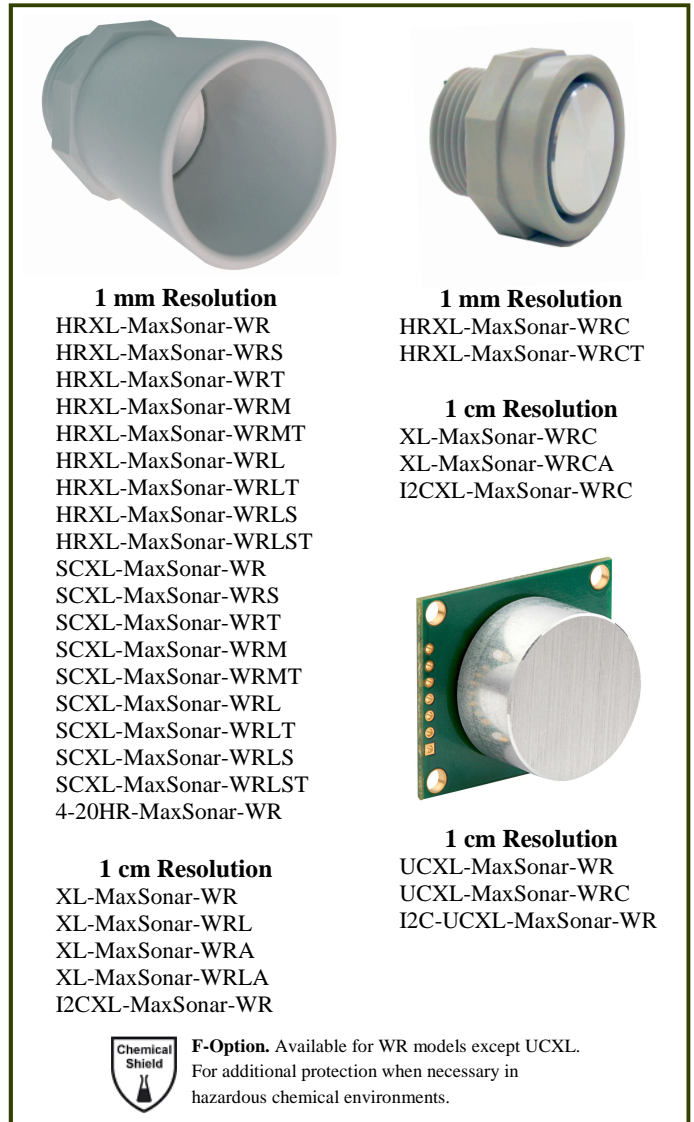
MB7973 — CE Lightning/Surge Protector

The MB7973 adds protection required to meet the Lightning/Surge IEC61000-4-5 specification.

MaxBotix® Inc.
Copyright 2005 - 2012 MaxBotix Incorporated
Patent 7,679,996

MaxBotix Inc., products are engineered and assembled in the USA

Page 18
Web: www.maxbotix.com
PD11502e