Application Note

PGA460 Control Based on MSPM0 for Distance Detection

Jingguo Wang and Eason Zhou

ABSTRACT

This application note describes a distance detection solution based on the PGA460 and MSPM0C1104. This solution includes related evaluation steps and sample code.

Project collateral discussed in this application note can be downloaded from the following URL: https://www.ti.com/lit/zip/slaaem0.

Table of Contents

List of Figures

List of Tables

Trademarks

1 Introduction

The PGA460 device is a highly-integrated system on-chip ultrasonic transducer driver and signal conditioner with an advanced DSP core. MSPM0C110x are part of the MSP highly-integrated ultra-low-power 32-bit microcontroller unit (MCU) family based on the enhanced Arm® Cortex®-M0+ core platform operating at up to 24-MHz frequency. The MSPM0C110x can serve as the main controller for the PGA460, providing it with configuration parameters and operating modes based on serial peripheral interface (SPI) communication, and transmitting the measurement result of PGA460 to PC.

This application report provides the evaluation steps and sample code (https://www.ti.com/lit/zip/slaaem0) for PGA460 base on the MSPM0C1104. You can also find it under SDK with this address: `C:\ti\mspm0_sdk_x_xx_xx_xx\examples\nortos\LP_MSPM0C1104\demos\pga460_control_spi`.

2 Hardware Introduction

To evaluate the solution based on PGA460 and MSPM01104, the following hardware elements are required:

- BOOSTXL-PGA460 (PGA460-Q1 ultrasonic sensor signal conditioning evaluation module with transducers)
- LP-MSPM0C1104 (MSPMC1104 LaunchPad™ development kit for 24-MHz Arm® Cortex®-M0+ MCU)
- A computer with Windows® 7 or later, and .NET Framework 4.5
- Micro-USB to USB cable (included with the purchase of LP-MSPM0C1104)

In the solution, MSPM0C1104 acts as the main controller to configure PGA460 via SPI, and sends the measurement results to PC via universal/asynchronous receiver/transmitter (UART). Figure 2-1 shows the simply system block diagram.

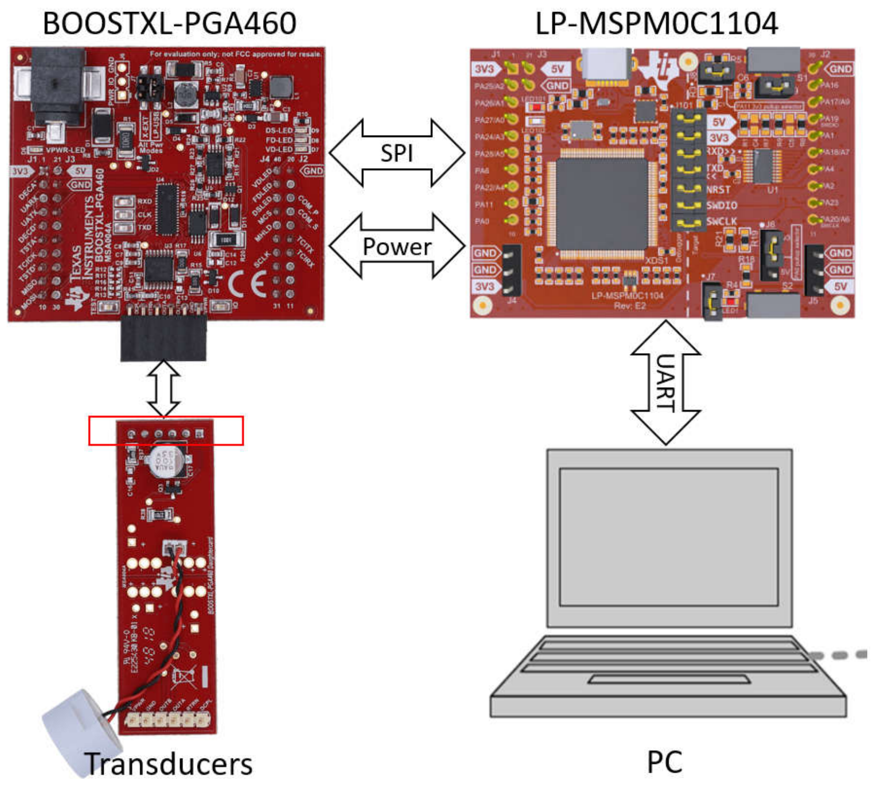


Figure 2-1. System Block Diagram

PGA460-Q1 support multiple communication interfaces, including USART, TCI, SPI and One-Wire UART. USART Synchronous Mode is identical to a SPI without a chip select because the addressing is handled by the three-bit UART_ADDR value to enable up to eight device son a single bus. In this solution, the three-wire SPI communication interface was used on MSPM0C1104 to control PGA460. To enable SPI communication module in PGA460, pull down P18(COM_P) and pull up P17(COM_S), P37(MHLD), P36(MCS) on PGA460. This is realized by controlling PA2, PA17 and PA23 on MSPM0C1104. The power supply (5V, 3V3, GND) and SPI (MOSI, MISO, Clock) can be connected correspondingly. All the hardware connection between PGA460 EVM and LP-MSPM0C1104 is shown in Table 2-1.

Table 2-1. Hardware Connection

Connection Type	Connection Name	LP-MSPM0C1104 Pin Number: Pin Name	BOOSTXL-PGA460 Pin Number: Pin Name
SPI	SPI: MOSI	PA18: SPI_PICO	P10: MOSI
	SPI: MISO	PA4: SPI_POCI	P9: MISO
	SPI: Clock	PA11	P34: SCLK_LP
SPI enable	COM_PD	GND	P18: COM_P
	COM_SEL	PA2	P17: COM_S
	MEM_HOLD	PA17	P36: MHLD
	MEM_CS	PA23	P37: MCS
Power connections	Power: 3.3V	J1:3V3	P1: 3V3
	Power: 5V	J3:5V	P21: 5V
	Power: Ground	GND	P22: GND

The final setup is shown in Figure 2-2.

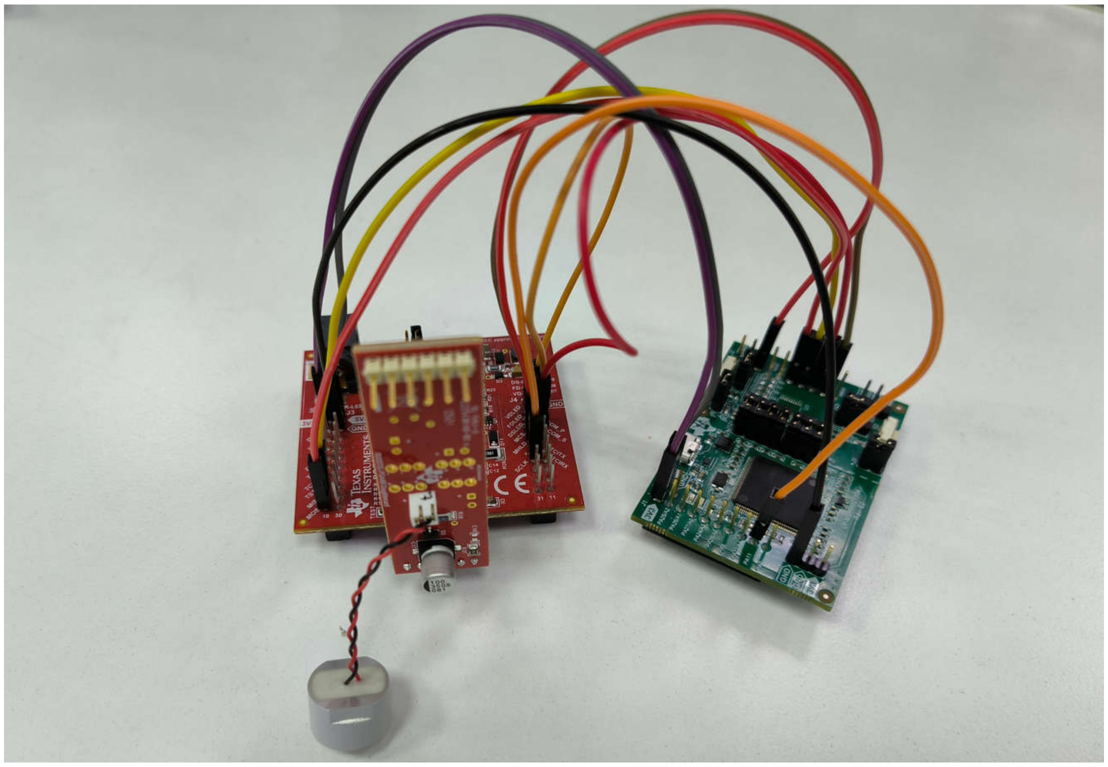
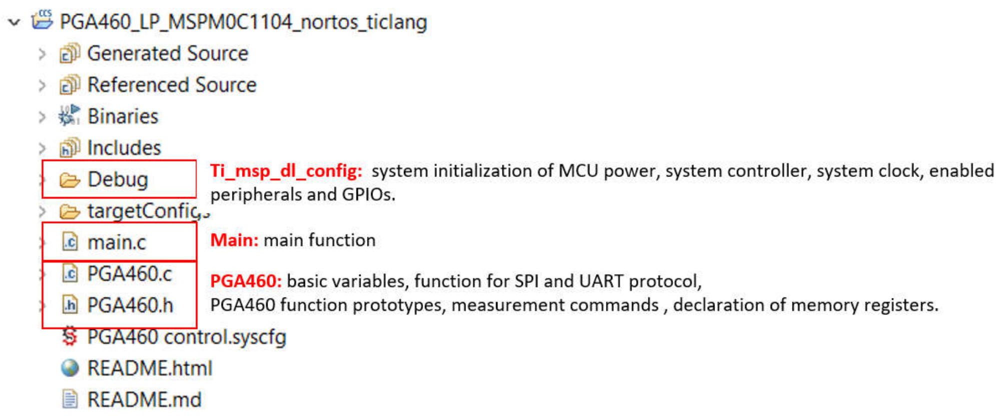


Figure 2-2. Hardware Setup

3 Software Introduction

The software project is shown in Figure 3-1, developed in Code Composer Studio™ (CCS) software. It mainly consists of three parts. For other files, they are the default files for the MSPM0 project.



```
PGA460_LP_MSPM0C1104_nortos_ticlang
  Generated Source
  Referenced Source
  Binaries
  Includes
  Debug              Ti_msp_dl_config: system initialization of MCU power, system controller, system clock, enabled
                     peripherals and GPIOs.
  targetConfigs
  main.c             Main: main function
  PGA460.c           PGA460: basic variables, function for SPI and UART protocol,
  PGA460.h           PGA460 function prototypes, measurement commands , declaration of memory registers.
  PGA460 control.syscfg
  README.html
  README.md
```

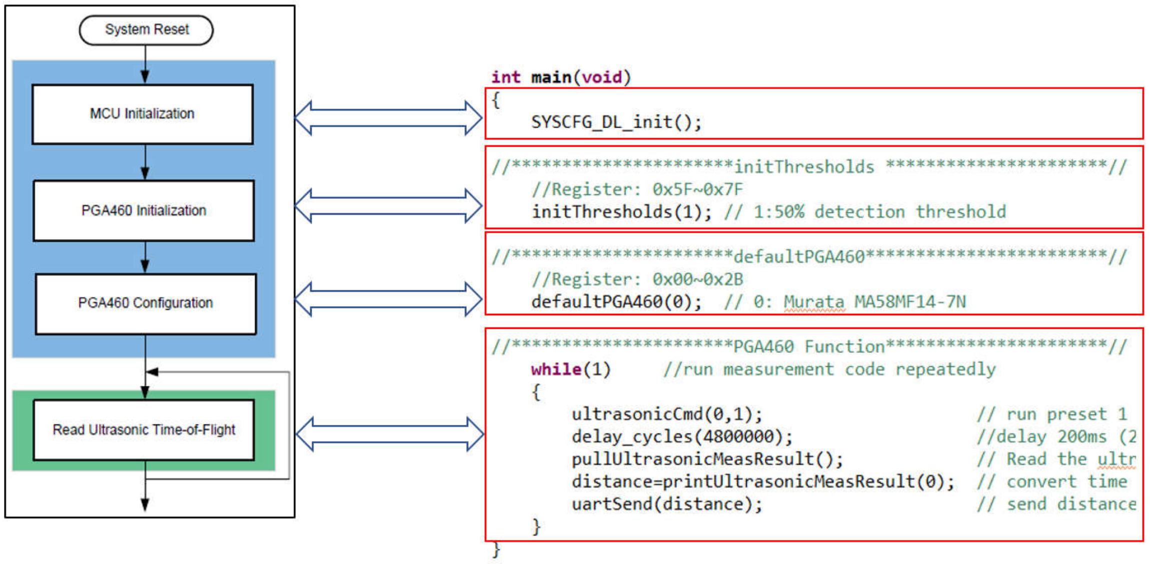
Figure 3-1. Software Project View

For ti_msp_dl_config part, it is generated by sysconfig (Graphic code generation tool), the MSPM0 initialization is for system initialization of MCU power, system controller, system clock, enabled peripherals and general-purpose input/output (GPIO).

For PGA460 part, it declares all the memory registers, basic variables, direct commands in *PGA460 Ultrasonic Signal Processor and Transducer Driver Data Sheet*. It also contains some basic function for SPI and UART protocol.

For the main part, it includes the highest system function code, after system initialization, the system continually sends operational commands to the PGA460 and transmits the measurement results back to the PC.

The PGA460 device can only be operated as a slave device and must be paired with an external microcontroller unit (MCU) which acts as the master device. The master device is responsible for the initialization, configuration, and regular polling operation of the PGA460 device. Figure 3-2 shows the high-level overview of the software flow and corresponding code for standard PGA460 operation.



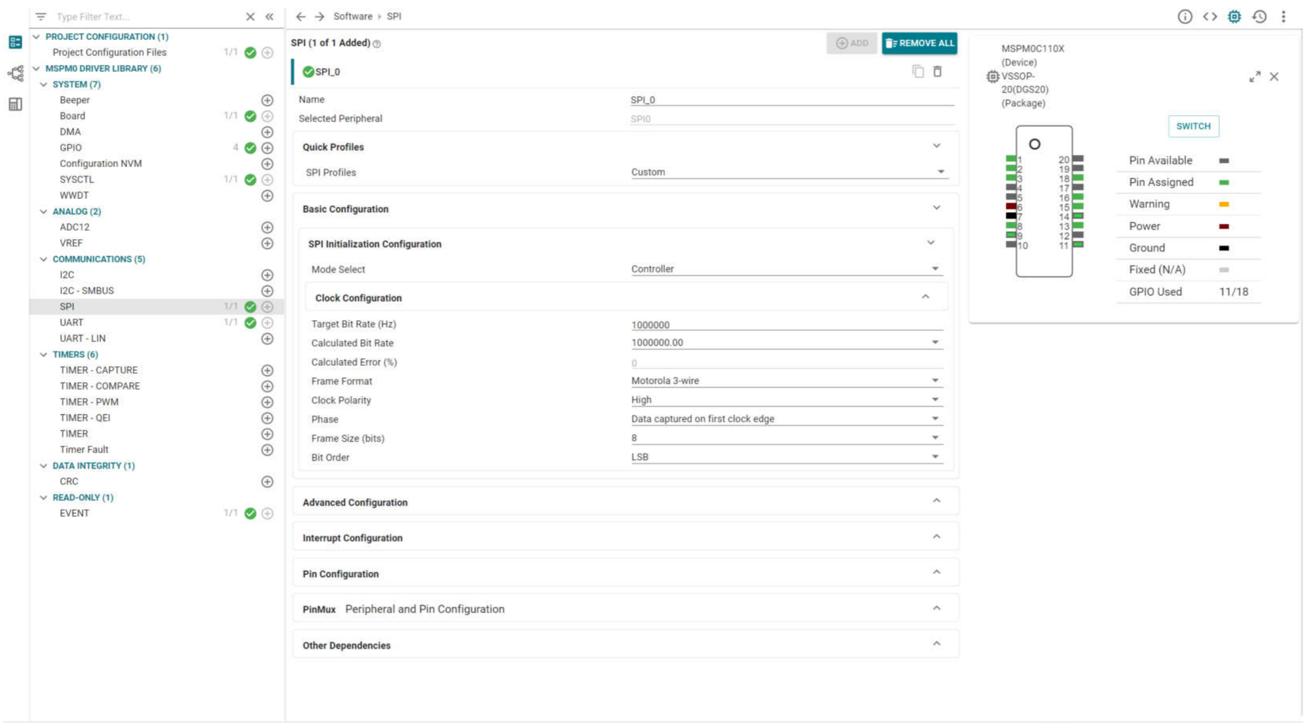
```
int main(void)
{
    SYSCFG_DL_init();
//**********************initThresholds **********************//
    //Register: 0x5F~0x7F
    initThresholds(1); // 1:50% detection threshold
//**********************defaultPGA460**********************//
    //Register: 0x00~0x2B
    defaultPGA460(0);  // 0: Murata MA58MF14-7N
//**********************PGA460 Function**********************//
    while(1)     //run measurement code repeatedly
    {
        ultrasonicCmd(0,1);                      // run preset 1
        delay_cycles(4800000);                   //delay 200ms (2
        pullUltrasonicMeasResult();              // Read the ultr
        distance=printUltrasonicMeasResult(0);   // convert time
        uartSend(distance);                      // send distance
    }
}
```

Figure 3-2. High-level Software Flowchart

3.1 MCU Initialization

The first step of the high-level software flowchart is MCU initialization. For this step, use sysconfig, which can set peripherals parameters and initialize the MCU conveniently and efficiently.

Using the SPI configuration as an example, the SPI terminals of the master controller must be adapted to the PGA460-compatible format and baud rate. Select controller mode in this solution, and specify the master controller clock is referenced to a source and frequency that can support the designated baud rate. In this solution, set baud rate as 1MHz, and customers can make an adjustment within reasonable limits. As for frame format, the MCU configuration must meet the requirements of the PGA460, which use Motorola 3 wire frame format. The clock polarity is high (SPO = 1), and data captured on the first clock edge transition (SPH = 0). Frame size is 8 bits, and bit order is LSB first. These parameters can be easily configured in the sysconfig as shown in Figure 3-3.



SPI (1 of 1 Added)

SPI_0

Name	SPI_0
Selected Peripheral	SPI0
Quick Profiles	
SPI Profiles	Custom
Basic Configuration	
SPI Initialization Configuration	
Mode Select	Controller
Clock Configuration	
Target Bit Rate (Hz)	1000000
Calculated Bit Rate	1000000.00
Calculated Error (%)	0
Frame Format	Motorola 3-wire
Clock Polarity	High
Phase	Data captured on first clock edge
Frame Size (bits)	8
Bit Order	LSB

Advanced Configuration

Interrupt Configuration

Pin Configuration

PinMux Peripheral and Pin Configuration

Other Dependencies

Pin Available	▬
Pin Assigned	🟩
Warning	🟧
Power	🟥
Ground	⬛
Fixed (N/A)	▭
GPIO Used	11/18

Figure 3-3. SPI Configuration in Sysconfig

After configuring the parameters of the peripherals, configuration files can be generated through debugging as shown in Figure 3-4.

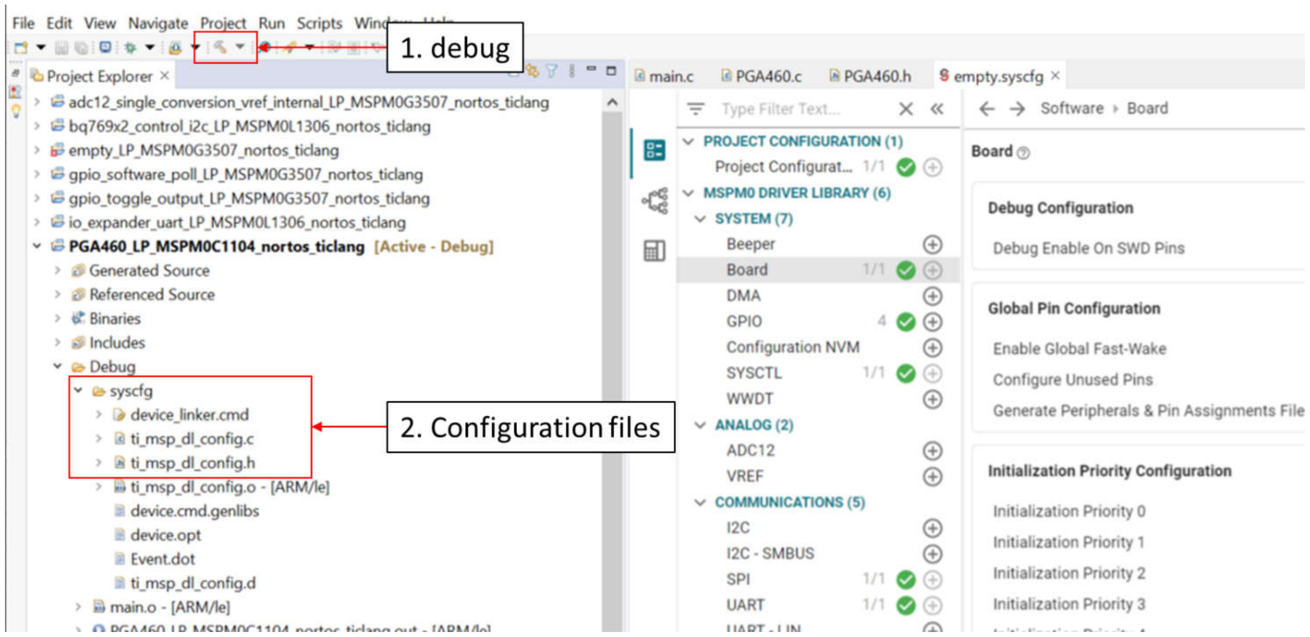

File Edit View Navigate Project Run Scripts Window Help
1. debug
Project Explorer
adc12_single_conversion_vref_internal_LP_MSPM0G3507_nortos_ticlang
bq769x2_control_i2c_LP_MSPM0L1306_nortos_ticlang
empty_LP_MSPM0G3507_nortos_ticlang
gpio_software_poll_LP_MSPM0G3507_nortos_ticlang
gpio_toggle_output_LP_MSPM0G3507_nortos_ticlang
io_expander_uart_LP_MSPM0L1306_nortos_ticlang
PGA460_LP_MSPM0C1104_nortos_ticlang [Active - Debug]
Generated Source
Referenced Source
Binaries
Includes
Debug
syscfg
device_linker.cmd
ti_msp_dl_config.c
ti_msp_dl_config.h
ti_msp_dl_config.o - [ARM/le]
device.cmd.genlibs
device.opt
Event.dot
ti_msp_dl_config.d
main.o - [ARM/le]
2. Configuration files
main.c
PGA460.c
PGA460.h
empty.syscfg
Type Filter Text...
Software › Board
PROJECT CONFIGURATION (1)
Project Configurat... 1/1
MSPM0 DRIVER LIBRARY (6)
SYSTEM (7)
Beeper
Board 1/1
DMA
GPIO 4
Configuration NVM
SYSCTL 1/1
WWDT
ANALOG (2)
ADC12
VREF
COMMUNICATIONS (5)
I2C
I2C - SMBUS
SPI 1/1
UART 1/1
Board
Debug Configuration
Debug Enable On SWD Pins
Global Pin Configuration
Enable Global Fast-Wake
Configure Unused Pins
Generate Peripherals & Pin Assignments File
Initialization Priority Configuration
Initialization Priority 0
Initialization Priority 1
Initialization Priority 2
Initialization Priority 3

Figure 3-4. Configuration Files

3.2 PGA460 Initialization and Configuration

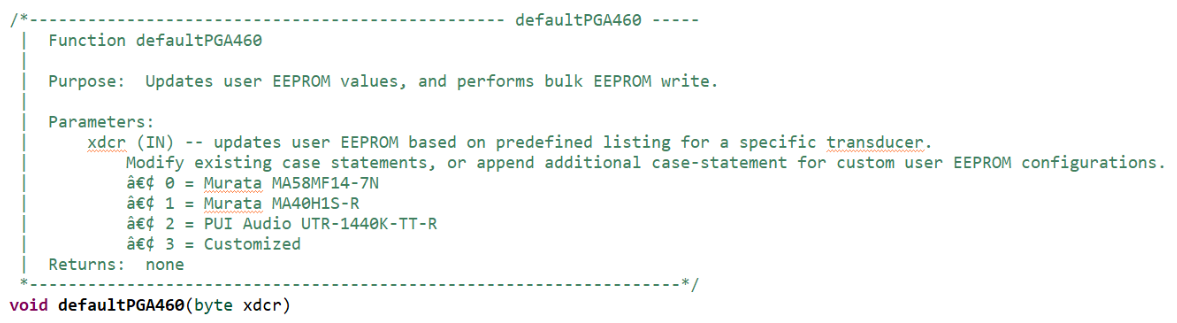
There are two functions used to Initialize and Configure PGA460, shown in Figure 3-2.

The first is initThresholds(), shown in Figure 3-5. It updates threshold mapping for both presets, and performs bulk threshold write. There are two kinds of parameters, one is updating all threshold levels to a fixed level based on specific percentage of the maximum level, which can be activated when the input parameter is from 0 to 2. The other is customized configuration, which can be activated when the input parameter is 3. Here we choose 50% as the detection threshold (Parameter = 1), which is get after the thgreshold adjustment under GUI.

```
/*------------------------------------------------- initThresholds -----
 |  Function initThresholds
 |
 |  Purpose:  Updates threshold mapping for both presets, and performs bulk threshold write
 |
 |  Parameters:
 |      thr (IN) -- updates all threshold levels to a fixed level based on specific percentage of the maximum level.
 |          All times are mid-code (1.4ms intervals).
 |          Modify existing case statements, or append additional case-statement for custom user threshold configurations.
 |          â€¢ 0 = 25% Levels 64 of 255
 |          â€¢ 1 = 50% Levels 128 of 255
 |          â€¢ 2 = 75% Levels 192 of 255
 |          â€¢ 3 = customized
 |  Returns:  none
 *-------------------------------------------------------------------*/
void initThresholds(byte thr)
```

Figure 3-5. PGA460 Detection Threshold Initialization

After the thresholds have been updated, the system can now continuously run the burst-and-listen command to pull the resulting measurement data. However, for proper system configuration, more parameters should be settled, in order that the ultrasonic module can behave as expected, such as time varying gain (TVG), burst frequency and so on. The PGA460 default configuration function is used to set suitable parameters according to the used transducer, shown in Figure 3-6. There are four recommended configurations for different transducers, and the last one is for customized configuration. Choose parameter series 0, for the Murata transducer MA58MF14-7N.

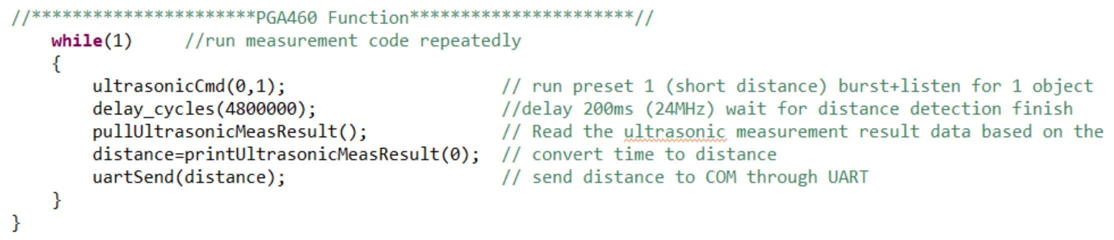


```
/*------------------------------------------------- defaultPGA460 -----
 |  Function defaultPGA460
 |
 |  Purpose:  Updates user EEPROM values, and performs bulk EEPROM write.
 |
 |  Parameters:
 |      xdcr (IN) -- updates user EEPROM based on predefined listing for a specific transducer.
 |          Modify existing case statements, or append additional case-statement for custom user EEPROM configurations.
 |          â€¢ 0 = Murata MA58MF14-7N
 |          â€¢ 1 = Murata MA40H1S-R
 |          â€¢ 2 = PUI Audio UTR-1440K-TT-R
 |          â€¢ 3 = Customized
 |  Returns:  none
 *-------------------------------------------------------------------*/
void defaultPGA460(byte xdcr)
```

Figure 3-6. PGA460 Default Configuration Function

3.3 Distance Detection

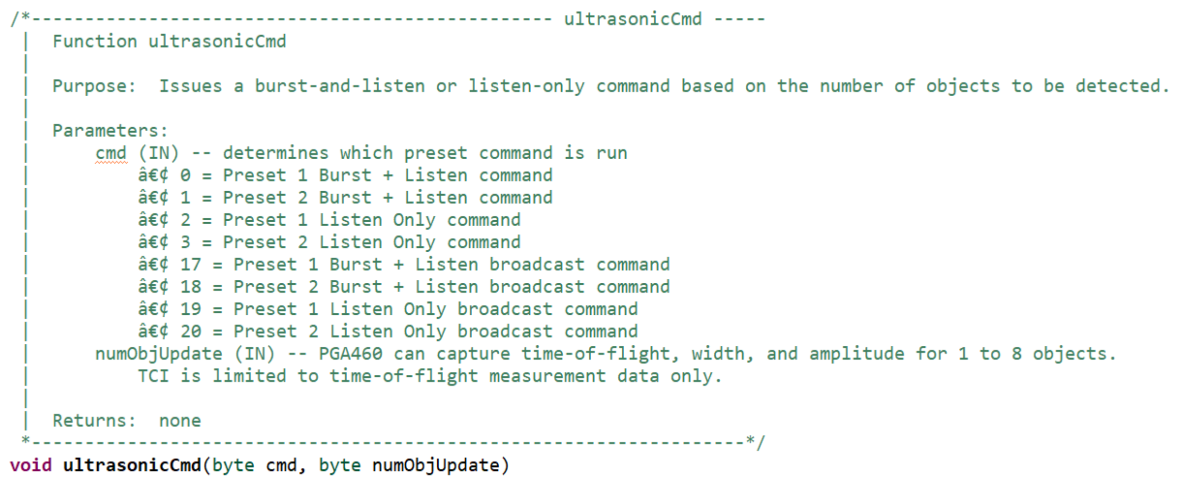
After the initialization and default configuration update of the PGA460, the system is able to continuously execute the burst-and-listen command to retrieve the resulting measurement data. Figure 3-7 illustrates the operational process of the PGA460.



```
//*********************PGA460 Function*********************//
    while(1)     //run measurement code repeatedly
    {
        ultrasonicCmd(0,1);                   // run preset 1 (short distance) burst+listen for 1 object
        delay_cycles(4800000);                //delay 200ms (24MHz) wait for distance detection finish
        pullUltrasonicMeasResult();           // Read the ultrasonic measurement result data based on the
        distance=printUltrasonicMeasResult(0); // convert time to distance
        uartSend(distance);                   // send distance to COM through UART
    }
}
```

Figure 3-7. PGA460 Run Operation

The operation includes four steps, the first step is to send running commands from MCU to PGA460, as shown in Figure 3-8. In this function, preset 1 (P1) or preset 2 (P2) can be selected, if a Burst & Listen command or Listen Only command is issued. Here, preset 1 (P1) was chosen.



```
/*------------------------------------------------- ultrasonicCmd -----
 |  Function ultrasonicCmd
 |
 |  Purpose:  Issues a burst-and-listen or listen-only command based on the number of objects to be detected.
 |
 |  Parameters:
 |      cmd (IN) -- determines which preset command is run
 |          â€¢ 0 = Preset 1 Burst + Listen command
 |          â€¢ 1 = Preset 2 Burst + Listen command
 |          â€¢ 2 = Preset 1 Listen Only command
 |          â€¢ 3 = Preset 2 Listen Only command
 |          â€¢ 17 = Preset 1 Burst + Listen broadcast command
 |          â€¢ 18 = Preset 2 Burst + Listen broadcast command
 |          â€¢ 19 = Preset 1 Listen Only broadcast command
 |          â€¢ 20 = Preset 2 Listen Only broadcast command
 |      numObjUpdate (IN) -- PGA460 can capture time-of-flight, width, and amplitude for 1 to 8 objects.
 |          TCI is limited to time-of-flight measurement data only.
 |
 |  Returns:  none
 *-------------------------------------------------------------------*/
void ultrasonicCmd(byte cmd, byte numObjUpdate)
```

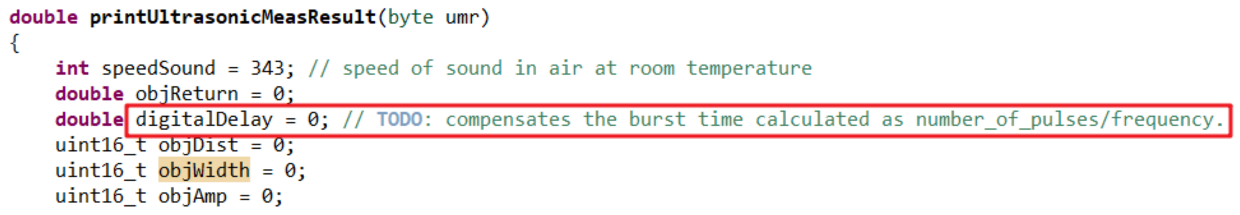
Figure 3-8. UltrasonicCmd Function

The second step is to wait for some time until the distance detection is finished. The decision was to wait for 200ms. Here, one cycle points to the CPU cycle, which runs with 24MHz.

The third step is to convert time to distance. The PGA460 device captures the interrupt time and outputs the distance equivalent, width, and peak amplitude for the returning echoes when the threshold is intersected. In this solution, only distance measurement result is needed. To solve for the time-of-flight, use velocity = distance/time. Because the speed of sound is typically at a value of 343m/s at room temperature, and the PGA460 device outputs the round-trip time at which the threshold is intersected in 1-µs resolution after bursting. The distance to the object is computed as the product of velocity and one-way time. Use Equation 1 as the PGA460-specific equation to solve for distance in meters.

$$distance(m) = \left[\frac{343m/s}{2} * (objMSB[1] \ll 8 + objLSB[2]) * 0.000001\right] + \left[\frac{343m/s}{2} * Pulses * \frac{1}{Frequency}\right] \quad (1)$$

As this demo only shows the basic function of PGA460, be sure to keep the digitalDelay to be 0, as shown in Figure 3-9. For more accurate distance detection, you should change its value according to your real setting.



```
double printUltrasonicMeasResult(byte umr)
{
    int speedSound = 343; // speed of sound in air at room temperature
    double objReturn = 0;
    double digitalDelay = 0; // TODO: compensates the burst time calculated as number_of_pulses/frequency.
    uint16_t objDist = 0;
    uint16_t objWidth = 0;
    uint16_t objAmp = 0;
```

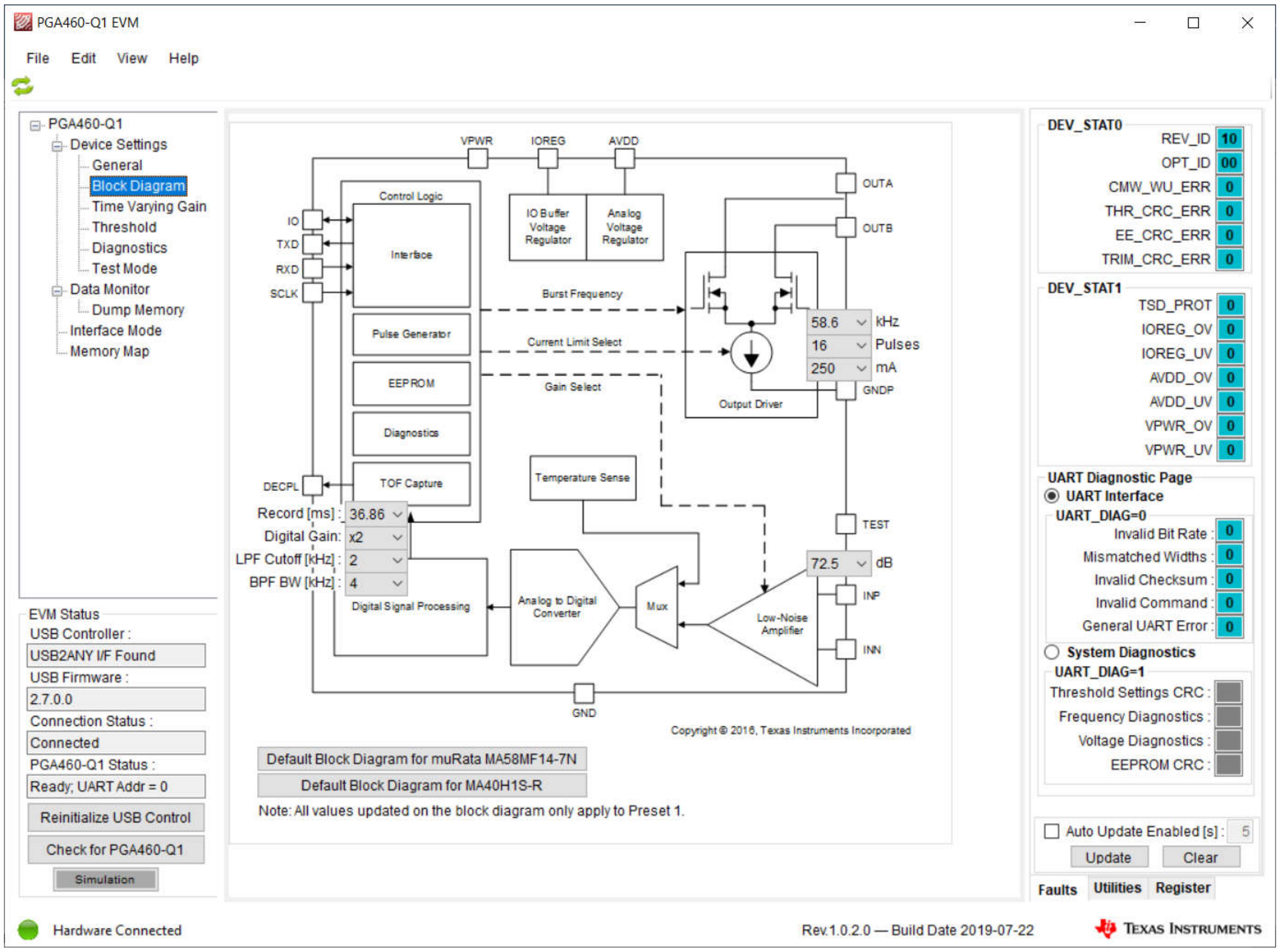
Figure 3-9. Digital Delay Configuration

The final step is converting a measurement result of double type to a string type and send it to PC from UART in uartSend function with 9600 baud rate.

4 Evaluation Steps

The following steps are used to evaluate PGA460 function with MSPM0:

1. Prepare the necessary hardware equipment as mentioned in Section 2, and connect the necessary wires as shown in Table 2-1. For the used transducer muRata MA58MF14-7N, its available detection distance is about 0.2~6m. To get better test results, the obstacle in the test environment would be durable and wide range, like a wall.
2. Install CCS on the computer and prepare the code example as shown in Section 3.
3. Perform system parameter design with PGA460 GUI, as shown in Figure 4-1. For the process of GUI, see the *PGA460-Q1 EVM Quick Start Guide* and the *PGA460-Q1 Ultrasonic Signal Conditioner EVM With Transducer User's Guide* in the following URL: BOOSTXL-PGA460 Evaluation board|TI.com.



Record [ms] : 36.86
Digital Gain: x2
LPF Cutoff [kHz] : 2
BPF BW [kHz] : 4

58.6 kHz
16 Pulses
250 mA
72.5 dB

Note: All values updated on the block diagram only apply to Preset 1.

EVM Status
USB Controller : USB2ANY I/F Found
USB Firmware : 2.7.0.0
Connection Status : Connected
PGA460-Q1 Status : Ready; UART Addr = 0

DEV_STAT0
REV_ID 10
OPT_ID 00
CMW_WU_ERR 0
THR_CRC_ERR 0
EE_CRC_ERR 0
TRIM_CRC_ERR 0

DEV_STAT1
TSD_PROT 0
IOREG_OV 0
IOREG_UV 0
AVDD_OV 0
AVDD_UV 0
VPWR_OV 0
VPWR_UV 0

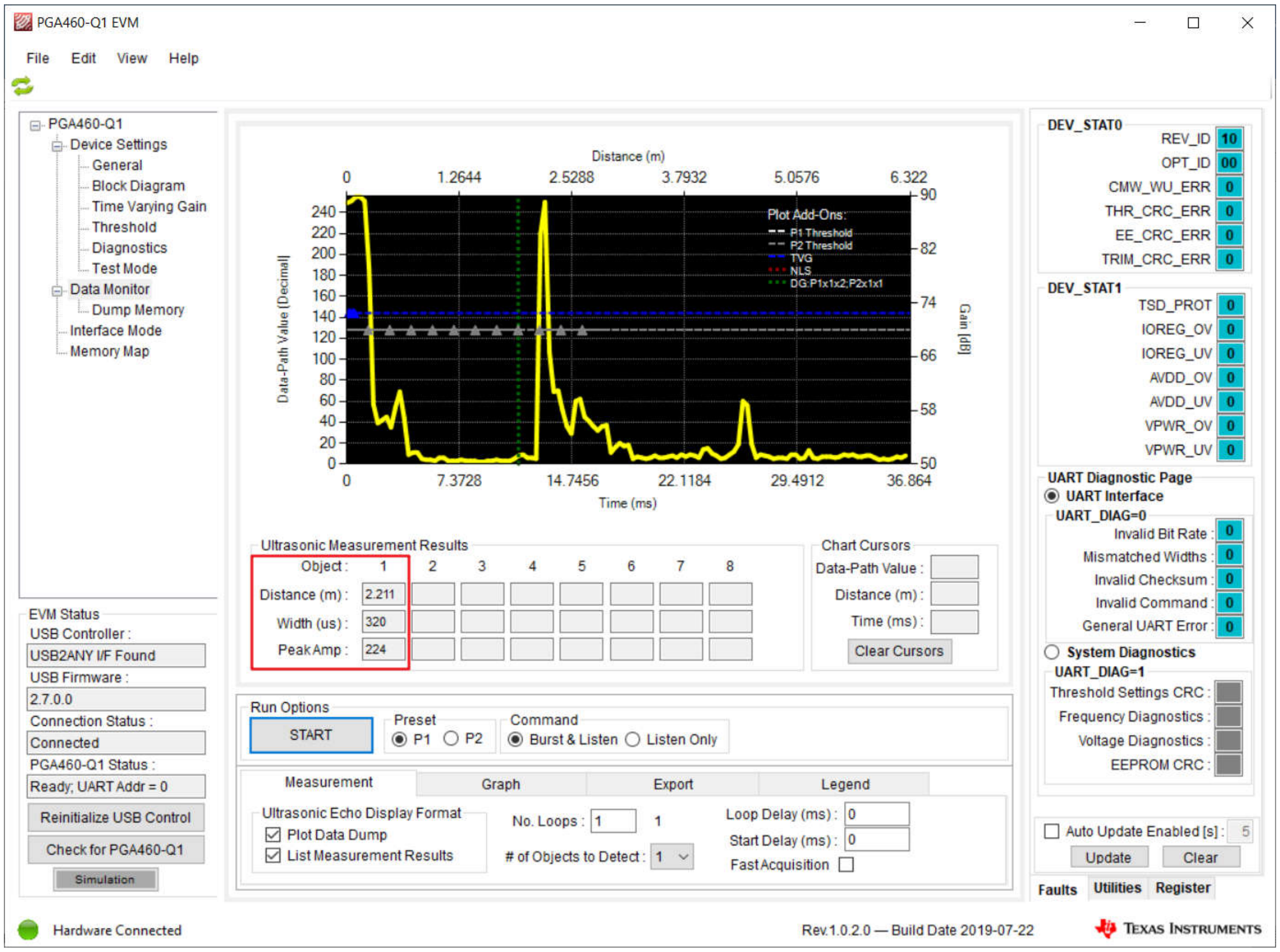
UART Diagnostic Page
[✓ UART Interface]
UART_DIAG=0
Invalid Bit Rate : 0
Mismatched Widths : 0
Invalid Checksum : 0
Invalid Command : 0
General UART Error : 0
System Diagnostics
UART_DIAG=1
Threshold Settings CRC :
Frequency Diagnostics :
Voltage Diagnostics :
EEPROM CRC :

Auto Update Enabled [s] : 5

Hardware Connected
Rev.1.0.2.0 — Build Date 2019-07-22

Figure 4-1. PGA460-Q1 EVM GUI

4. For customized transducer, you need to make parameter adjustments and check the detection results, as shown in Figure 4-2.



Ultrasonic Measurement Results

Object :	1	2	3	4	5	6	7	8
Distance (m) :	2.211							
Width (us) :	320							
Peak Amp :	224							

Chart Cursors
Data-Path Value :
Distance (m) :
Time (ms) :

EVM Status
USB Controller : USB2ANY I/F Found
USB Firmware : 2.7.0.0
Connection Status : Connected
PGA460-Q1 Status : Ready; UART Addr = 0

Run Options
Preset: [✓ P1] P2
Command: [✓ Burst & Listen] Listen Only

Ultrasonic Echo Display Format
[✓ Plot Data Dump]
[✓ List Measurement Results]
No. Loops : 1 1
# of Objects to Detect : 1
Loop Delay (ms) : 0
Start Delay (ms) : 0
Fast Acquisition ☐

DEV_STAT0	
REV_ID	10
OPT_ID	00
CMW_WU_ERR	0
THR_CRC_ERR	0
EE_CRC_ERR	0
TRIM_CRC_ERR	0

DEV_STAT1	
TSD_PROT	0
IOREG_OV	0
IOREG_UV	0
AVDD_OV	0
AVDD_UV	0
VPWR_OV	0
VPWR_UV	0

UART Diagnostic Page
[✓ UART Interface]

UART_DIAG=0	
Invalid Bit Rate :	0
Mismatched Widths :	0
Invalid Checksum :	0
Invalid Command :	0
General UART Error :	0

System Diagnostics

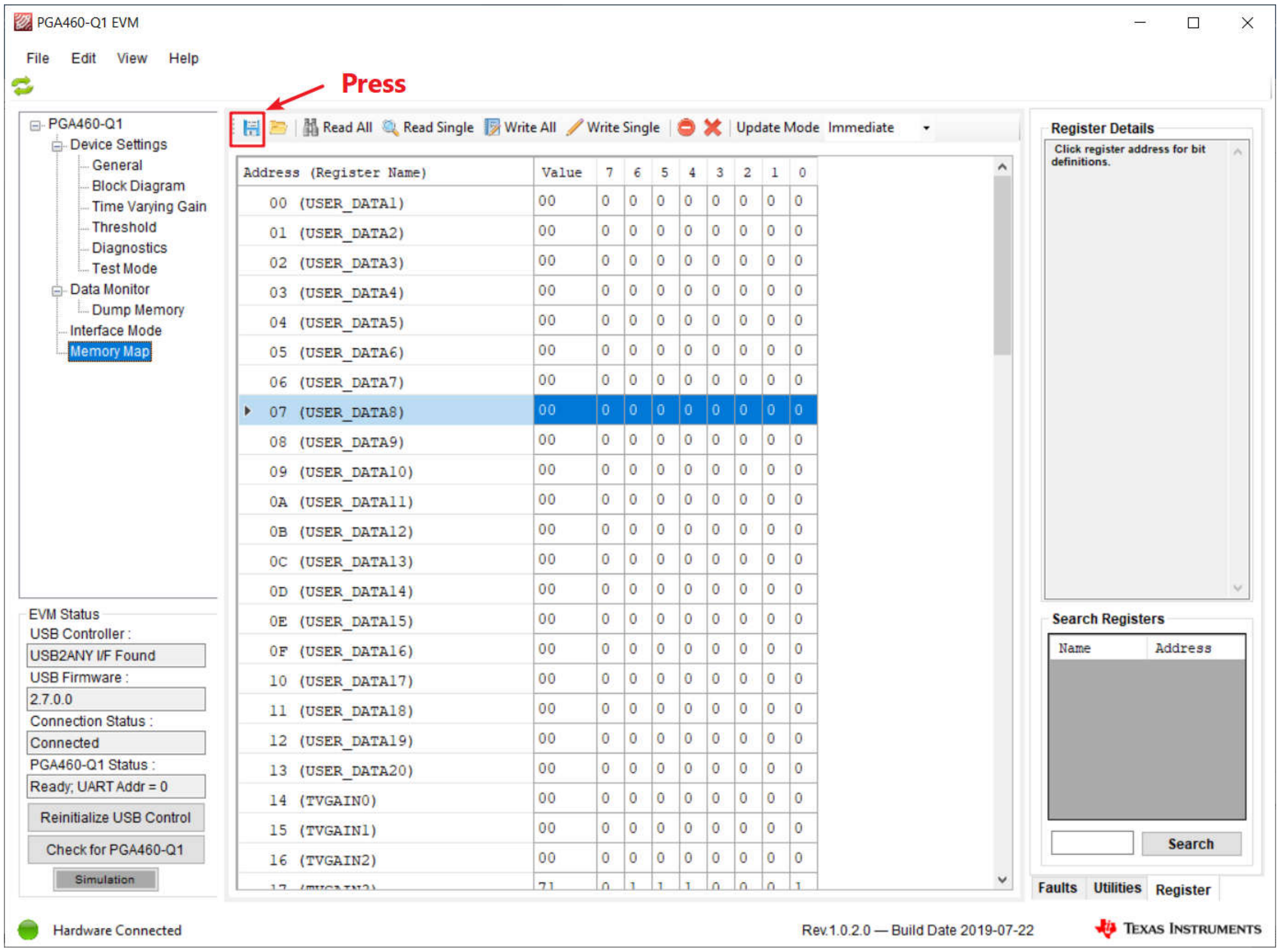
UART_DIAG=1	
Threshold Settings CRC :	
Frequency Diagnostics :	
Voltage Diagnostics :	
EEPROM CRC :	

☐ Auto Update Enabled [s] : 5

Hardware Connected

Figure 4-2. PGA460-Q1 GUI Detection Results

After the result meet the expectation, you can export the register configuration parameters as shown in Figure 4-3. Then, update the thresholds and configuration in “PGA460.h”. For more parameter adjustment, see the *PGA460 Ultrasonic Module Hardware and Software Optimization*. If you use the same transducer (muRata MA58MF14-7N), you can ignore this step.

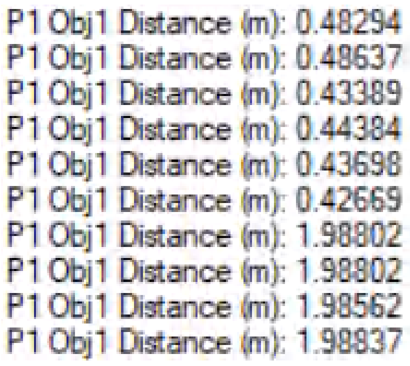


EVM Status
USB Controller : USB2ANY I/F Found
USB Firmware : 2.7.0.0
Connection Status : Connected
PGA460-Q1 Status : Ready; UART Addr = 0

Address (Register Name)	Value	7	6	5	4	3	2	1	0
00 (USER_DATA1)	00	0	0	0	0	0	0	0	0
01 (USER_DATA2)	00	0	0	0	0	0	0	0	0
02 (USER_DATA3)	00	0	0	0	0	0	0	0	0
03 (USER_DATA4)	00	0	0	0	0	0	0	0	0
04 (USER_DATA5)	00	0	0	0	0	0	0	0	0
05 (USER_DATA6)	00	0	0	0	0	0	0	0	0
06 (USER_DATA7)	00	0	0	0	0	0	0	0	0
07 (USER_DATA8)	00	0	0	0	0	0	0	0	0
08 (USER_DATA9)	00	0	0	0	0	0	0	0	0
09 (USER_DATA10)	00	0	0	0	0	0	0	0	0
0A (USER_DATA11)	00	0	0	0	0	0	0	0	0
0B (USER_DATA12)	00	0	0	0	0	0	0	0	0
0C (USER_DATA13)	00	0	0	0	0	0	0	0	0
0D (USER_DATA14)	00	0	0	0	0	0	0	0	0
0E (USER_DATA15)	00	0	0	0	0	0	0	0	0
0F (USER_DATA16)	00	0	0	0	0	0	0	0	0
10 (USER_DATA17)	00	0	0	0	0	0	0	0	0
11 (USER_DATA18)	00	0	0	0	0	0	0	0	0
12 (USER_DATA19)	00	0	0	0	0	0	0	0	0
13 (USER_DATA20)	00	0	0	0	0	0	0	0	0
14 (TVGAIN0)	00	0	0	0	0	0	0	0	0
15 (TVGAIN1)	00	0	0	0	0	0	0	0	0
16 (TVGAIN2)	00	0	0	0	0	0	0	0	0
17 (TVGAIN3)	71	0	1	1	1	0	0	0	1

Figure 4-3. Memory Map in GUI

5. Download the code and check if the measurement results in the serial port meet the requirements. Use a COM tool to catch the results send to PC through UART to USB, shown in Figure 4-4.



```
P1 Obj1 Distance (m): 0.48294
P1 Obj1 Distance (m): 0.48637
P1 Obj1 Distance (m): 0.43389
P1 Obj1 Distance (m): 0.44384
P1 Obj1 Distance (m): 0.43698
P1 Obj1 Distance (m): 0.42669
P1 Obj1 Distance (m): 1.98802
P1 Obj1 Distance (m): 1.98802
P1 Obj1 Distance (m): 1.98562
P1 Obj1 Distance (m): 1.98837
```

Figure 4-4. Running Results