

Vex Ultrasonic Sensor (US) Interfacing and Programming



Ultrasonic Sensors

- Generally have a transmitter and receiver.
- Transmits a high-frequency sound.
- Waits to receive echo.
- Calculates distance based on time it took to receive the echo.

$$d = (v)(t)$$

Round-Trip
Distance

$$d = (v)(t)/2$$

One-Way Distance

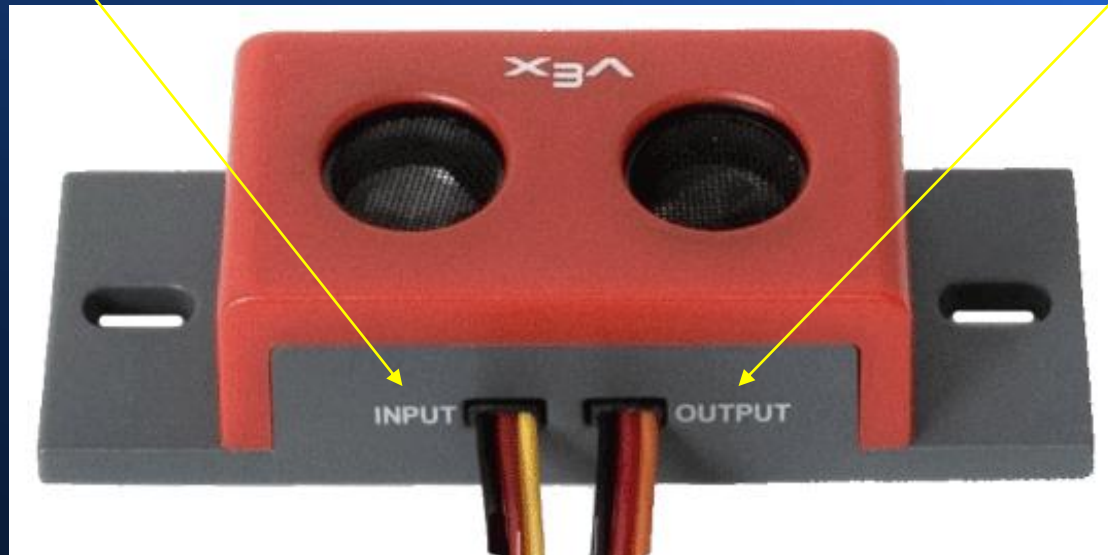
The Vex Ultrasonic Sensor

INPUT

- Sends ultrasonic wave
- Connect to Digital OUTPUT*

OUTPUT

- Receives the echo
- Connect to INTERRUPT**



The labeling is a bit counter-intuitive!

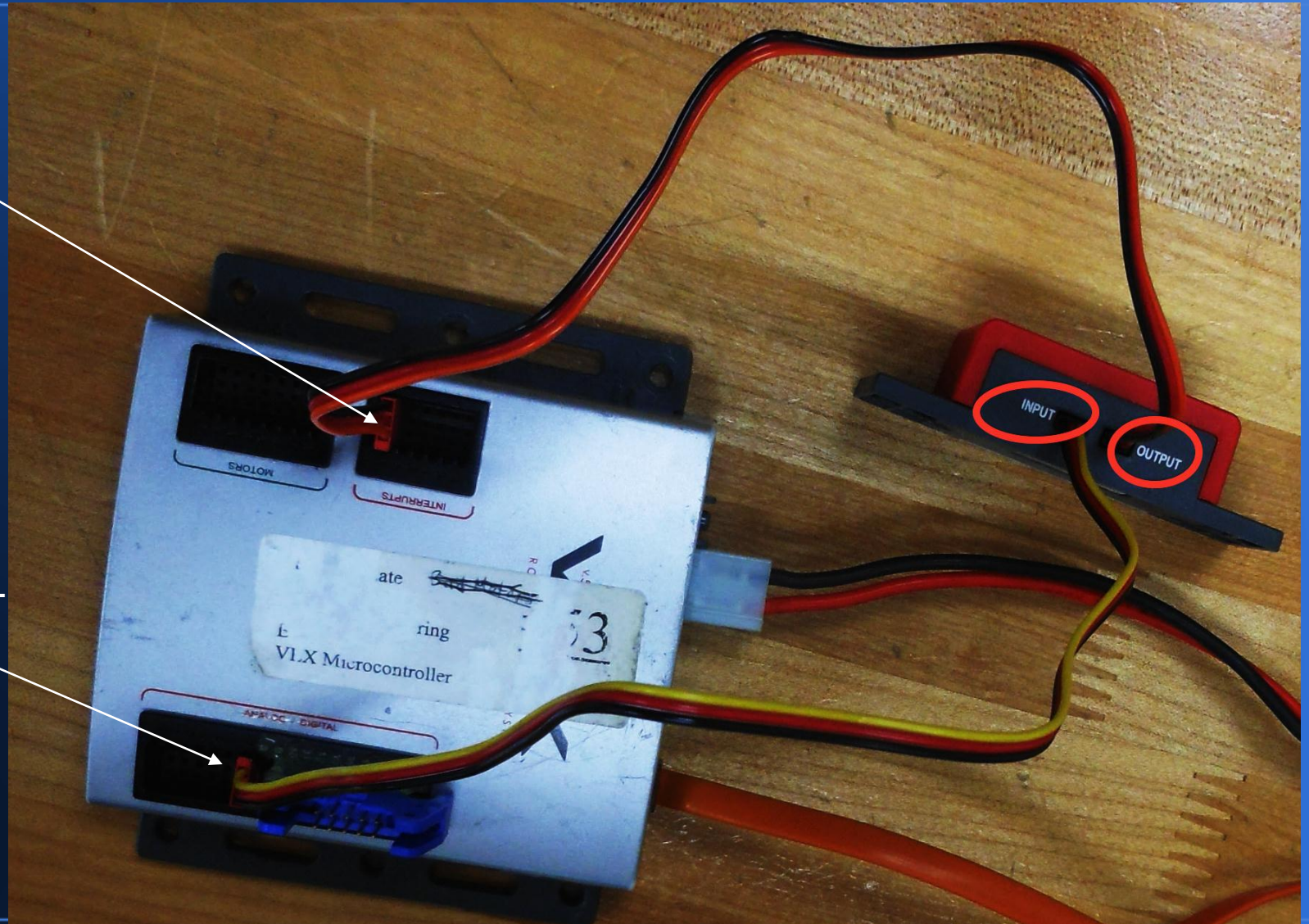
*User may have to turn a Digital INPUT into Digital OUTPUT in easyC (See Page 5).

**Any INTERRUPT port may be chosen.

Interfacing Example

INTERRUPT
Port #1

ANALOG/DIGITAL
Port #7



Check the Vex Controller Configuration

The screenshot shows the VEX IDE interface. On the left is a project tree with 'Config', 'Globals', 'BEGIN', 'Variables', and 'END' folders. The main workspace contains C++ code for a 'void main' function. A 'StartUltrasonic' block is highlighted in green, with its code: `StartUltrasonic (1 , 7); // OUTPUT wire to INT #1, INPUT wire to ANALOG/DIGITAL #7`. The 'Controller Configuration' window is open, displaying a VEX v.5 Robotics Design System schematic. It features three tables: 'INTERRUPTS', 'MOTORS', and 'ANALOG / DIGITAL'. The 'INTERRUPTS' table has a row for port #1 with the description 'US sensor OUTPUT wire'. The 'ANALOG / DIGITAL' table has a row for port #7 with the description 'US ser'. To the right of these tables is a physical pin header diagram with 16 pins. Pin 1 is labeled 'Analog signal' and has a wavy icon. Pin 7 has an arrow icon pointing out of a circle. A legend at the bottom right explains: 'Left-Click to set Digital I/O', 'Right-Click to set Analog Input'. A text box at the bottom right contains the number '5'.

```
1 #include "Main.h"
2
3 void main ( void )
4 {
5     int distance;
6
7     StartUltrasonic ( 1 , 7 ); // OUTPUT wire to INT #1, INPUT wire to ANALOG/DIGITAL #7
8 }
```

Press F5 to bring up this window.

OUTPUT wire goes to one of the INTERRUPT ports, e.g. Port #1 here.

INPUT wire goes to ANALOG/DIGITAL, e.g. Port #7 here; must configure so that the icon is an arrow going OUT of circle (click the icon until it changes).

Analog signal

5

Programming The Ultrasonic Sensor

- 4 Steps must be taken to use the sensor:
 - 1) Declare a variable that will store the data received by the sensor (number of inches.)
 - 2) Call the StartUltrasonic(x,y) function (once only.)
 - $x \rightarrow$ INTERRUPT port #
 - $y \rightarrow$ Digital OUTPUT port #
 - 3) Include variable = GetUltrasonic(x,y) in a loop in order to poll for values (similar to using a bumper.)
 - Stores the number of inches in “variable”
 - 4) Use the variable to do something!

Step 1: Declare a Variable

The integer variable called "distance" will be used to store how far away the ultrasonic sensor is from an obstacle in inches.

```
void main ( void )  
{  
5  int distance;  
6  
7 }
```

#	Type	Name	Value	Comment
1	int	distance		
>>				

The screenshot shows an IDE window titled "Main" with a project tree on the left containing "Config", "Globals", "BEGIN", "Variables", and "END". The "Variables" folder is selected. The main editor displays C code with line numbers 1-7. Line 5, "int distance;", is highlighted in cyan. A "Local Variables" dialog box is open, showing a table with one entry: #1, Type int, Name distance. Arrows point from the text and the dialog to the code line.

Step 2: Call the StartUltrasonic() Function

```
void main [ void ]  
{  
  StartUltrasonic ( 1 , 7 ); // OUTPUT wire to INT #1, INPUT wire to ANALOG/DIGITAL #7  
}
```

Drag "Ultrasonic" block from menu on left to get this window. This window is used to call StartUltrasonic(), GetUltrasonic(), etc.

OUTPUT wire on INTERRUPT port #1

INPUT wire on Digital OUTPUT port #7

```
1 #include "Main.h"  
2  
3 void main ( void )  
4 {  
5   int distance;  
6  
7   StartUltrasonic ( 1 , 7 ); // OUTPUT wire to INT #1, INPUT wire to ANALOG/DIGITAL #7  
8 }
```

Ultrasonic Sensor

Select command:

Start
 Get
 Stop

Interrupt Port #: 1 (Value Range: 1..6)

Output Port #: 7 (Value Range: 1..16)

Code:
StartUltrasonic (1 , 7);

Comment:
OUTPUT wire to INT #1, INPUT wire to ANALOG/DIGITAL #7

F6 - Globals and Constants Ctrl + F6 - Local Variables

OK Cancel Help

(Choosing "Stop" disables Ultrasonic capability.)

Step 3: Polling for Data

The screenshot displays an IDE interface with three main components:

- Flowchart (Left):** A vertical sequence of blocks: 'Config', 'Globals', 'BEGIN', 'Variables', a 'StartUltrasonic (1 , 7); // OUTPUT wire to INT #1, INPUT wire to ANALOG/DIGITAL #7' block, a 'WHILE' loop block containing a '{' block, a '}' block, and an 'END' block.
- Source Code (Right):** A C++ code editor showing:

```
1 #include "Main.h"
2
3 void main ( void )
4 {
5     int distance;
6
7     StartUltrasonic ( 1 , 7 ); // OUTPUT wire to INT #1, INPUT wire to ANALOG/DIGITAL #7
8     while ( 1 ) // Infinite loop for continuous polling
9     {
10    }
11 }
```
- While Loop Dialog (Bottom Right):** A configuration window titled 'While Loop' with the following fields:
 - Expression: while (1)
 - Code: while (1)
 - Comment: Infinite loop for continuous polling
 - Buttons: OK, Cancel, Help

The text below the flowchart explains the purpose of the while loop:

The GetUltrasonic() function will need to be inside some sort of loop in order to continuously poll for data, e.g. this infinite loop (while Loop).

Step 3: Polling for Data

The image shows a programming IDE with three main components:

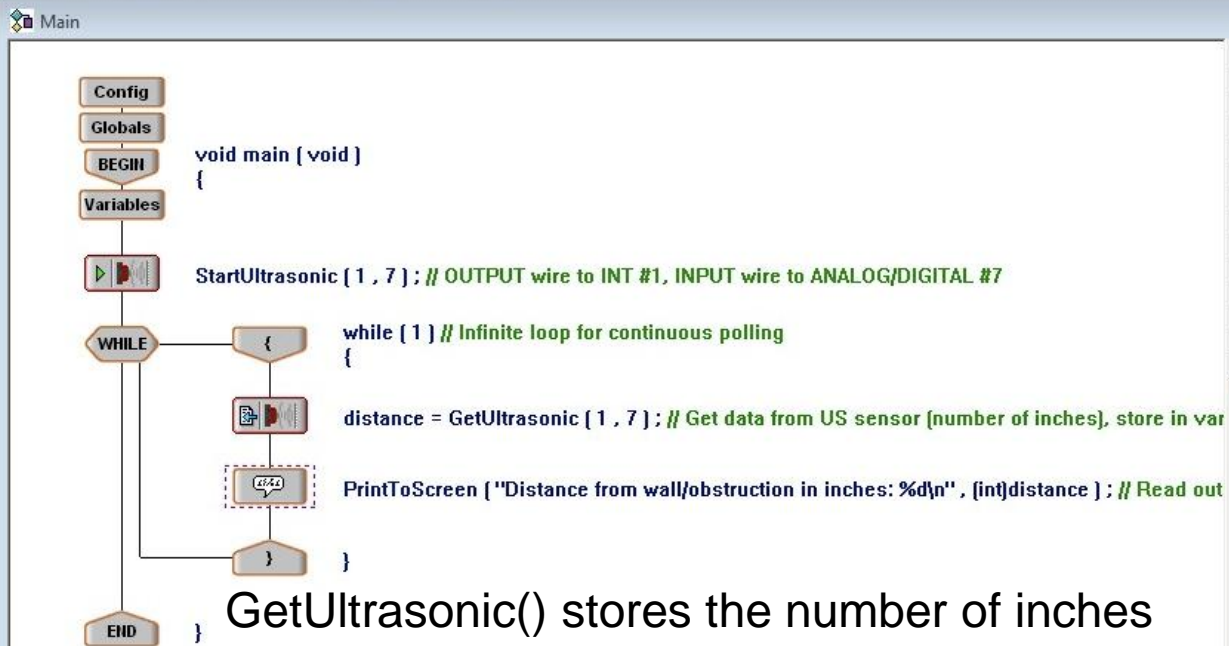
- Flowchart (Left):** A vertical flowchart starting with 'Config', 'Globals', 'BEGIN', and 'Variables'. Below these is a 'StartUltrasonic' block with parameters [1, 7]. This is followed by a 'WHILE' loop block containing a 'GetUltrasonic' block with parameters [1, 7]. The flowchart ends with 'END'.
- Code Editor (Top Right):** Shows C++ code for an ultrasonic sensor. The code includes a header file, a main function, and a while loop for polling. The line `distance = GetUltrasonic (1 , 7);` is highlighted in blue.
- Configuration Dialog (Bottom Right):** Titled 'Ultrasonic Sensor', it has three radio buttons: 'Start', 'Get' (selected and circled in red), and 'Stop'. It has three dropdown menus: 'Interrupt Port #' (set to 1), 'Output Port #' (set to 7), and 'Retrieve to:' (set to 'distance'). Below these are fields for 'Code:' (containing `distance = GetUltrasonic (1 , 7);`) and 'Comment:' (containing 'Get data from US sensor (number of inches), store in variable "distance"').

After StartUltrasonic() is called, the GetUltrasonic() function can be used in the same manner as GetDigitalInput() for bumpers.

Specify the Interrupt Port and Digital Port numbers (for the OUTPUT an INPUT wires, respectively.)

Retrieve the data to the "distance" variable.

Step 4: Use the Data



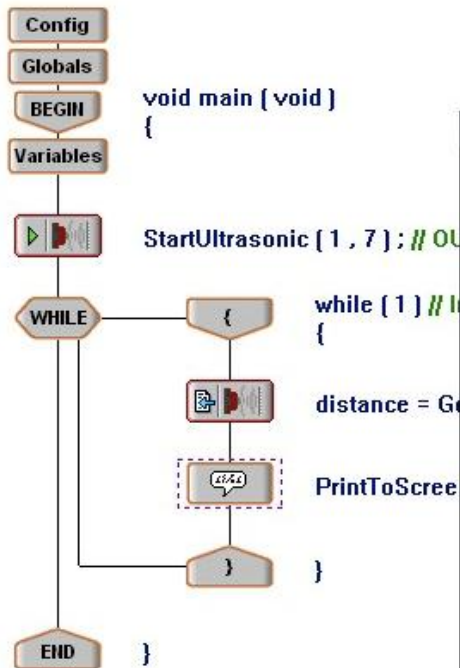
GetUltrasonic() stores the number of inches the sensor is from the obstacle to the variable "distance." The sensor can detect distances between ~3in. to ~115in.

After GetUltrasonic() stores a value in "distance," the variable may be used for any purpose. Here, a simple message is printed stating the distance to an object on each iteration of the loop.

```
1 #include "Main.h"
2
3 void main ( void )
4 {
5     int distance;
6
7     StartUltrasonic ( 1 , 7 ); // OUTPUT wire to INT #1, INPUT wire to ANALOG/DIGITAL #7
8     while ( 1 ) // Infinite loop for continuous polling
9     {
10        distance = GetUltrasonic ( 1 , 7 ); // Get data from US sensor (number of inches), store in
11        PrintToScreen ( "Distance from wall/obstruction in inches: %d\n" , (int)distance ); // Read
12    }
13 }
```

The 'Print To Screen' dialog box is open. It has a 'Message:' field containing 'Distance from wall/obstruction in inches:'. Below this are three dropdown menus: 'Variable:' set to 'distance', 'Directive:' set to '%d', and 'Type-cast:' set to 'int'. There is a checked checkbox for 'Newline Character'. The 'Code:' field contains 'PrintToScreen ("Distance from wall/obstruction in inches: %d\n" , (int)distance);'. The 'Comment:' field contains 'Read out this message in the Terminal for reference to yourself'. At the bottom, there are 'OK', 'Cancel', and 'Help' buttons, along with keyboard shortcuts: 'F6 - Globals and Constants' and 'Ctrl + F6 - Local Variables'.

Step 4: Use the Data (Example)



The screenshot shows the 'IFI/intelitek Loader' window. The 'Terminal' tab is active, displaying the output of the program. The output consists of 25 lines of the text 'Distance from wall/obstruction in inches: 25'. The window also shows the 'Port Setting' and 'Options' menus, and a status bar at the bottom indicating 'COM port is open...' and the hardware configuration 'VEX PIC18F8520 COM4 115200'. In the background, a code editor shows the following code:

```
2
3 void main ( void )
4 {
5     int distance;
6     StartUltrasonic ( 1 , 7 ); // OUTPUT
7     // Infinite loop for continuous
8     while ( 1 ) // Infinite loop
9     {
10        distance = GetUltrasonic ( 1 , 7 );
11        PrintToScreen ( "Distance from wall/obstruction in inches: " + distance );
12    }
```

The sensor detects that the wall is 25 inches away.