I2CXL-MaxSonar®- EZ™ Series

High Performance Sonar Rangefinder

MB1202, MB1212, MB1222, MB1232, MB1242[6]

The I2CXL-MaxSonar-EZ series offers industrial ultrasonic sensors with an easy to use I2C interface. These sensors have a high acoustic power output along with real-time auto calibration for changing conditions (voltage and acoustic or electric noise) that ensure users receive the most reliable (in air) ranging data for every reading taken. The low power 3V – 5.5V operation provides very short to long-range detection and ranging, in a compact form factor. The I2CXL-MaxSonar-EZ detect objects from 0-cm to 765-cm (25.1 feet) and provide sonar range information from 20-cm or 25-cm out to 765-cm with 1-cm resolution. Objects from 0-cm* to minimum distance typically range as minimum distance.[1] (*Objects from 0-mm to 1-mm may not be detected).* [1]***See Close Range Operation***

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 as close as 1-mm from the sensor. 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)[4] 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-EZ products
- Operational temperature from 0°C to +65°C (32F to +149F) [5]

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
- Landing flying objects
- Collision avoidance
- Bin level measurement

Notes:

[1]See Close Range Operation
[2] Users are encouraged to evaluate the sensors performance in their application
[3] By design
[4] Recommended time between readings of 100ms (10Hz Rate)
[5] Please reference page 6 for minimum operating voltage verses temperature information.
[6]Please reference page 19 for part number key.

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 for detecting objects from 0 to 20cm 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.

Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996

Downloaded from Arrow.com.

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.

I2CXL-MaxSonar-EZ Pin Out

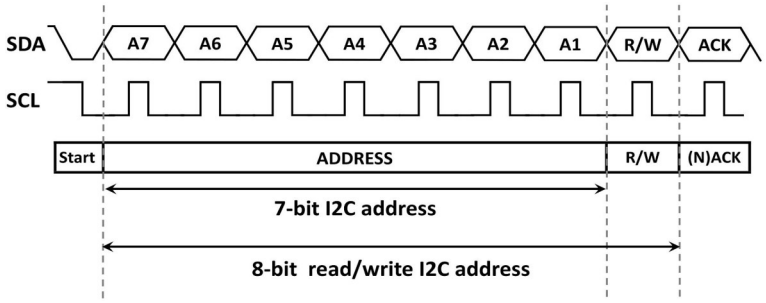
GND: Return for the DC power supply. GND (& V+) must be ripple and noise free for best operation.
V+: Operates on 3VDC to 5.5VDC. The average current draw for 3.3V operation is 2.7mA (50mA peak) and for 5V operation is 4.4mA (100mA peak) respectively. Peak current is used during sonar pulse transmit. Please reference page 6 for minimum operating voltage verses temperature information.
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.

I2CXL-MaxSonar-EZ 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 I2C-MaxSonar-EZ 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.

I2C-MaxSonar-EZ Commands



Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

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

Operating at read rates beyond 10Hz

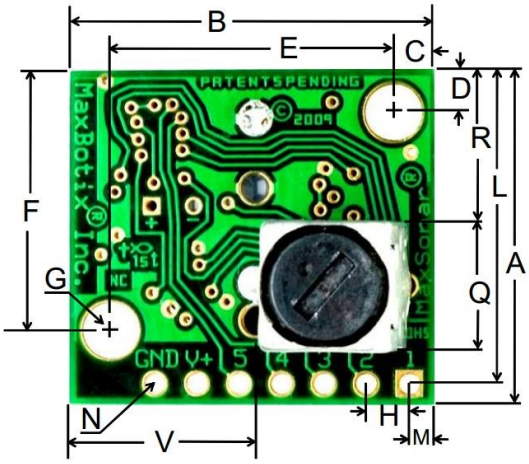
The length of a ranging cycle is proportional to the distance between the sensor and the target. If no target is found, the ranging cycle will take the longest amount of time, around 100mS. If a target is found at minimum distance the ranging cycle will take the shortest amount of time, around 25mS. The sensor cannot be interrupted while ranging and all I2C communications will be ignored during this time. After the ranging cycle is complete, the sensor will resume I2C communications. Therefore, the maximum refresh rate of the sensor is proportional to the range reported. If there is a target immediately in front of the sensor, it can operate at a very high refresh rate, and as the distance increases the maximum refresh rate will decrease.

If the user wishes to operate the sensor at the maximum possible refresh rate for any given range, they may monitor the Address Announce / Status pin which will allow them to know exactly when the sensor is ready for a new request. When the pin goes low following a range reading, the sensor is ready to resume I2C communications. If the user cannot monitor this pin, then there should be a minimum 100mS pause between range commands to allow the sensor to complete a fully cycle regardless of the reported range.

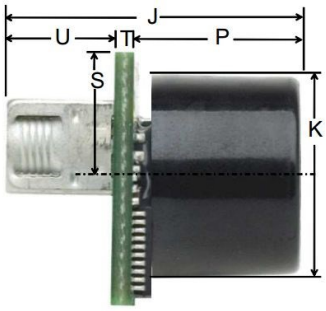
Additionally, sending I2C requests while the sensor is actively ranging could introduce additional noise into the sensor which could negatively impact performance, as such it is recommended to keep the I2C bus quiet during all ranging operations.

Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

I2C-MaxSonar-EZ Mechanical Dimensions

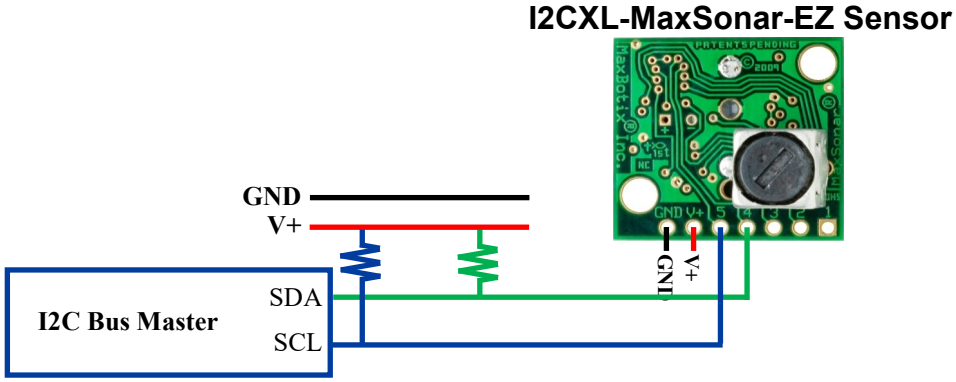


A	0.785”	19.9mm	L	0.735”	18.7mm
B	0.870”	22.1mm	M	0.065”	1.7mm
C	0.100”	2.54mm	N	0.038” dia.	1.0mm dia.
D	0.100”	2.54mm	P	0.537”	13.64mm
E	0.670”	17.0mm	Q	0.304”	7.72mm
F	0.610	15.5mm	R	0.351”	8.92mm
G	0.124” dia.	3.1mm dia.	S	0.413”	10.5mm
H	0.100”	2.54mm	T	0.063”	1.6mm
J	0.989”	25.11mm	U	0.368”	9.36mm
K	0.645”	16.4 mm	V	0.492”	12.5mm
values are nominal			Weight, 5.9 grams		



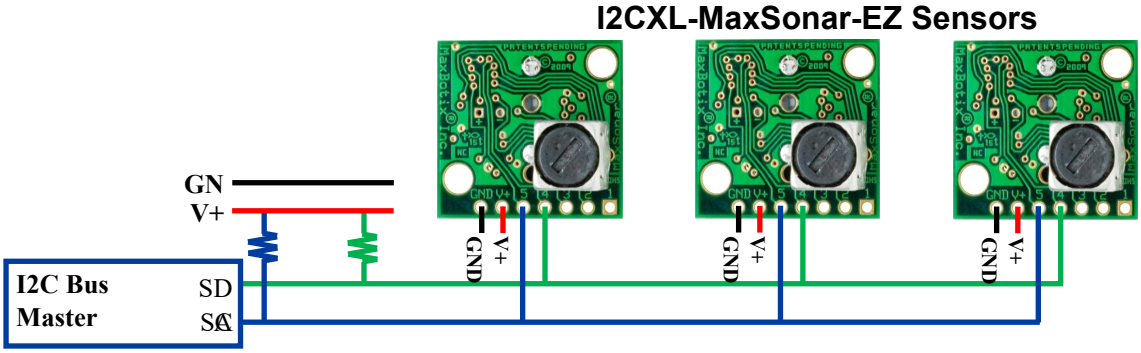
Single Sensor Wiring Diagram

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.



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



Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

Selecting a I2CXL-MaxSonar-EZ Sensor

Different applications require different sensors. The I2CXL-MaxSonar-EZ product line offers varied sensitivity to allow users to select the best sensor to meet their needs.

People Detection Wide Beam High Sensitivity	Best Balance		Large Targets Narrow Beam Noise Tolerance	Very Large Targets Narrow Beam Extreme Noise Tolerance
MB1202	MB1212	MB1222	MB1232	MB1242

The diagram above shows how each product balances sensitivity and noise tolerance. This does not affect the pin outputs or other sensor operation. To get a better view of how each sensor will function to targets of different sizes reference the I2CXL-MaxSonar-EZ beam patterns.

Sensor Minimum Distance

For the MB1202 and MB1212 the sensor minimum reported distance is 25-cm. However, the I2CXL-MaxSonar-EZ will range and report targets to the front sensor face. Large targets closer than 25-cm will typically range as 25-cm.

For the MB1222, MB1232, & MB1242 the sensor minimum reported distance is 20-cm. However, the I2CXL-MaxSonar-EZ will range and report targets to the front sensor face. Large targets closer than 20-cm will typically range as 20-cm.

Real-time Noise Rejection

While the I2CXL-MaxSonar®-EZ 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-EZ series has additional resistance against periodic noise and small target rejection capabilities over the standard XL-MaxSonar-EZ/AE series.
*In high noise environments, if needed, use 5V power to keep acoustic signal power high. In addition, a high acoustic noise environment may use some of the dynamic range of the sensor, so consider a part with less gain such as the MB1222, MB1232, or MB1242. For applications with large targets, consider a part with clutter rejection like the MB7360.

Target Size Compensation

The I2CXL-MaxSonar-EZ 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 then the actual distance.

Real-Time Auto Calibration

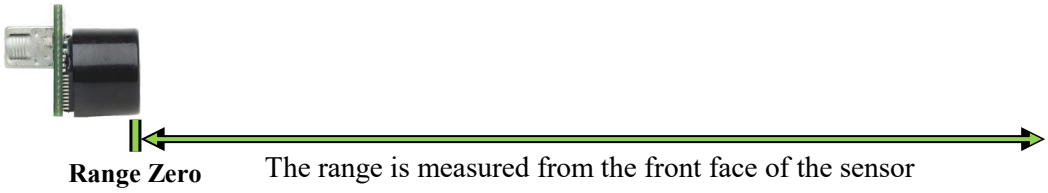
Each time before the I2CXL-MaxSonar-EZ sensor takes a range reading it calibrates itself. The sensor then uses this data to range objects. If the humidity or applied voltage changes during sensor operation, the sensor will continue to function normally. **The sensor does not apply compensation for the 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)

Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

Range "0" Location

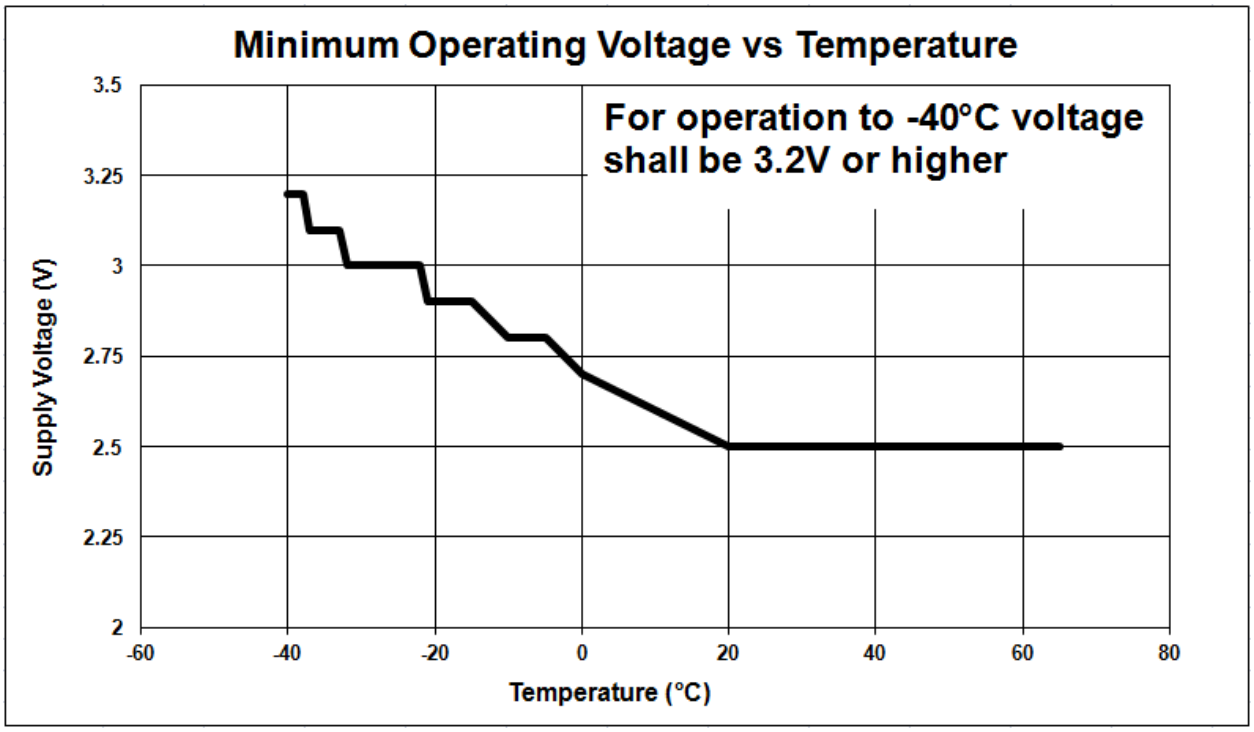
The I2CXL-MaxSonar-EZ reports the range to distant targets starting from the front of the sensor as shown in the diagram below.

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



Voltage vs Temperature

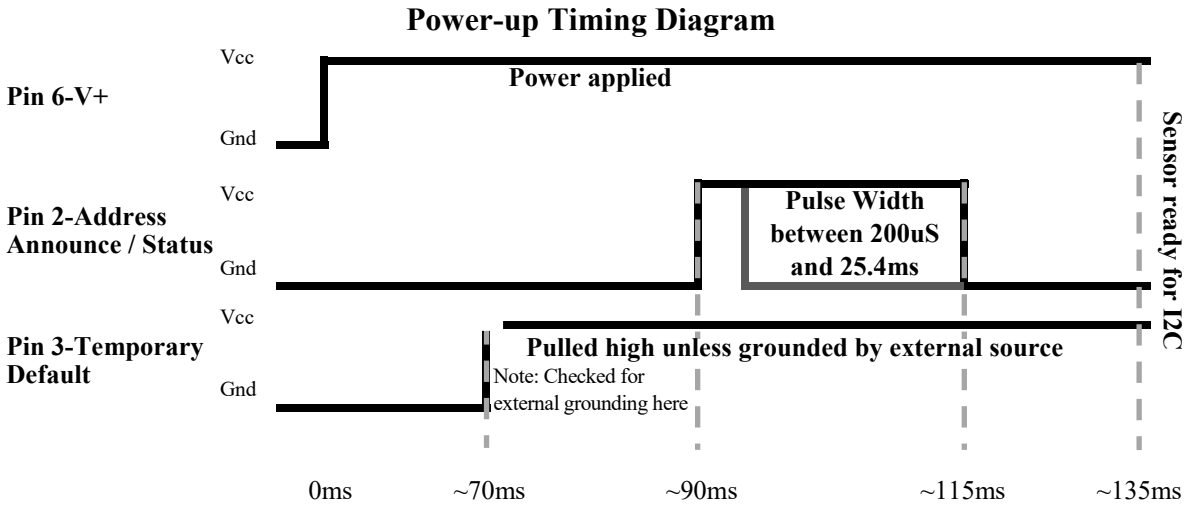
The graph below shows minimum operating voltage of the sensor verses temperature.



Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

I2CXL-MaxSonar-EZ Timing

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



After the sensor is commanded to take a range reading it sends an ultrasonic pulse, processes incoming acoustic waveforms for ~25mS to ~100mS, and then determines the range. Then the sensor will resume I2C communications. If the sensor is addressed while in the middle of a range reading, all requests for 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.

Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

I2CXL-MaxSonar®-EZ™ Beam Characteristics

Background Information Regarding our Beam Patterns

Each I2CXL-MaxSonar-EZ 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.

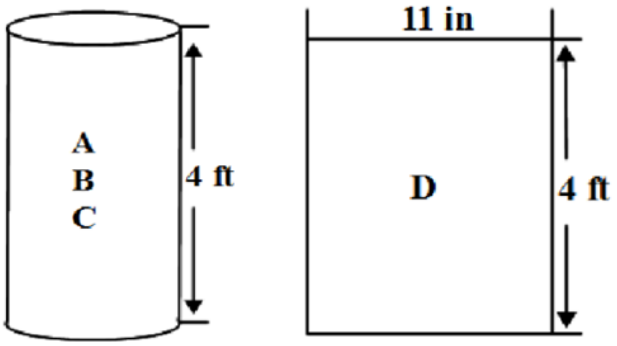
People Sensing: For users that desire to detect people, the detection area to the 1-inch diameter dowel, in general, represents the area that the sensor will reliably detect people.

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-EZ line, the I2CXL-MaxSonar-EZ lines offers slightly tighter beam patterns compared to the same model number (i.e. I2CXL-MaxSonar-EZ1 versus XL-MaxSonar-EZ1).

Beam Pattern Target Shapes

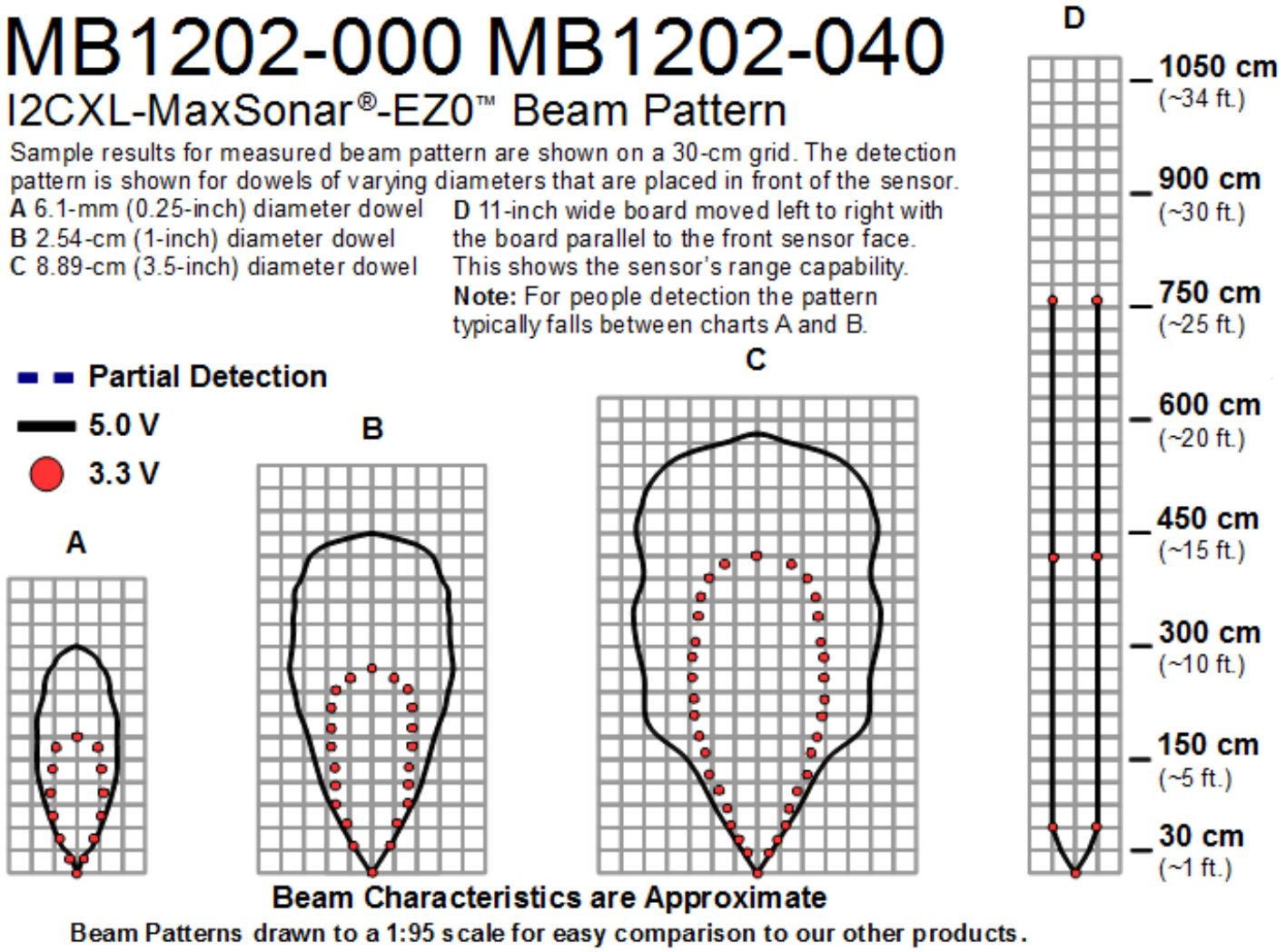
A 6.1-mm (0.25-inch) diameter dowel 4ft length
B 2.54-cm (1-inch) diameter dowel 4ft length
C 8.89-cm (3.5-inch) diameter dowel 4ft length
D 11-inch wide board 4ft in length moved left to right with the board parallel to the front sensor face. This shows the sensor's range capability.



Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

MB1202: I2CXL-MaxSonar-EZ0

The I2CXL-MaxSonar-EZ0 is the highest sensitivity and widest beam sensor of the I2CXL-MaxSonar-EZ sensor series. The wide beam makes this sensor ideal for a variety of applications including people detection, autonomous navigation, and wide beam applications.



MB1202 Features and Benefits

- Widest and most sensitive beam pattern in I2CXL-MaxSonar-EZ line
- Low power consumption
- Easy to use interface
- Will pick up the most noise clutter of any of the sensors in I2CXL-MaxSonar-EZ line
- Detects smaller objects
- Best sensor to detect soft object in I2CXL-MaxSonar-EZ line
- Can be powered by many different types of power sources
- Can detect people up to approximately 14 feet

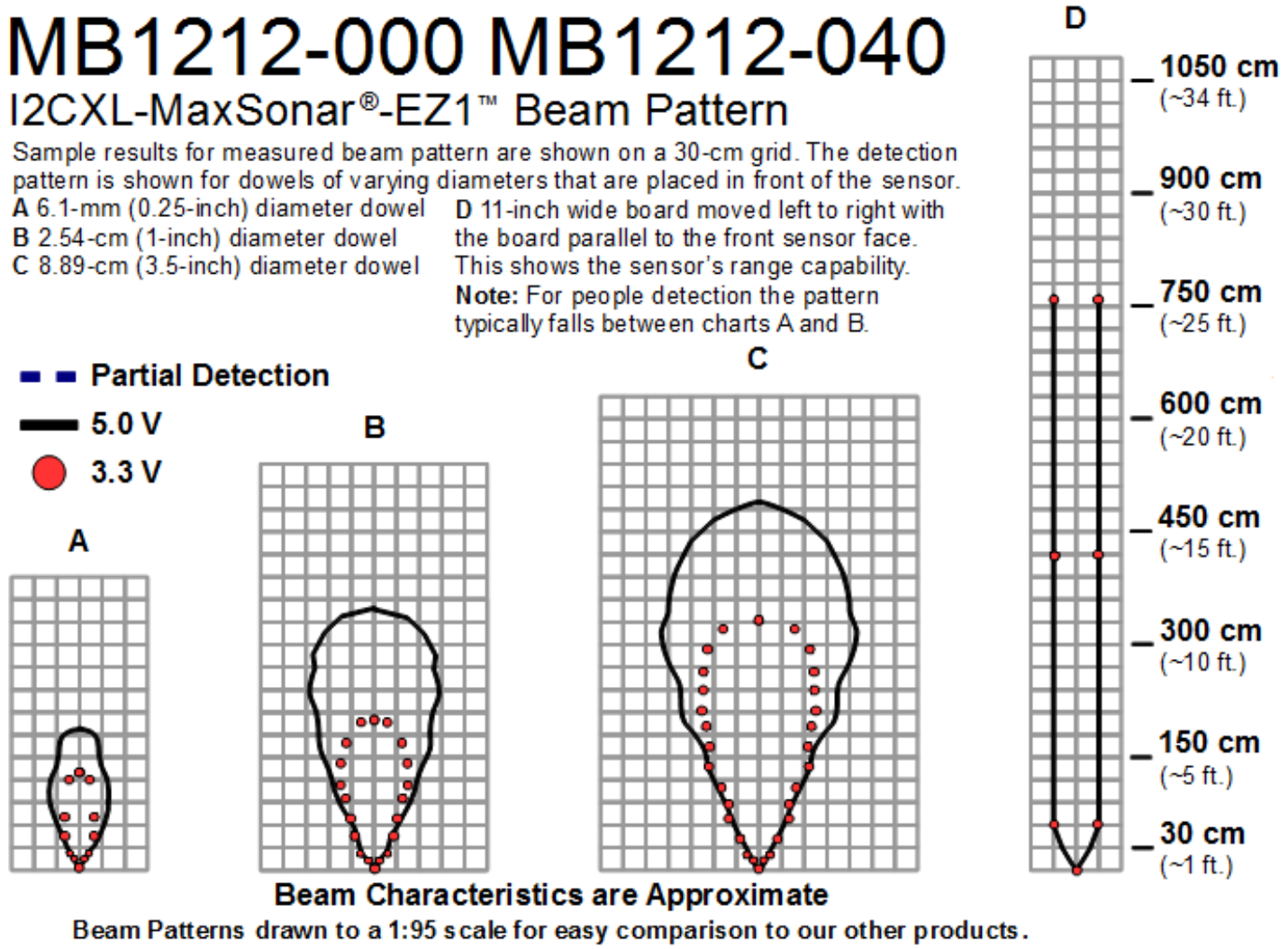
MB1202 Applications and Uses

- Great for people detection
- Security
- Motion detection
- Landing flying objects
- Useable with battery power
- Autonomous navigation
- Educational and hobby robotics
- Collision avoidance

Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

MB1212: I2CXL-MaxSonar-EZ1

The I2CXL-MaxSonar-EZ1 is our most recommended indoor sensor, and offers a lower hardware gain and a slightly smaller, narrower beam pattern than the MB1202. This makes the I2CXL-MaxSonar-EZ1 a great choice for people detection applications.



MB1212 Features and Benefits

- Wide and sensitive beam pattern in the I2CXL-MaxSonar-EZ line
- Low power consumption
- Easy to use interface
- Detects smaller objects
- Can be powered by many different types of power sources
- Great sensor to detect soft object in I2CXL-MaxSonar-EZ line
- Can detect people up to approximately 10 feet

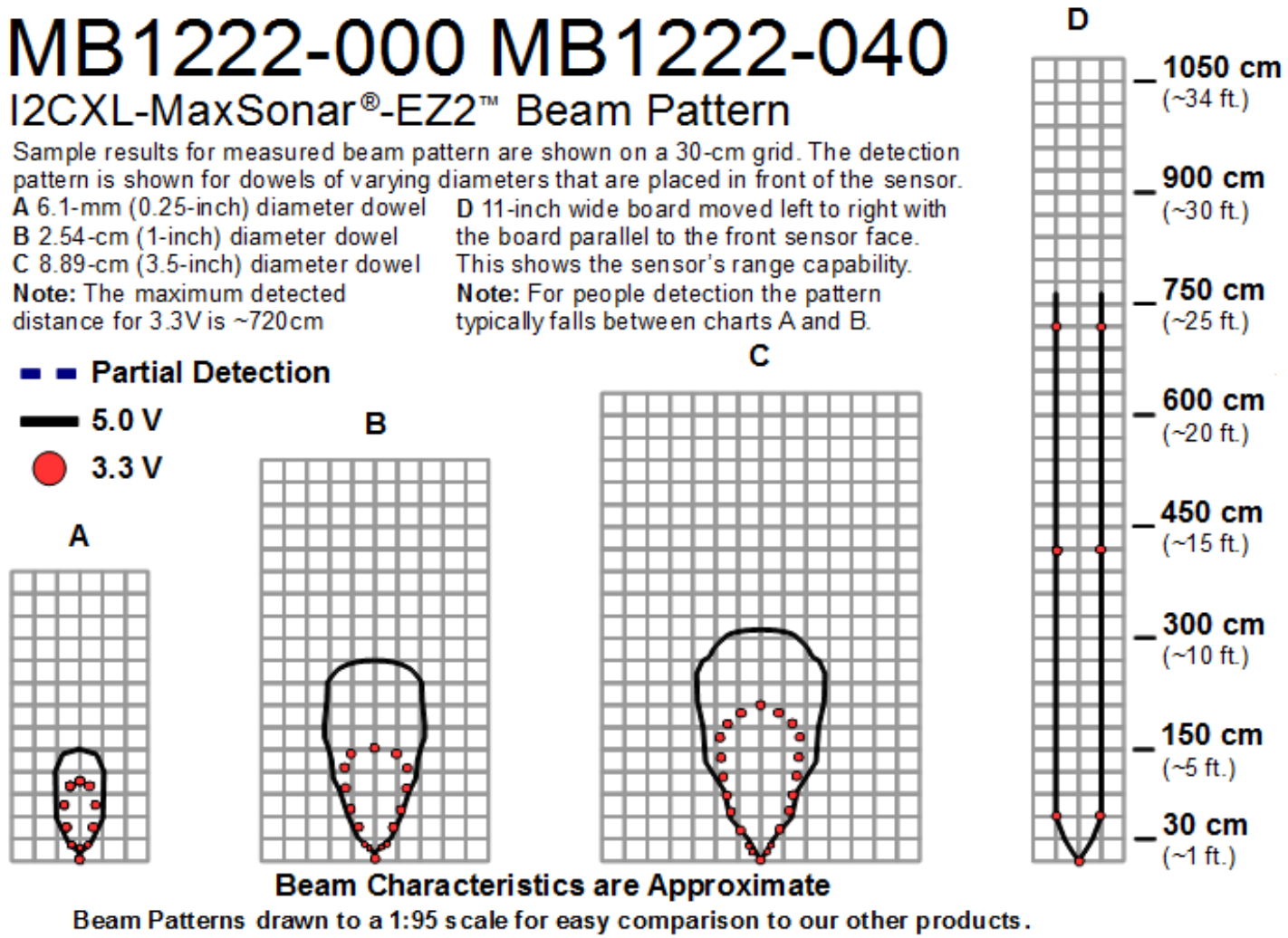
MB1212 Applications and Uses

- Great for people detection
- Security
- Motion detection
- Landing flying objects
- Useable with battery power
- Autonomous navigation
- Educational and hobby robotics
- Collision avoidance

Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

MB1222: I2CXL-MaxSonar-EZ2

The I2CXL-MaxSonar-EZ2 offers a good balance between wide and narrow beam sensors, and large and narrow object detection. The I2CXL-MaxSonar-EZ2 works for nearly all indoor applications where the wider or narrower beam of other models could be a problem, including people detection, large-target detection, long range detection, and applications requiring high noise tolerance.



MB1222 Features and Benefits

- Balanced beam pattern in I2CXL-MaxSonar-EZ line
- Low power consumption
- Easy to use interface
- Detects smaller objects
- Can be powered by many different types of power sources
- Good sensor to detect soft object in I2CXL-MaxSonar-EZ line
- Can detect people up to approximately 8 feet

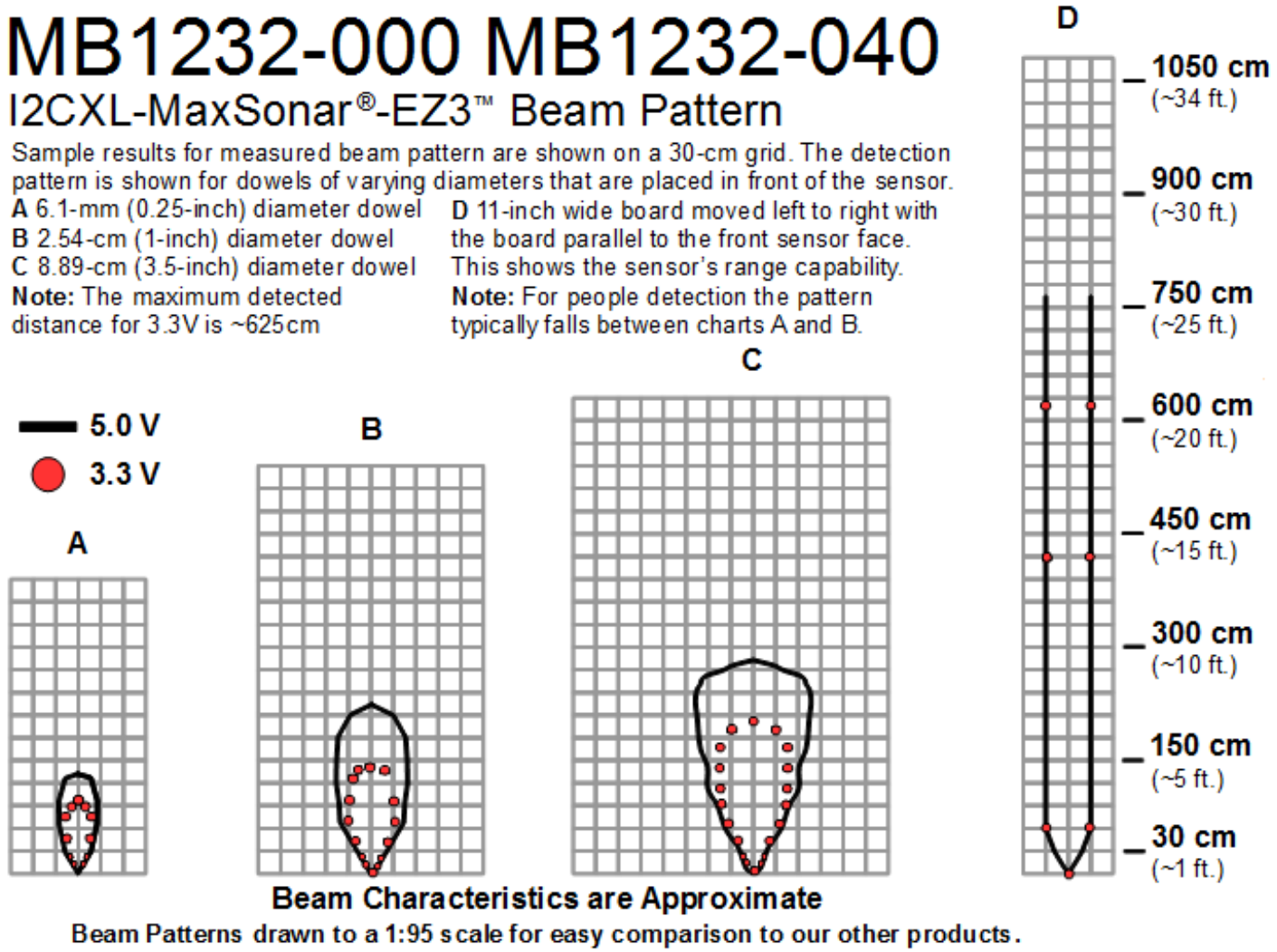
MB1222 Applications and Uses

- Great for people detection
- Security
- Motion detection
- Landing flying objects
- Useable with battery power
- Autonomous navigation
- Educational and hobby robotics
- Collision avoidance

Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

MB1232: I2CXL-MaxSonar-EZ3

The I2CXL-MaxSonar-EZ3 has high noise tolerance and a narrow beam with more sensitivity compared to the MB1242. The lower sensitivity of the I2CXL-MaxSonar-EZ3 makes it great for detecting only large targets at greater distances and filtering out smaller targets.



MB1232 Features and Benefits

- Narrow beam pattern in the I2CXL-MaxSonar-EZ line
- Low power consumption
- Easy to use interface
- Rejects small clutter
- Can be powered by many different types of power sources

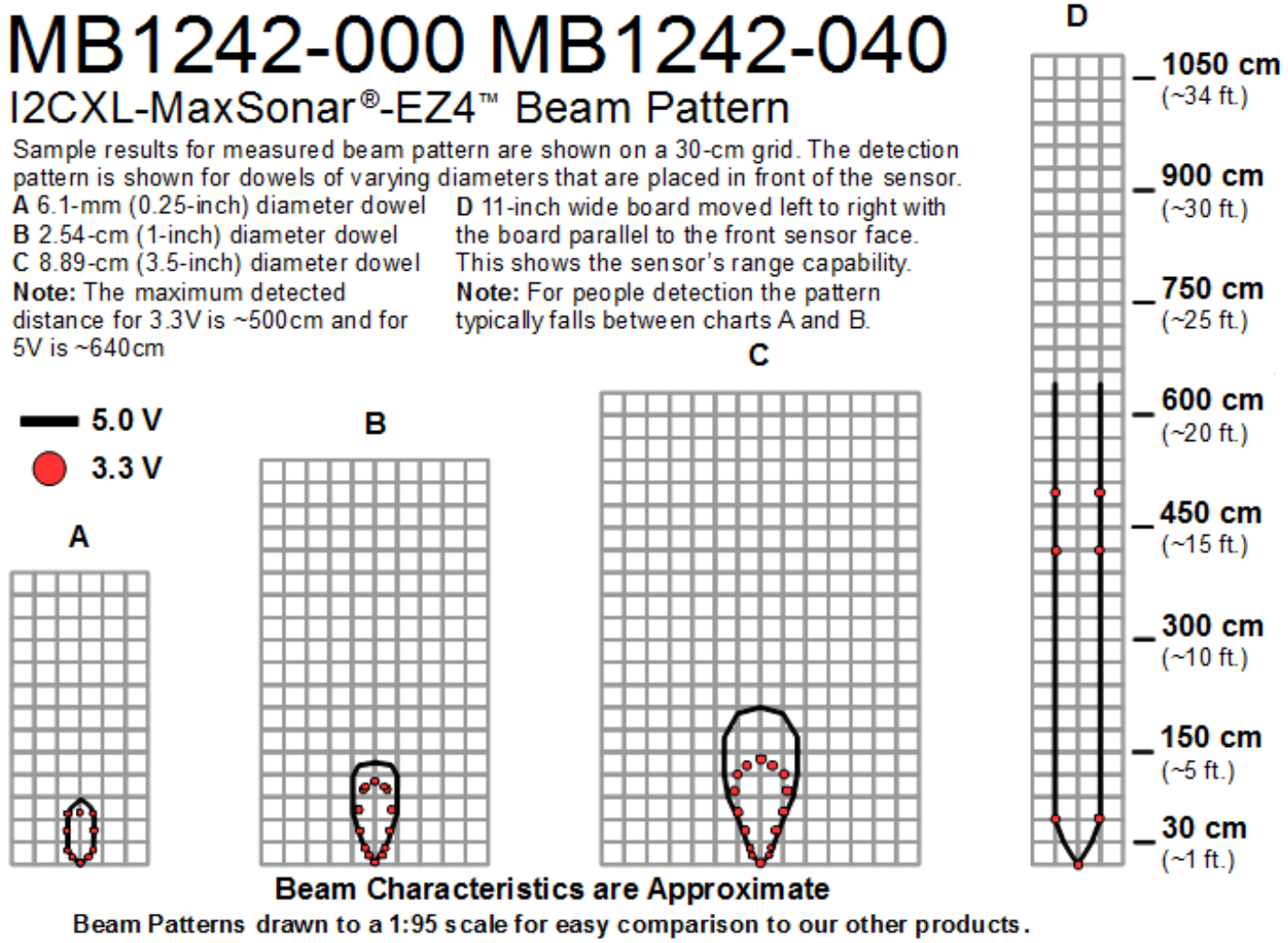
MB1232 Applications and Uses

- Great for people detection
- Security
- Motion detection
- Landing flying objects
- Useable with battery power
- Autonomous navigation
- Educational and hobby robotics
- Collision avoidance

Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

MB1242: I2CXL-MaxSonar-EZ4

The I2CXL-MaxSonar-EZ4 has the highest noise tolerance and the narrowest beam of any of our indoor sensors. The sensor is calibrated and tested to provide stable range readings to large targets even in electrically and acoustically noisy environments.



MB1242 Features and Benefits

- Narrowest beam sensor in I2CXL-MaxSonar-EZ line
- Low power consumption
- Easy to use interface
- Rejects small clutter
- Can be powered by many different types of power sources

MB1242 Applications and Uses

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

Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

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()
// 8. You do not have to keep track of the error codes passed out by the installed functions if you do not want to.
//      I inluded the error tracking so that it was easy for others to build a reliable system -- and to ease
//      troubleshooting. (not using it makes for cleaner code if you trust your interface)
```


Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

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

}
```


Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

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;
  }
}
```


Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

I2C Code Examples

Arduino Uno (as of Arduino 1.0.6)

```
////////////////////////////////////////
// Function: Change the sensor address //
////////////////////////////////////////
//Uses the I2C library to change the address of a sensor at a given address
//Collects and reports an error bit where: 1 = there was an error or 0 = there was no error.
//INPUTS: byte oldaddress = the current address of the sensor that we want to change
//INPUTS: byte newddress  = the address that we want to change the sensor to
//OUPUTS: bit  errorlevel = reports if the function was successful in changing the address: 1 = the function had an
//      error, 0 = the function was successful
boolean change_address(byte oldaddress,byte newaddress){
  //note that the new address will only work as an even number (odd numbers will round down)
  boolean errorlevel = 0;
  oldaddress = oldaddress & B11111110;  //Do a bitwise 'and' operation to force the last bit to be zero -- we are
                                        //writing to the address.
  errorlevel = !i2c_start(oldaddress) | errorlevel; //Start communication at the new address and track error codes
  errorlevel = !i2c_write(170) | errorlevel;        //Send the unlock code and track the error codes
  errorlevel = !i2c_write(165) | errorlevel;        //Send the unlock code and track the error codes
  errorlevel = !i2c_write(newaddress) | errorlevel; //Send the new address
  i2c_stop();
  return errorlevel;
}
```

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

  //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 as you
                                //will get old data.
    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);
  }
}
```


Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

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");
```


Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

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
}
```


Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

Part Numbers

All part numbers are a combination of a six-character base followed by a dash and a three-digit product code. Please review the following table for more information on the three-digit product code.

M	B	1	2	X	2	-	X	X	X
Base Number							Housing	Options	Connection

0	Not Applicable
1	3/4” NPS WR
2	3/4” NPS WRC
3	Ultra Compact
4	Ultra Compact Flush Mount
5	1” NPS
6	1” BSPP
7	30MM 1.5
8	Extended Horn
A	MB7974 Snap On Horn
B	90 Degree Horn
C	Grain Bin Horn

0	No Options (Bagged)
1	F-Option
2	P-Option
3	F-Option and P-Option
4	No Options (Trayed)
5	TTL (Bagged)
6	TTL (Trayed)

0	No Wire
1	Wire Attached
2	90 Degree 7 Pin Header
3	Straight 7 Pin Header
4	7 Pin Molex Connector
5	5m Cable 8 Pin M12 Connector
6	7 Pin Screw Block Terminal
7	MB7985 Radio Bridge Connector 5ft
8	Custom Connector or Cable

The following table displays all of the active and valid part numbers for this product.

Active Part Numbers for MB1202, MB1212, MB1222, MB1232 and MB1242				
MB1202-000	MB1212-000	MB1222-000	MB1232-000	MB1242-000
MB1202-040	MB1212-040	MB1222-040	MB1232-040	MB1242-040

Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.

After reviewing this datasheet, do you have any more questions?

We offer Technical Support on all of our products even if you purchased them through one of our many vendors worldwide.

You can **fill out a Technical Support form** for assistance on a sensor here --> **Technical Support**

Not sure which sensor you need for your application?

We offer Sensor Selection Assistance, click the link here to fill out a form for support --> **Sensor Selection Help**

Looking for tutorials to help you get started?

Frequently Asked Questions about Our Sensors

We receive many questions about our products and services. This resource offers answers to common inquiries we receive about our product lines and their application.

Fully Calibrated Beam Patterns

All of our sensors are factory calibrated to provide consistent beam patterns, detection zones, to fit into a wide variety of applications. In our product lines, each model number comes with a different beam pattern that reflects the sensitivity and the detection zone of how it sees a target. Additionally, we strive to maintain consistency between our finished products, and you will see little to no deviation between sensors of the same model. This allows you to have confidence in your final application when using multiple sensors.

Understanding Range Readings

The success of an application may hinge upon knowing the exact location of a target. However, a sensor may report one meter even if the target is not exactly one meter away from the sensor. Sensor specifications, such as resolution, precision, and accuracy, help you to understand sensor performance.

How to Use Multiple Ultrasonic Sensors

This guide covers three ways to run your sensors in a Multiple Sensor environment and issues you may face.

Contact us now with any questions at sensors@maxbotix.com or call +1-218-454-0766.

Please call during our preferred business hours of 8:00 am – 4:30 pm CST on Monday through Thursday and 8:00 am – 2:00 pm CST on Friday, or you may leave us a voicemail anytime.

Copyright 2005 - 2023 MaxBotix Incorporated Patent 7,679,996
Downloaded from Arrow.com.