Ultrasonic Distance Measurement using the TLV320AIC3268 miniDSP CODEC TIDA-00403

TI Design Guide

1.0 Introduction

Generic ultrasonic distance meters can provide poor results due to unknown variables in the system they are eventually used in. Issues such as transmitter and receiver ringing, burst control, and detection uncertainties affect the accuracy of the measurements as well as the usable range of these meters.

This design guide should be used as an aid for developing ultrasonic distance measurement algorithms using the TLV320AIC3268 (AIC3268) in conjunction with TI's PurePath Studio (PPS) design suite. Using the on chip miniDSP and efficient Class-D amplifier, a full customizable 2 chip solution can be designed. The ultrasonic burst generation characteristics as well as detection algorithms can be user modified to fit specific use cases in industrial applications allowing users to overcome the limitations of other fixed function sensors.

2.0 Getting Started

Here, we will setup the TLV320AIC3268EVM-U into an ultrasonic measurement solution. Using standard off the shelf components and TI EVMs, an evaluation system can be made. Once the design is up and running, the later parts of this document should be used as a technical reference for tuning and customizing the ultrasonic algorithms for the desired end use. After tuning has been completed, a customized PCB can be designed using just the TLV320AIC3268 and a host processor for I2C configuration of the TLV320AIC3268 and distance calculation integrated circuits and their supporting components.

For a background on ultrasonic distance measurement, see **Appendix A.**

2.1 Hardware

For ease of design, TI EVMs were used exclusively to get a full working system. This allows computer connectivity for easy programming and control of the ultrasonic algorithms as well as viewing of various DSP processing in real time on an oscilloscope using the PCM5242RHBEVM.

The user can always add extra peripherals into the end design such as oscilloscope monitoring as well other distance display method such as an LCD screen.

Required Hardware / Components:

Part	Description
TLV320AIC3268EVM-U	Host for PPS process flow and ultrasonic algorithms, Class- D transmitter output, mic receiver input
40TR12B-R	40KHz Ultrasonic transmitter (Tx) and receiver (Rx) pair
PCM5242RHNEVM	DAC used for viewing PPS process flows in real time on oscilloscope. (Can be removed for final design)
Oscilloscope	Used to view miniDSP processes in real time for debugging and design aid. Preferable 4 Channels
MCLK Source	12.288MHz source for AIC3268. Internal PLL derives BCLK and LRLCK as I2S master

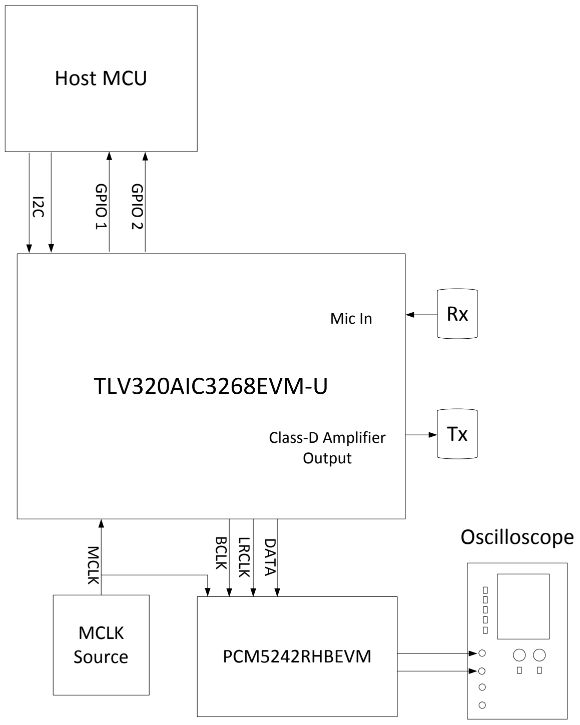


Figure 1: Hardware Block Diagram

Figure 1 shows the basic layout of the various components necessary to get this design up and running. The TLV320AIC3268EVM-U is the heart of the system. Using PurePath Studio, the TLV320AIC3268EVM-U can be configured for ultrasonic measurements. It then flags GPIO 2 when a ultrasonic burst is sent and GPIO 1 once the reflection has been received. Thus, the host MCU can monitor the 2 GIPOs for a Time of Flight TOF measurement to calculate distance. In final design, the Host MCU can also write the setup obtained in PurePath Studio to the AIC3268 via I2C. The PCM5242RHBEVM DAC is used for viewing the digital ultrasonic DSP data on an oscilloscope for verification. In final design.

2.1 TLV320AIC3268EVM-U Hardware Setup

The TLV320AIC3268EVM-U should be setup as indicated in Figure 2. The ultrasonic transmitter should be connected to the two terminals on the terminal block labeled *SPK OUT*. The receiver should be connected to *MIC IN* using a 3.5mm stereo jack between the ground and the innermost tip on a stereo tip/ring/sleeve stereo jack. USB Micro jack is used for computer power and communication for PurePath Studio setup

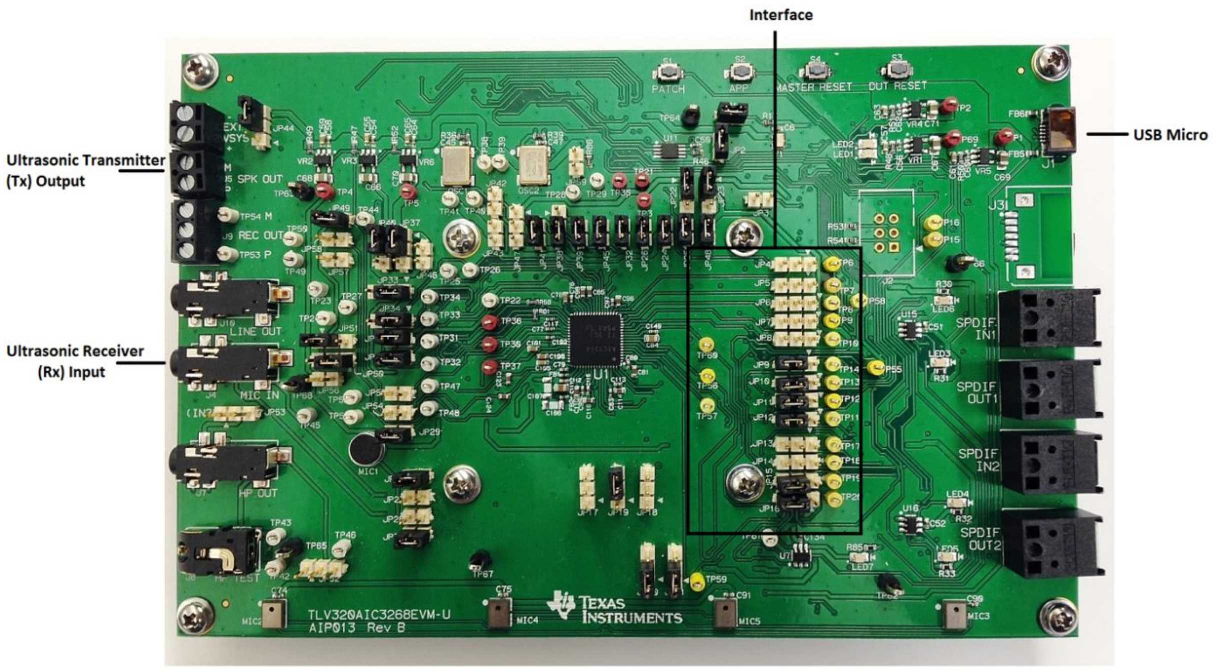

Interface
USB Micro
Ultrasonic Transmitter (Tx) Output
Ultrasonic Receiver (Rx) Input

Figure 2: TLV320AIC3268EVM-U Board Top View

TLV320AIC3268EVM-U Jumper Settings (Pin 1 indicated by the white arrow)**:**

JP1	Open	**JP16**	Shorted 2-3	**JP31**	Shorted	**JP46**	Open
JP2	Shorted	**JP17**	Open	**JP32**	Shorted	**JP47**	Open
JP3	Open	**JP18**	Open	**JP33**	Shorted 1-2	**JP48**	Shorted
JP4	See Figure 3	**JP19**	Shorted 2-3	**JP34**	Shorted 1-2	**JP49**	Shorted
JP5	See Figure 3	**JP20**	Shorted 1-2	**JP35**	Shorted	**JP50**	Shorted 1-2
JP6	See Figure 3	**JP21**	Shorted 1-2	**JP36**	Shorted	**JP51**	Shorted 1-2
JP7	Open	**JP22**	Shorted 2-3	**JP37**	Shorted 1-2	**JP52**	Open
JP8	See Figure 3	**JP23**	Shorted 2-3	**JP38**	Shorted 2-3	**JP53**	Open
JP9	Shorted 2-3	**JP24**	Shorted	**JP39**	Shorted	**JP54**	Open
JP10	Shorted 2-3	**JP25**	Shorted	**JP40**	Shorted 2-3	**JP55**	Open
JP11	Shorted 2-3	**JP26**	Shorted	**JP41**	Shorted	**JP56**	Open
JP12	Shorted 2-3	**JP27**	Open	**JP42**	Open	**JP57**	Open
JP13	Open	**JP28**	Open	**JP43**	Open	**JP58**	Open
JP14	Open	**JP29**	Shorted	**JP44**	Shorted 2-3		
JP15	Shorted 2-3	**JP30**	Shorted	**JP45**	Shorted		

The Interface jumpers and test points on the board allow for connection to external peripherals. The headers are 100 mil spaced with pin 1 indicated by the white arrow being GROUND. This allows for a shielded coaxial cable to be used to connect between the EVMs of the system.

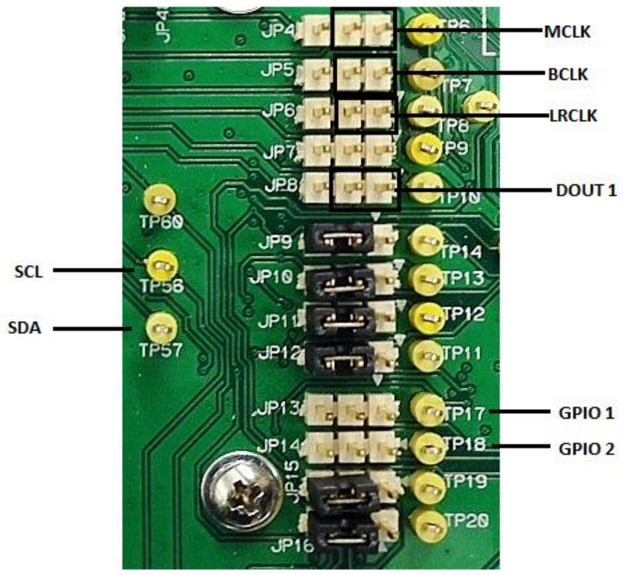


Figure 3: TLV320AIC3268EVM-U Interface Connections

TLV320AIC3268EVM-U Interface connections:

Port	Type	Connected To/From
JP4 MCLK	Input	12.288MHz Clock Source
JP5 BCLK	Output	PCM5242RHBEVM
JP6 LRCLK	Output	PCM5242RHBEVM
JP8 DOUT 1	Output	PCM5242RHNEVM (To Din 1)
TP18 GPIO 2	Output	Host MCU / Oscilloscope
TP17 GPIO 1	Output	Host MCU / Oscilloscope
TP56 SCL	Input/Output	Host MCU
TP57 SDA	Input/Output	Host MCU

2.2 PCM5242RHBEVM Hardware Setup

The PCM5242RHBEVM should be setup as indicated in figure 4 with the jumpers in the positions shown. The USB micro jack is used for computer power and control. The XLR outputs are used to connect to an oscilloscope for viewing of miniDSP processes within the AIC3268.

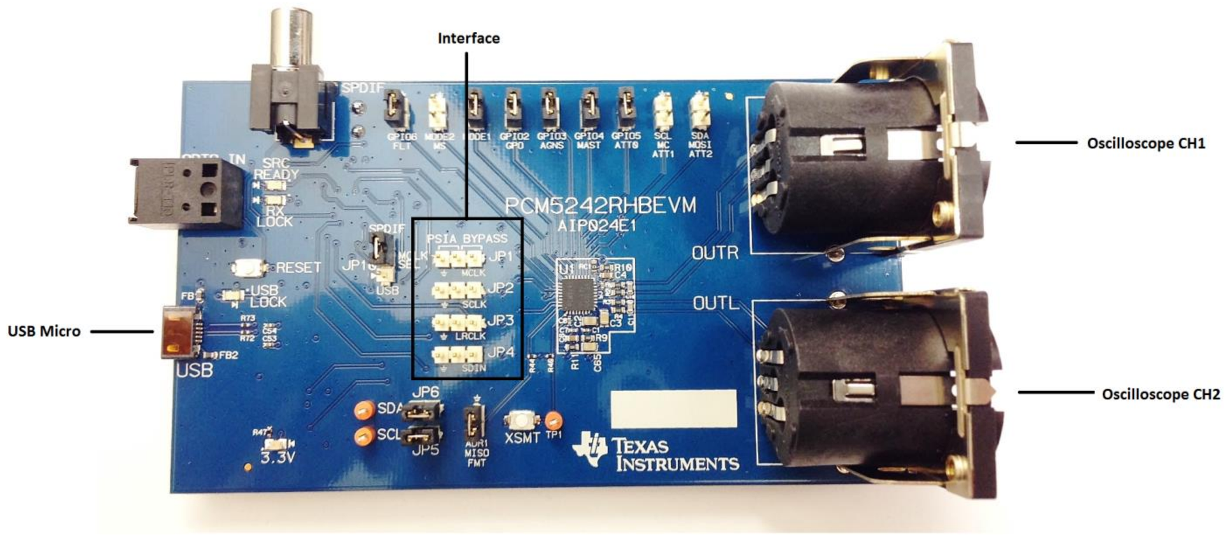


Figure 4: PCM5242RHBEVM Board Top View

PCM5242RHBEVM Jumper Settings (Pin 1 indicated by the white arrow)**:**

JP1	See Figure 5	**ADR1/MISO/FMT**	Shorted
JP2	See Figure 5	**SDA/MOSI/ATT2**	Open
JP3	See Figure 5	**SCL/MC/ATT1**	Open
JP4	See Figure 5	**GPIO5/ATT0**	Shorted
JP5	Shorted	**GPIO4/MAST**	Shorted
JP6	Shorted	**GPIO3/AGNS**	Shorted
JP10	Shorted 2-3	**GPIO2/GPO**	Shorted
		MODE2/MS	Open
		GPIO6/FLT	Shorted

The Interface jumpers on the board allow for connection to external peripherals. The headers are 100 mil spaced with pin 1 indicated by a GROUND symbol. This allows for a shielded coaxial to be used to connect between the EVMs of the system.

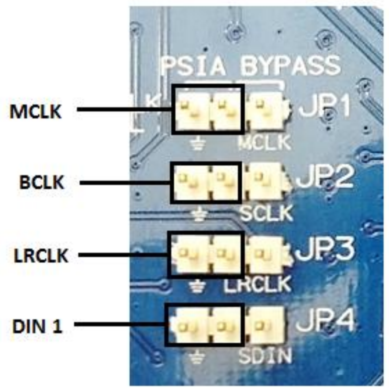


Figure 5: PCM5242RHBEVM Interface Connections

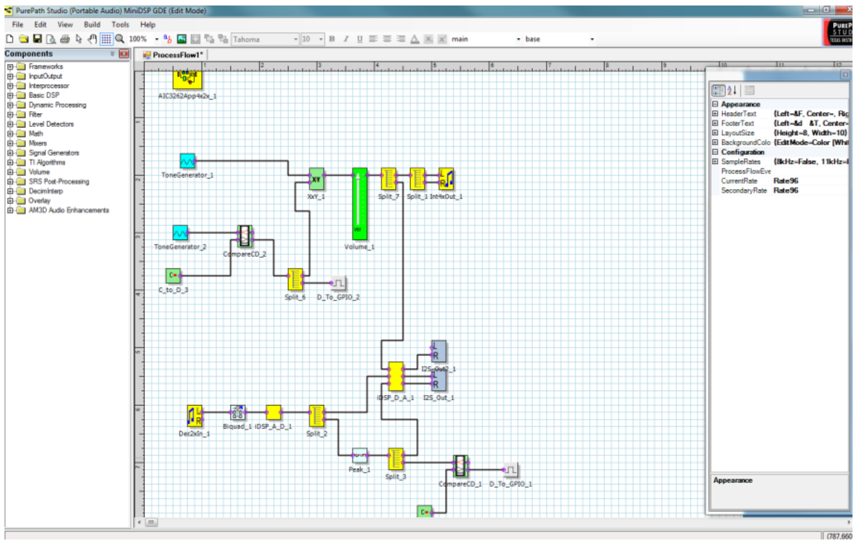
PCM5242RHBEVM Interface connections:

Port	Type	Connected To/From
JP1 MCLK	Input	12.288KHz Clock Source
JP2 BCLK	Input	TLV320AIC3268EVM-U
JP3 LRCLK	Input	TLV320AIC3268EVM-U
JP4 DIN 1	Input	TLV320AIC3268EVM-U (DOUT 1)

3.0 Running with PurePath Studio and Host Computer

The steps outlined here should be used during the customization and testing phase of the ultrasonic algorithms. A host computer will configure and load the PurePath Studio miniDSP configurations onto the AIC3268 which allows for rapid prototyping. Coefficients in PPS can be quickly changed and loaded on to the board to control ultrasonic characteristics.

1. Request permission and download PPS software form the AIC3268EVM web page here: http://www.ti.com/tool/AICPUREPATH_STUDIO
2. Download the TLV320AIC3204 Graphical User Interface software from the TLV320AIC3204EVM-K page here: http://www.ti.com/lit/zip/sloc128 (**Note**: this software is used for its generic command line interface features which custom I2C scripts can be written)
3. Connect EVMs and hardware as described in section **2.0 Hardware**.
4. Plug in the TLV320AIC3268EVM-U to the host computer via the mini USB.
5. From the TI Designs webpage, download the TIDA-00403 *Software* folder. Follow the installer instructions to download *TIDA-00403_AIC3268_Ultrasonic_ProcessFlow1.pfw* which will be found in the installed directory.
6. Open the PPS *GDE.exe* to run the PurePath Studio GUI.
7. Once the GUI is running, go to *File/Open* and navigate to the saved directory containing *TIDA-00403_AIC3268_Ultrasonic_ProcessFlow1.pfw.*
8. The process flow should now open and look like the following:



9. Insure that your MCLK source is on and connected to both the AIC3268 and the PCM5242 EVMs.
10. Then in the PPS GUI go to *Build/Download and Run.* The GUI should then download the process flow to the AIC3268. Once it has completed the background in PPS will turn a light blue.
11. Next the Class-D amplifier and Microphone input must be activated through I2C. **Note:** This is not controlled by PPS but can be added into the source code if desired. For design purposes, it was chosen to keep this script isolated from PPS to help with debugging. We can determine if our PPS script is causing issues with the Class-D output and microphone inputs.
12. Open the TLV320AIC3204EVM software downloaded earlier by right clicking and selecting ***Run as Administrator***.
13. Click on the *Command Line Interface* button on the GUI. A new window should pop up.
14. In the *Command Buffer* window copy and paste the flowing script using Ctrl+C and Ctrl+V key commands:

```
###############################################################################################
#
# ADC Input Channel Configuration --- IN2L / IN2R
###############################################################################################
#
w 30 00 01 # Select Page 1
w 30 08 00 # Set the input common mode to 0.9V
w 30 34 20 # Route IN2L and CM1 to LEFT ADCPGA with 20K input impedance
w 30 36 80
w 30 37 20 # Route IN2R and CM1 to RIGHT ADCPGA with 20K input impedance
w 30 39 80


###############################################################################################
#
# Output Channel Configuration
###############################################################################################
#
w 30 00 01 # Select Page 1
w 30 03 00 # Set PTM mode for Left DAC to PTM_P3 (default, writing here optional)
w 30 04 00 # Set PTM mode for Right DAC to PTM_P3 (default, writing here optional)
w 30 16 c3 # Enable DAC to LOL/R routing and power-up LOL/R
w 30 2E 0c # Route LOL to SPK @ -6dB
w 30 2F 0c # Route LOR to SPK_RIGHT_CH_IN @ -6dB
w 30 30 11 # Set SPK Gain @ 6dB, unmute SPK_RIGHT_CH_IN
```

```
w 30 2D 06 # Power-up SPK, route SPK_RIGHT_CH_IN to SPK
w 30 00 00 # Select Page 0
w 30 3f c0 # Power up the Left and Right DAC Channels
w 30 40 00 # Unmute the DAC digital volume control
```

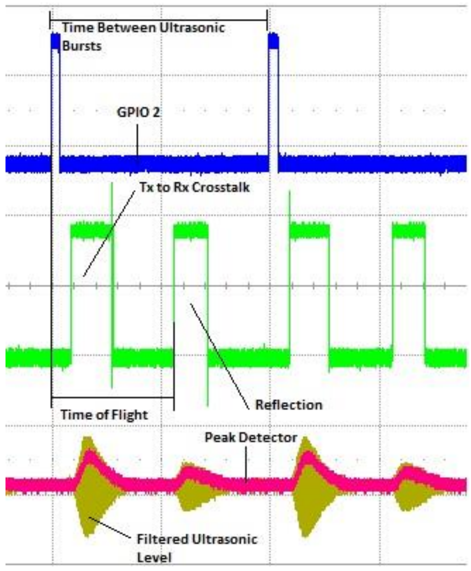
Note: This script is used to configure the microphone input and Class-D amplifier. It has not been integrated into the PPS process flow to aid in debugging. See **Appendix C**.

15. Click *Execute Command Buffer* button. The *STATUS* light in the window to the right should display green. The write was successful.

16. Plug in the USB cable from the PCM5242RHBEVM to the same host computer.

17. If everything was setup correctly, you should now be able to view the ultrasonic processes on an oscilloscope including GPIO triggers as well as microphone signal and peak detector.

It should look like this if scaled correctly:



18. By placing an object closer or further from the transmitter and receiver you should see the detected reflection shift relative to the ultrasonic burst transmissions in Blue.

For final design it is desirable to run the setup without a host computer.

Once coefficients in the PPS process flow have been defined a host computer is no longer needed. A host MCU such as an MSP430 can be used to store the AIC3268 PPS configuration and write the correct register values to the AIC3268 upon startup via I2C. After initial startup, the host MCU can be used to monitor the state of GPIO 1 and GPIO 2 and determine the distance measured through a TOF calculation.

4.0 Going Further

1. Use a MCU to read GPIO 1 and GPIO 2 for distance calculations
2. Store PPS configuration on the MCU to configure the AIC3268 on startup via I2C to remove the computer and PPS dependency.
3. Store multiple PPS configurations for different measurement modes
4. Automatically detect correct measurement mode with multiple configurations

5.0 Design Overview

Simplified Ultrasonic Measurement Algorithm Block Diagram

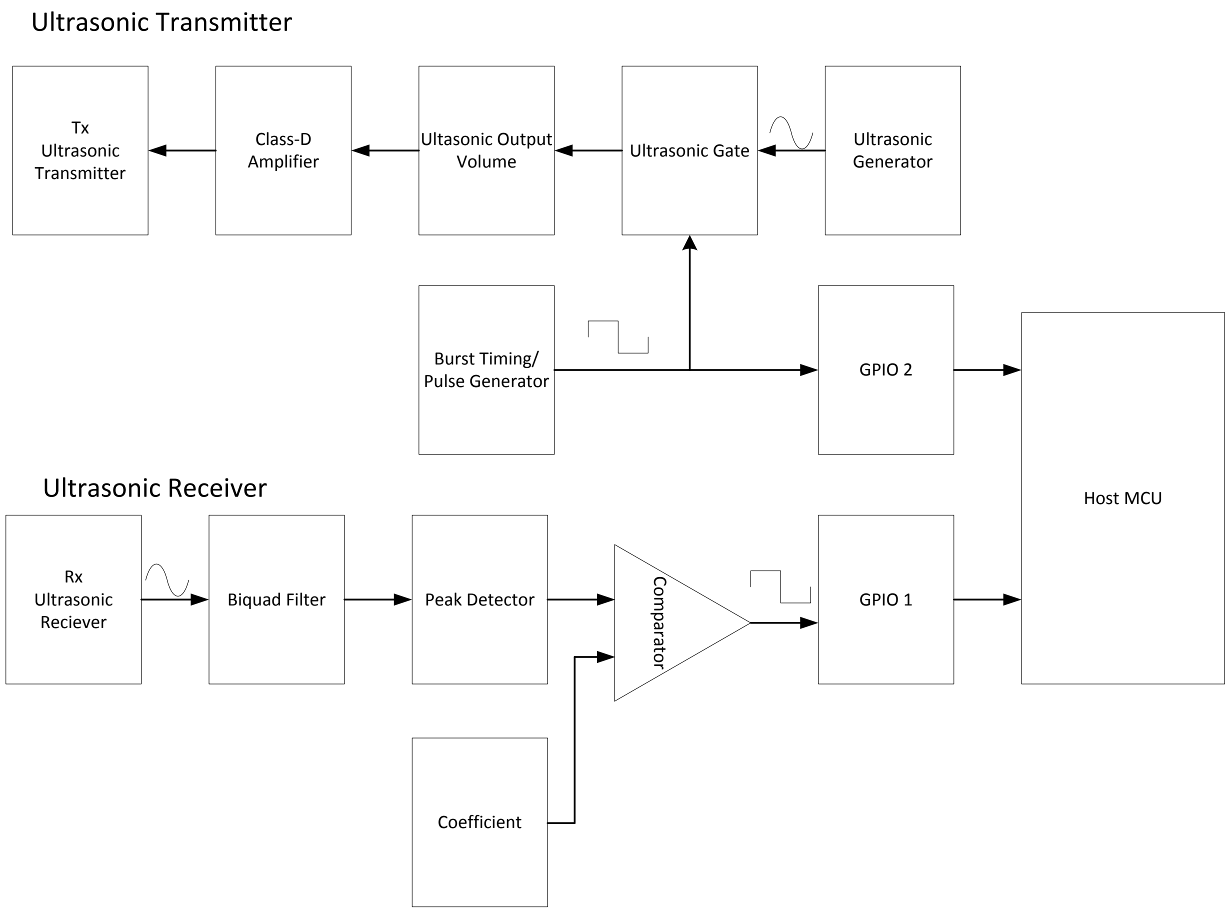


Figure 6: A simplified block diagram of ultrasonic algorithm executed in hardware and PPS

PurePath Studio process flows are initially executed using a host computer with USB to setup and the TLV320AIC3268EVM. Coefficients are loaded into each processing block in the designed PPS process flow which allows for optimization of specific measurement cases. The PCM5242RHBEVM is used to view the miniDSP processes in real time on an oscilloscope while monitoring the GPIOs. After which, the PPS generated header (.h) file, which contains the final setup scripts, can be formatted and saved to the MCU for automatic configuration of the AIC3268 for standalone operation.

See **Appendix B** for information on the .h file

After startup and configuration, an MCU can monitor the timing on GPIO 1 and GPIO 2 which are designed to pulse every time an ultrasonic burst is sent and an ultrasonic reflection is received. GPIO 2 displays the transmitted burst. GPIO 1 displays the received burst. Based on the timing of these GPIOs the MCU can then calculates the distance from an object.

MCLK source was the MCLCK out an Audio Precision analyzer. For final design, a clock from the host MCU would be used.

5.1 Clocking

In this design the TLV320AIC3268 is setup as an I2S master, only requiring an external MCLK source for complete function. Master mode was chosen in order to view various processes during the PPS design phase through an external PCM5242RHBEVM high performance DAC. An I2S output bus can be implemented at various points in the PPS process flow so the internal miniDSP functions can viewed in real-time on an oscilloscope.

The sample rate of the AIC3268 was selected as 96KHz Fs as to allow for 40KHz ultrasonic operation.

Note: the sample rate must be at least twice the speed of the ultrasonic frequency to be used.

Nominal Ultrasonic Frequency	40KHz
MCLK	12.288MHz
BLCK	6.144MHz
LRCLK = (Fs)	96KHz

Note: PurePath Studio does not have control over the clocking of the AIC3268 other than the sample rate. To configure the PLL clocks correctly, the following script was added into the PPS framework code:

```
reg[0][4][11] = 0x01                          ; ASI1_BDIV = DAC_Mod Clk
reg[0][4][12] = 0x81                          ; Bit clk N = 1
reg[0][4][10] = 0x24                          ; Bitclk & WClk outputs
```

6.0 PurePath Studio Process Flow

The flowing sections below describe the PPS process flows used for this ultrasonic measurement design. The coefficients loaded into each functional block in the process flow can be change and optimized for a specific use case.

The process flow can be directly run from a host computer to the EVM or for standalone operation. With an MCU where the .h file can be used to write the code. See **Appendix B.**

Note: the AIC3262App4x2x_1 framework is used for this design with AIC3268EVM.

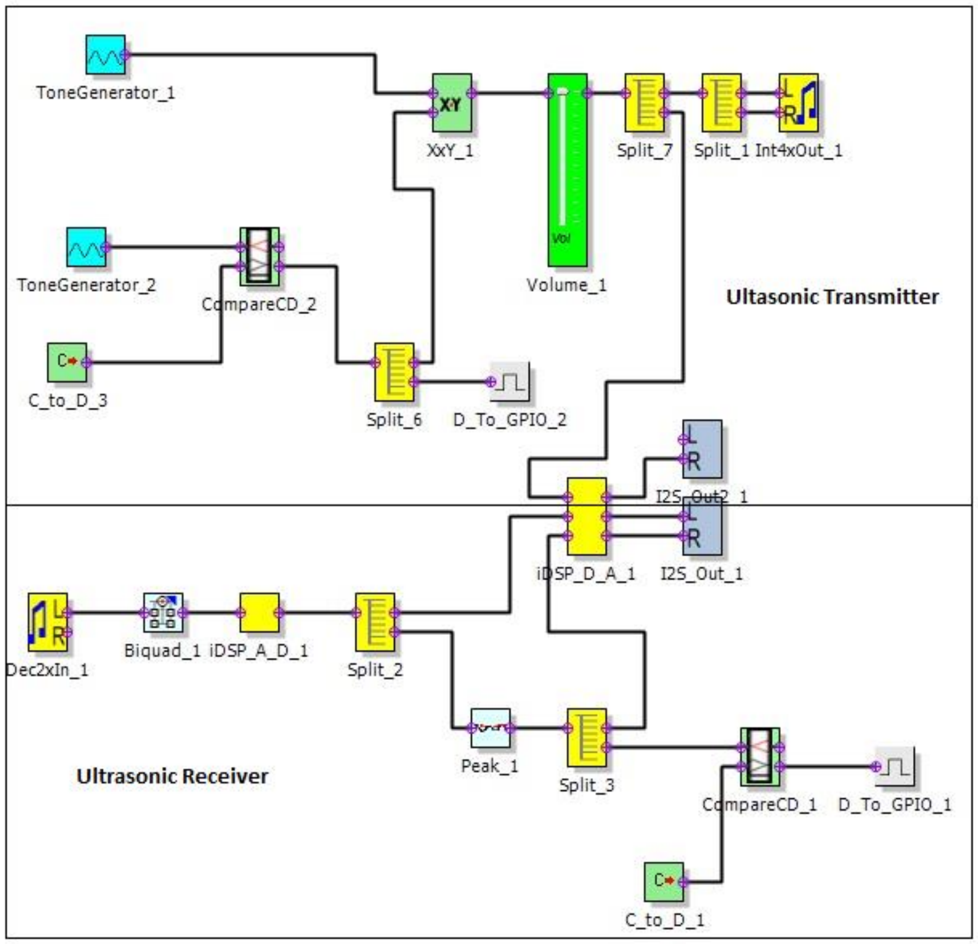


Figure 7: PurePath Studio Process Flow

1. Ultasonic Transmitter

This section of the process flow self-generates a periodic ultasonic burst at 40KHz. The period and duty cycle of the burst can be tightly controlled for different use cases.

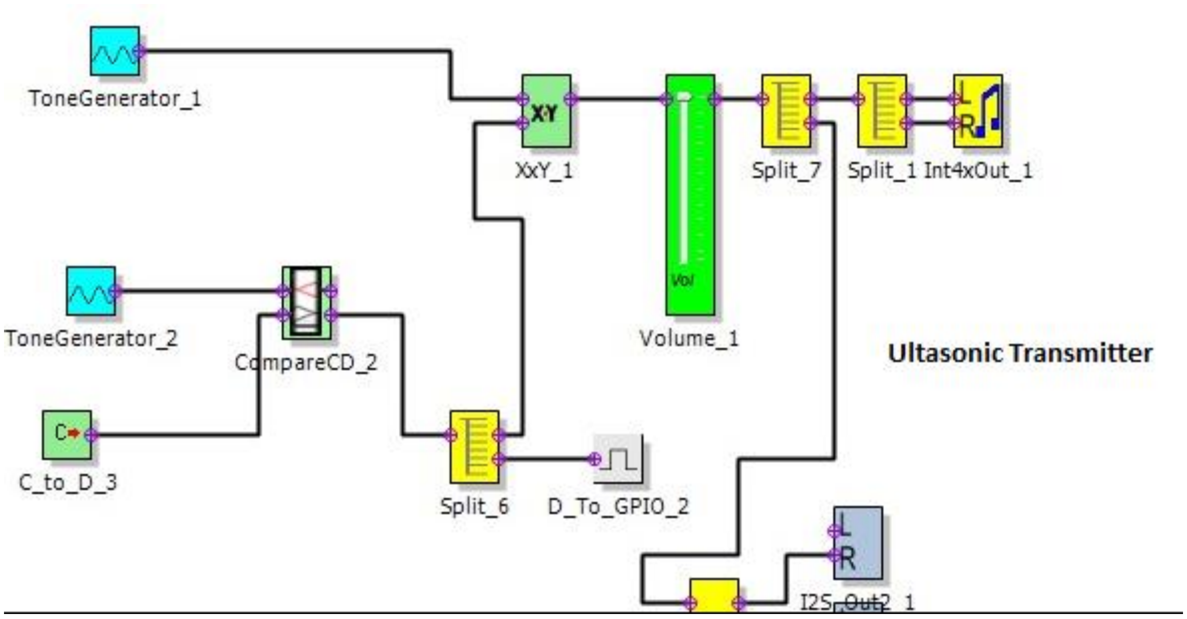


Figure 8: Ultasonic Transmitter Process Flow

a. ToneGenerator_1

This block produces the 40KHz ultrasonic wave which can be simply generated by typing 40000 for the frequency in runtime properties. The amplitude is set to the maximum of 1.

b. Burst Timing Block

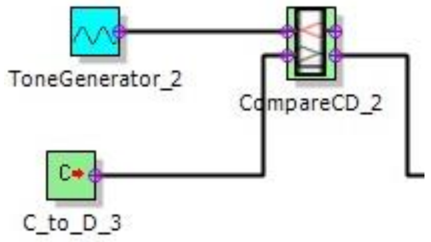


Figure 9: Burst Timing Block consists of ToneGenerator_2, C_to_D_3, CompareCD_2

This subsystem is a simple way to generate a periodic square wave pulse timer using a comparator. The wave form generated is a 100Hz sine wave that then gets compared against data in CompareCD_2. C_to_d_3 is a coefficient to data converter that stores a coefficient. When the amplitude of the sine wave is greater than the coefficient stored, the output is a logic level high. The logic level is driven low once the sine wave amplitude falls below the level of the coefficient. Thus we have a periodic pulse generator which will be used to control the ultrasonic output burst.

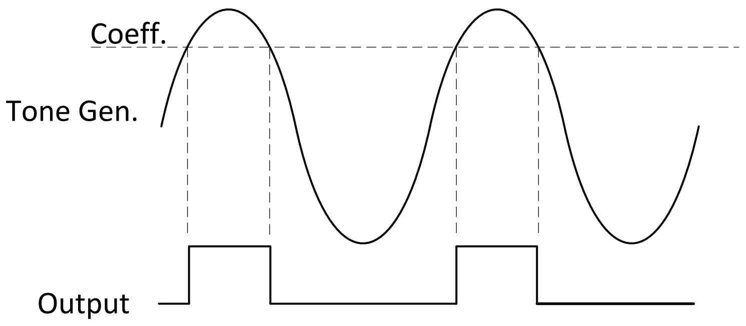


Figure 10: Operation of burst timing block

From figure 10 above, it should be evident that by changing the frequency of the tone generator, the period of the burst can be controlled. By changing the comparison coefficient, the width or duty cycle of the burst can also be controlled independently.

With this flexibility, the designer has complete control over the length of time of the ultrasonic burst as well as the time delay before the a burst automatically triggers again. This is especially useful for making distance measurements of close objects since the burst time is directly proportional to the minimum measurable distance. However, very short burst time is not optimal under all circumstances. Testing of the system shows that longer burst are better for detecting object that are far way.

The period of the bursts is also equally important. For objects far away the user must insure that additional bursts are not sent before a reflection is received. The tone generator frequency can be decreased to achieve this goal.

c. Split_6

Splits the CompareCD_2 signal between D_To_OGPIO_2 and XxY_1

d. D_To_GPIO_2

D_To_GPIO_2 takes the generated pulse from **6.1 b.** Burst Timing Block and send the data to GPIO_2. The signal on GPIO 2 now replicates the pulse used to control the ultrasonic frequency transmission. This GPIO is then available for monitoring by a host MCU which is used to determine the timing of when an ultrasonic burst was initially transmitted.

Note: Configuration of GPIO 2 is not controlled by PPS but was added to the framework code:

```
reg[0][4][87] = 0x0d                        ;Set GPIO2 as Output Driven high
reg[0][0][120] = 0x80                      ; Set GPIO Write Capability
```

e. XxY_1

XxY_1 is a multiplication block which uses the pulse from section **6.1 b**. Burst Timing Block and multiplies its value by ToneGnerator_1. Since the pulse generated is either logic 0 when low or logic 1 when high, every time the pulse is high, the ultrasonic frequency is passed through the multiplication function. The pulse generated in Burst Timing Block now actually controls the burst timing of the ultrasonic signal.

f. Volume_1

Controls the output level amplitude of the ultrasonic wave. Large amplitude increase transducer ringing but allow for further distance measurements. Small amplitude signals reduce transducer ringing making it easier to get close distance measurements.

g. **Split_7**

Splits ultrasonic signal into 2 identical signals for external monitoring via I2S if desired.

h. **Split_1**

Makes copies of the input signal for proper output to Class-D amplifier for ultrasonic transmission.

i. **Int4xOut_1**

Output of AIC3268 Interpolator which will send ultrasonic signal to the onboard Class-D amplifier.

Note: Configuration of the Class-D amplifier is done outside of PPS and an additional script is needed. See **Appendix C**. This script can be added into the PPS formwork code if reformatted. This should run at the end of configuration.

2. Ultrasonic Receiver

This section of the process flow determines the receiver detection characteristics. This system sets a threshold which determines the necessary received ultrasonic level to drive the detection GPIO 1 high.

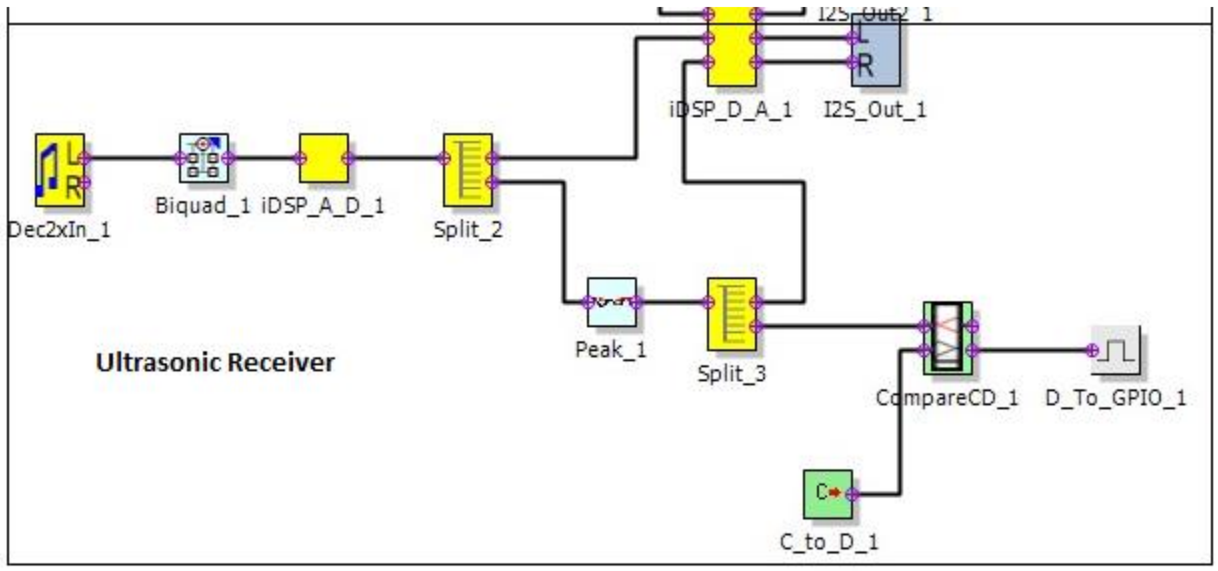


Figure 11: Ultrasonic Receiver Process Flow

a. Dec2xIn_1

This is the input decimator which will take in all of the raw data from the ultrasonic receiver connected to the microphone input on the EVM.

Note: Configuration of the microphone input is done outside of PPS and an additional script is needed. See **Appendix C**. This script can be added into the PPS formwork code if reformatted. This should run at the end of configuration.

The microphone PGA gain is not controlled by PPS but was added to the framework code:

```
reg[0][1][59] = 0x5f                           ;Set Mic PGA gain 47.5dB
```

b. Biquad_1

One benefit of AIC3268 is the biquad filters. The filter is used to remove all content other than the ultrasonic frequency of interest. This improves receiver detection capabilities and insures that receiver triggering is only on the frequency band of interest. This improves the reliability of the measurements considerably since background noise is less likely to trigger detection.

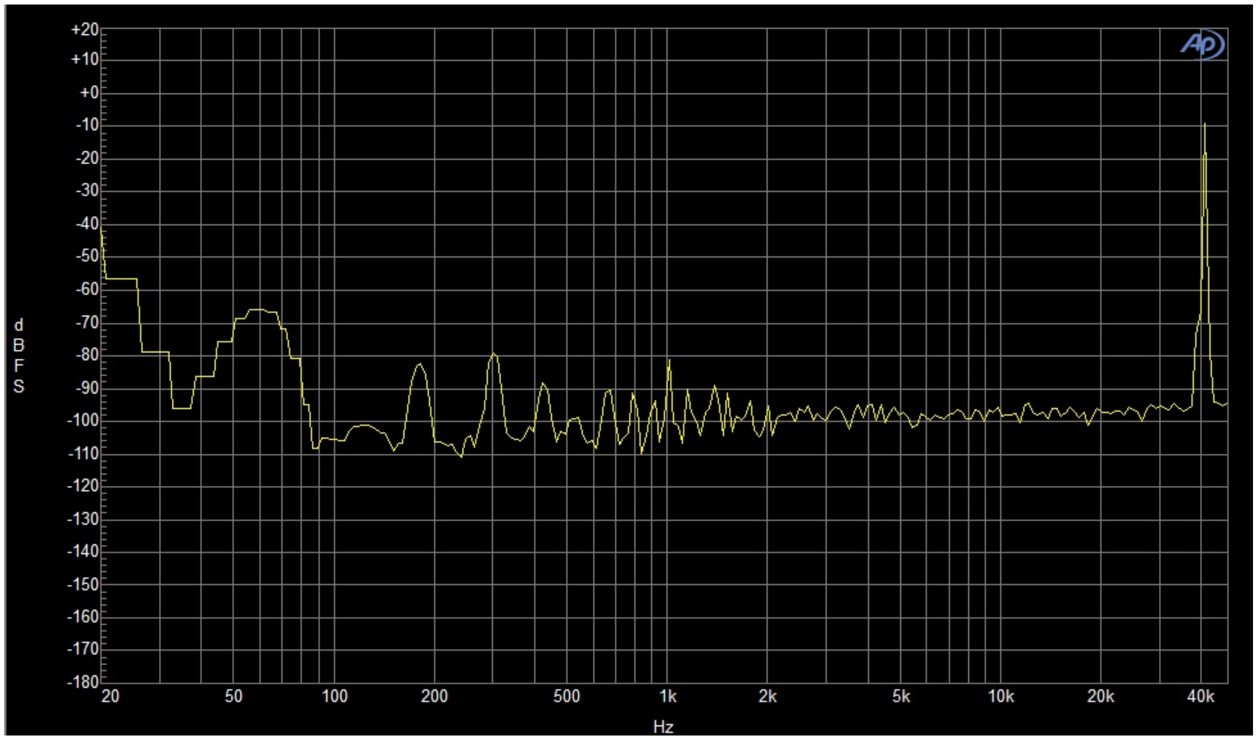


Figure 12: Ultrasonic Receiver without Biquad Filter

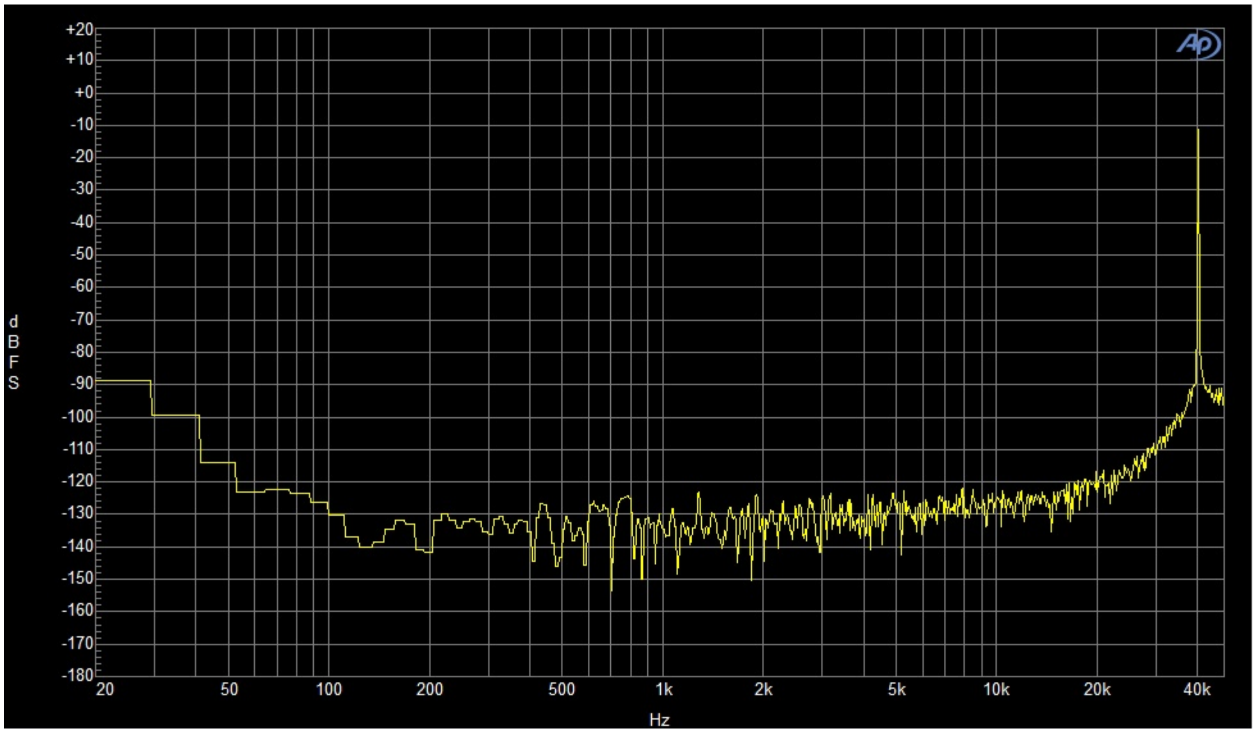


Figure 13: Ultrasonic Receiver with Biquad Filter

From figure 13 above we can see that the biquad filter significantly reduces the noise floor of the microphone. Background noise now has a low risk of trigging false detection.

For this design the flowing PPS filter was used:

Filter Type	High Pass
Filter Subtype	Chebychev
Fc (Hz)	40,000.00
Ripple (dB)	5.00
Scale (linear)	1

This filter displayed the most noise reduction over other filter types. Cascaded filters could also be used for even more flexibility.

c. iDSP_A_D_1

This is a transitional block that moves between miniDSP core A and miniDSP core D.

d. Split_2

Splits the signal so one copy can be sent to the I2S bus for monitoring the incoming signal from the ultrasonic receiver. The second copy goes to the receiver triggering block.

e. Receiver Detector and Triggering

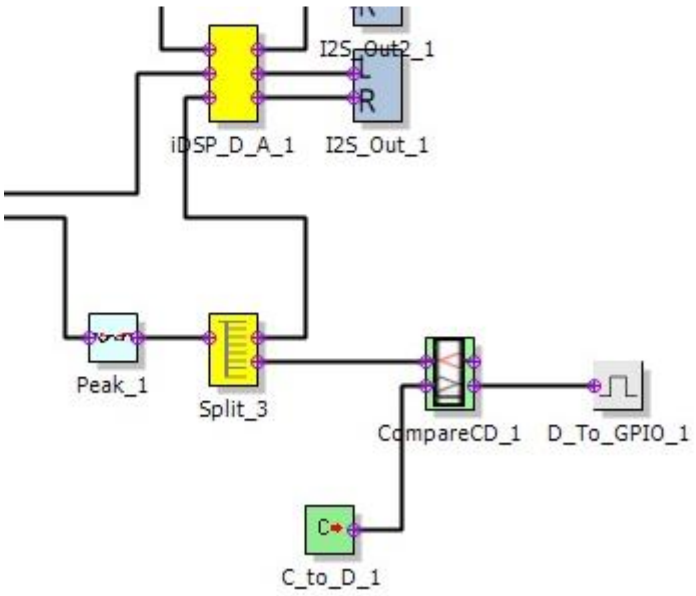


Figure 14: Shows the Receiver Detector and Triggering System

This section in figure 14 takes the biquad filtered signal from the ultrasonic receiver and determines if a reflected ultrasonic burst has been detected by the device.

Peak_1 is a peak detector that first envelopes the received ultrasonic burst. This then gets passed through a splitter so that the peak detector can be monitored on an I2S bus. The copy then goes to a comparison function that compares the peak envelope to a coefficient. The coefficient then determines the trigger threshold. If the enveloped received burst is greater than the coefficient, GPIO 1 is driven high and the receiver has detected a reflection.

Note: Configuration of GPIO 1 is not controlled by PPS but was added to the framework code:

```
reg[0][4][86] = 0x0d                     ;Set GPIO1 as Output driven high
```

This allows a triggering threshold to be set which is an aid in noisy environments. The triggering threshold is also useful for close up measurements.

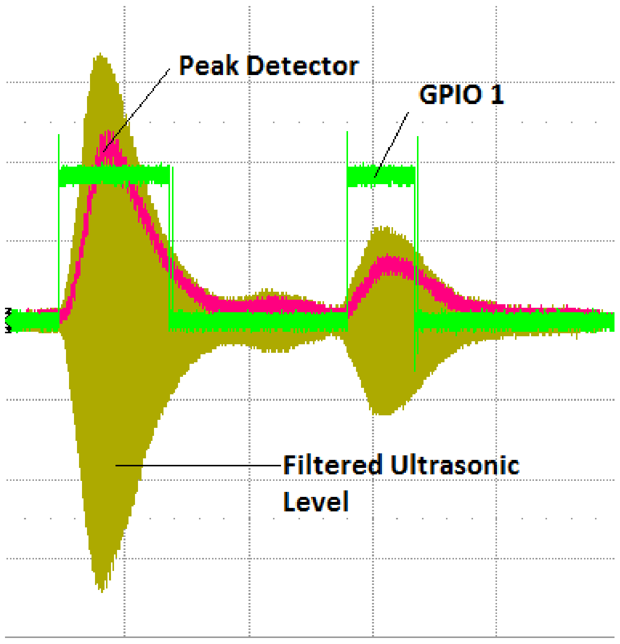


Figure 15: Receiver Detector with GPIO Triggering Viewed on Oscilloscope with PCM4242RHBEVM

Note: For very close distance measurements, increasing the threshold of the trigger as well as reducing the amplitude of the transmitted burst will not only reduce transceiver ringing, but can insure that the Tx to Rx crosstalk does not trigger the receiver detector. Therefore, the total receiver wait time before monitoring for a reflection is zero! The receiver detector never sees a crosstalk pulse and thcan immediately monitor for a reflection making the minimum measurable distance very small.

Based on our testing, distances as small as 4.0 inches were accurately detected with the proper coefficients entered into the process flow.

7.0 Example Measurement

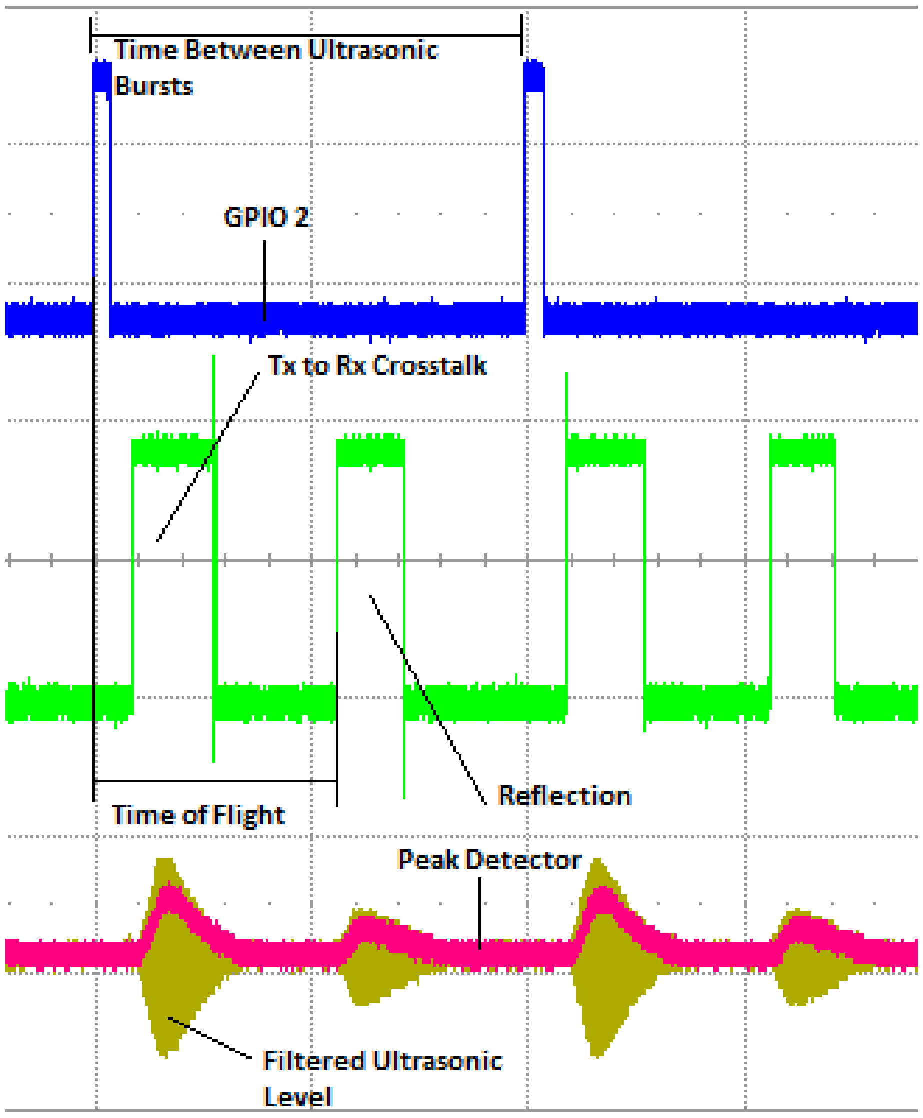


Figure 16: Waveforms for an Example Measurement.

In figure 16 above, the periodic transmitted burst of ultrasonic energy can be seen by the GPIO 2 trace in blue. In this example the Tx to Rx crosstalk can be seen. We must wait for this to pass before we can monitor GPIO 1 in green for a reflection.

We can also see the enveloping feature of the Peak Detector in red which is used to then trigger GPIO 1. When the level is above the set threshold, GPIO 1 is triggered. For this example the coefficients in the PPS Process Flow was optimized for long distance measurements.

8.0 Sources of Error

With this design, there was one observed source of error. GPIO 2 which is used to determine when an ultrasonic burst is transmitted triggers ahead of when the ultrasonic burst is actually transmitted by the Class-D amplifier. This is caused by latency in the process flow.

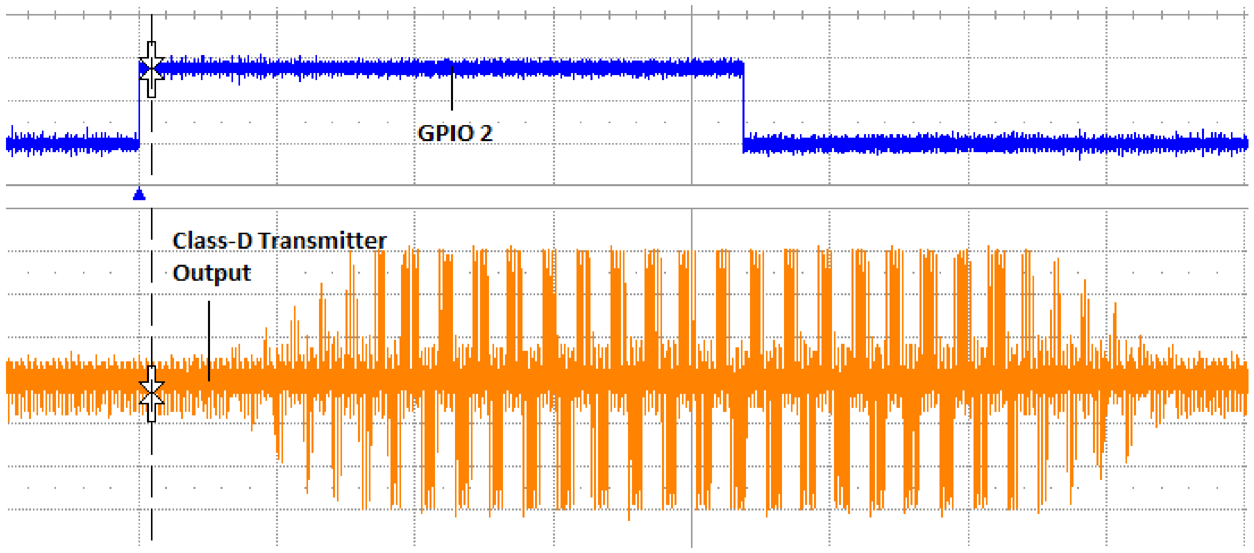


Figure 17: Transmission Timing Error

In figure 17 above, the blue trace from GPIO 2 is used to determine when an ultrasonic burst is sent by a host MCU. However, the orange trace depicts the actual ultrasonic transmission from the AIC3268 Class-D amplifier which is lagging behind GPIO 2 by around 100uS. This has the effect of making the measured distance to an object longer than it actually is.

Since this error is constant, it can be easily removed in the code used to calculate the distance of an object. A calibration factor can be used to do so.

Appendix A: Theory of Ultrasonic Distance Measurement:

Distance measurements using ultrasonic frequencies are made by a simple Time Of Flight (TOF) calculation. A burst of ultrasonic energy is transmitted into the air through a transmitter. This wave front collides with an object and scatters. A microphone or receiver oriented on the same plane as the transmitter detects the scattered pulse. If the time between the initial transmitted burst and received pulse is known, then the distance between the ultrasonic device and the object is known, assuming that the speed of sound is constant within the traveled distance.

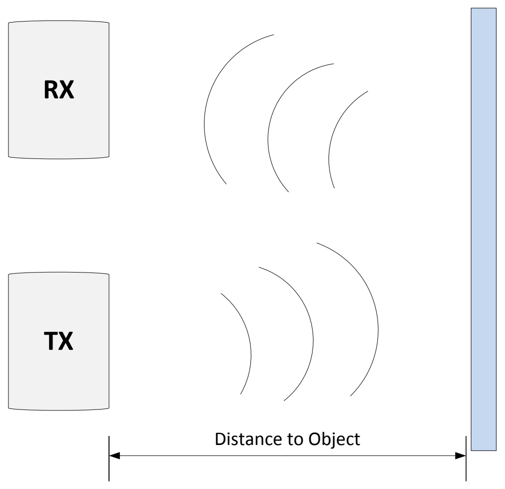


Figure 18: Transmitter Tx sends a ultrasonic pulse to a planer object where the reflection is received by Rx

Since the speed of sound is dependent on air temperature it we can make an assumption that the device will be used in room temperature or 20 degrees Celsius for calculations. This is generally a decent assumption because fluctuations in temperature have a minimal effect of the speed of sound. However, for more accuracy and for better performance over very wide operating temperature, the air temperature can be measured and added into the calculation.

$$Speed\ of\ Sound = 331.4 + (0.6\ \times\ ℃)\ M/S$$

Equation 1

Assuming $℃ = 20$ (Room Temperature)

$$Speed\ of\ Sound = 343.4\ M/S = 1130.1\ Ft/S = 13561.2\ in/S$$

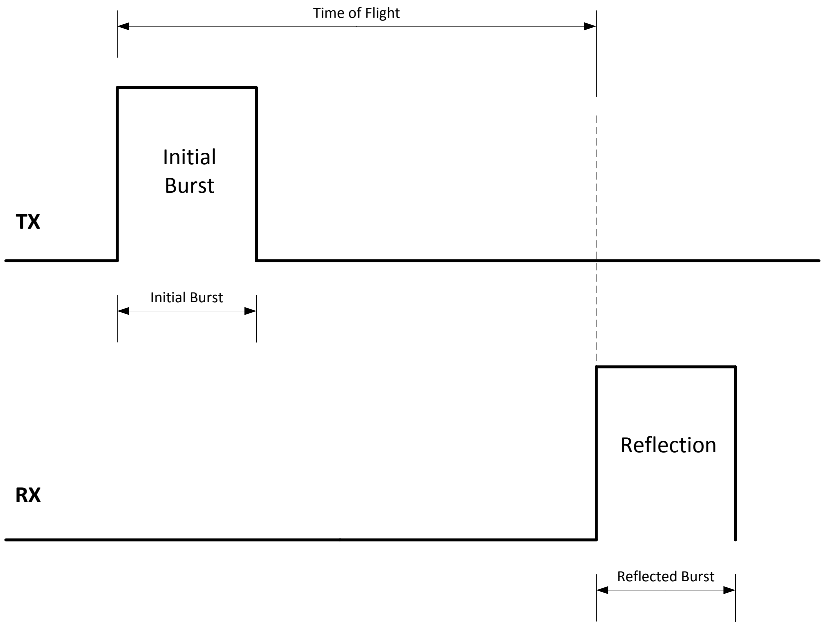


Figure 19: Time of flight is measured time from the starting edge of the transmitter pulse to the starting edge of the receiver detection pulse

Once the time of flight is determined and the speed of sound known, the distance from an object can be calculated.

$$Distance\ from\ Object\ = \frac{Time\ of Flight\ (S)}{2\ \times Speed\ of\ Sound}$$

Equation 2

In equation 2, there is a factor of 2 in the denominator since the time of flight from transmitter to receiver is twice that of the time from the transmitter to the object that causes the reflection. The ultrasonic pulse travels twice as far since it also must travel back to the receiver after the wave reflects off of the object of interest.

A1. Transmitter and Receiver Crosstalk

One of the most prominent issues with ultrasonic sensors is the crosstalk between the transmitter and receiver. When the transmitter sends an outgoing burst, the nearby receiver will receive a time delayed copy of the outgoing burst due to the proximity of the transmitter and receiver. In order to accurately determine that a reflection from an object was observed by the receiver, not the crosstalk from the initial burst, there must be a receiver wait time before the receiver begins monitoring for a reflection. This total receiver wait time must be adequate for the crosstalk volume level to decay below the detection trigger threshold of the receiver.

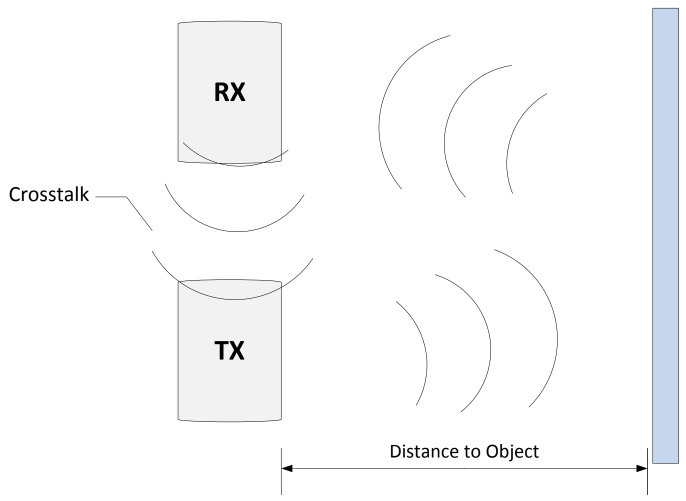


Figure 20: Crosstalk

This distance based time delay is one factor that determines the minimum distance the ultrasonic sensor can measure. This is one reason why a burst is used rather than a continuous wave front. If we could place the transmitter and receiver at the same exact point in space, then there would be no time delay. Therefore the receiver would see an exact copy of the transmitted burst at exactly the same time.

If we could achieve this, then only the duration of the transmitted burst would be a limiting factor.

A2. Transmitter Burst Duration

Transmitter burst duration is the second limiting factor in demining the minimum measurable distance between the ultrasonic sensor and an object. The shorter the burst time, the quicker the crosstalk decays and the minimum measurable distance is shortened. The necessary total receiver wait time before accurately detecting a reflection will be the sum of the time of flight between receiver and transmitter, as discussed above, and the burst duration.

A3. Total Receiver Wait Time Determines Minimum Detectable Distance

The total receiver wait time is the theoretical total wait time before the ultrasonic receiver can accurately begin detecting a reflection. This is the sum of the time of flight for the transmitter and receiver crosstalk as well as the transmitter burst duration.

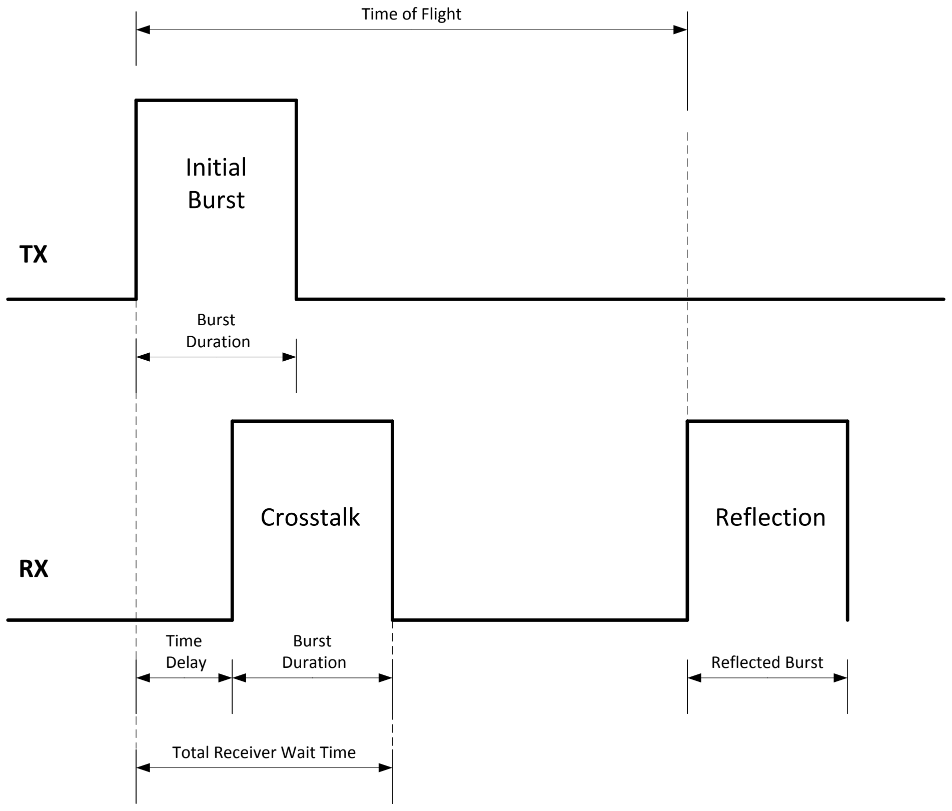


Figure 21: The minimum detectable distance is limited by the sum of burst duration and the crosstalk time delay due to the distance between Tx and Rx.

$$Total\ Reciever\ Wait\ Time = Burst\ Duration + \frac{(Rx - Tx\ Distance)}{Speed\ of\ Sound}$$

Equation 3

$$Minimum\ Detectable\ Distance = \frac{Total\ Reciever\ Wait\ Time \times Speed\ of\ Sound}{2}$$

Equation 4

A4. Transducer Ringing

In practice, the minimum detectable distance is larger than Equation 4 above. This is caused by transducer ringing, an observed phenomenon with this design. Transducer ringing in effect ultimately makes the burst duration longer than the actual time it was triggered. Not only is the transmitted burst longer due to ringing of the transmitter but the receiver will add yet another time delay due it its own ringing of the received ultrasonic wave. The result is a teardrop shaped ultrasonic signal visible on the receiver from both the crosstalk and observed reflection from an object.

Since the effective burst time has been increased, the minimum detectable distance has also increased. The transmitted burst detected by the receiver, due to crosstalk, has been lengthened so the total receiver wait time has also increased.

A5. Maximum Detectable Distance

The maximum detectable distance has been observed to be dependent on the transmitted amplitude of the ultrasonic wave as well as receiver resolution and sensitivity. Larger transmitted amplitudes as well as increased receiver sensitivity increase the maximum measureable distance.

Appendix B: PurePath Studio .h File

PurePath Studio can generate a header .h configuration script for the TLV320AIC3268. Once the desired coefficients for each processing block are determined, the .h file generated by PPS can be used to configure the AIC3268 externally with an MCU. To generate the .h file, *Generate Device Driver Interface* must be selected in the *Tools/Options/Build* windows in PPS. With this selected, a new .h file will be generated every time the process flow is saved to the save to directory.

This .h file can then be integrated in a MCU such as an MSP430 to write these configurations via I2C. For more information see http://www.ti.com/litv/pdf/slaa605a

Appendix C: External Configurations

Some AIC3268 configuration is not controlled by PPS and must be manually configured. Although they can be integrated in PPS SystemSettingsCode and run automatically, running these scripts externally first helps determine where errors may be occurring. At the end of design, any external scripts must be added into the host MCU I2C configuration code and run with the .h scrip in **Appendix B** as to fully configure that AIC3268 for standalone operation.

To run these scripts first download and run PPS to the EVM with the desired coefficients. Then, use the command line interface, found in the CS software, to run the scripts below afterwards.

The CS software for the AIC3204 can be used for its command line interface and is found here: http://www.ti.com/lit/zip/sloc128

Below is a script that will configure the microphone input as well as Class-D output amplifier correctly. These should require no modification by the user.

```
#######################################################################################
# ADC Input Channel Configuration --- IN2L / IN2R
#######################################################################################
w 30 00 01 # Select Page 1
w 30 08 00 # Set the input common mode to 0.9V
w 30 34 20 # Route IN2L and CM1 to LEFT ADCPGA with 20K input impedance
w 30 36 80
w 30 37 20 # Route IN2R and CM1 to RIGHT ADCPGA with 20K input impedance
w 30 39 80
```

```
###########################################################################################
# Output Channel Configuration
###########################################################################################
w 30 00 01 # Select Page 1
w 30 03 00 # Set PTM mode for Left DAC to PTM_P3 (default, writing here optional)
w 30 04 00 # Set PTM mode for Right DAC to PTM_P3 (default, writing here optional)
w 30 16 c3 # Enable DAC to LOL/R routing and power-up LOL/R
w 30 2E 0c # Route LOL to SPK @ -6dB
w 30 2F 0c # Route LOR to SPK_RIGHT_CH_IN @ -6dB
w 30 30 11 # Set SPK Gain @ 6dB, unmute SPK_RIGHT_CH_IN
w 30 2D 06 # Power-up SPK, route SPK_RIGHT_CH_IN to SPK
w 30 00 00 # Select Page 0
w 30 3f c0 # Power up the Left and Right DAC Channels
w 30 40 00 # Unmute the DAC digital volume control
```