INTRODUCTION

This application note describes the use of PIC® microcontroller Core Independent Peripherals (CIP) in ultrasonic range detection applications. The featured peripherals of Microchip’s PIC16F176X family enable the described example ultrasonic range detection circuit to calculate the distance traveled by the ultrasonic signal with minimum intervention from its Central Processing Unit (CPU).

Figure 1 shows the block diagram of an ultrasonic range detection system based on the PIC16F1769 microcontroller. The CIPs and on-chip peripherals used in the design are the following:

- Data Signal Modulator (DSM)
- Configurable Logic Cell (CLC)
- Hardware Limit Timer (HLT)
- Comparator (CMP)
- Operational Amplifier (OPA)
- Pulse-Width Modulation (PWM)
- Capture Compare PWM (CCP)
- Timers

The interconnection of these peripherals significantly reduces the component count needed to implement an ultrasonic range detection application. Refer to Appendix B: “Ultrasonic Proximity and Range Finder” for the detailed schematic diagram.
ULTRASONIC RANGE DETECTION OVERVIEW

Ultrasonic Transducer Background

Figure 2 shows the equivalent circuit of an ultrasonic transducer. The transducer is predominantly a capacitive load, but the inductance in the transducer forms a resonant circuit at around 40 kHz. The transmitter is tuned for maximum output while the receiver is tuned for maximum voltage output at an incoming 40 kHz signal. This has an attenuating filtering effect on all other frequencies and is useful in eliminating noise when amplifying the received signal.

![Figure 2: The Equivalent Circuit of an Ultrasonic Device](image)

One issue with ultrasonic transducers is that they will continue to oscillate or ring after the removal of the drive signal. This ringing is due to the resonant mechanical behavior of the transducer. The transducer is tuned to ring like a bell at its specified ultrasonic frequency when driven, and it takes a short period of time for the ring to damp out after the drive is removed. While the transmitter is ringing, the signal will couple through the PCB or travel through the air between the transmitter and receiver, and look like a received signal, as shown in Figure 3. Therefore, a delay is required before the receiver is activated to ensure that the ringing has damped out, and any signal, other than that of a reflected pulse, is received. The amount of time required for the ringing to damp out determines the minimum detectable distance of the receiver.

![Figure 3: Ultrasound Device](image)

- F0: Resonant Frequency
- C0: Parallel Capacitance
- R1: Series Resistance
- L1: Series Inductance
- C1: Series Capacitance
The whole circuit is controlled by the PIC16F1769 microcontroller, using its on-chip peripherals, as shown in Figure 4. The microcontroller generates a high-frequency drive, sets the duration of the drive and measures the time delay until the ultrasonic signal is received. The high-frequency drive produced by the microcontroller allows the ultrasonic transmitter to broadcast a 40 kHz pulse or several oscillations at that frequency. The PWM peripheral generates the specified frequency to drive the ultrasonic transmitter. This PWM is tied to the carrier input while a CLC1 peripheral is tied to the modulator input of the DSM peripheral. The CLC1 is configured to operate as an SR flip-flop with its set and reset inputs connected to Timer3 and Timer0, respectively. The SR flip-flop from the CLC1 connects and disconnects the PWM to the output pin through the DSM to generate a pulse of ultrasonic sound. The CLC1 output is also connected to the HLT peripheral and to the reset input of the CLC2 SR flip-flop. The HLT operates as a one-shot timer to create a time delay between the transmitter and receiver. The time delay ensures that any ringing has been dampened out. The HLT is tied to the CLC2 set input, to allow the receiver circuit to operate after the delay time.

When the signal is received and amplified by the OPA1, the peak detector will be compared to a receiver reference voltage. This receiver reference voltage is generated by the PRG connected to a unity gain OPA2. The peak detector and the unity gain OPA2 output are tied to the positive and negative inputs of the CMP1, respectively. The CMP1 output is tied to the CCP to generate a capture event, when the peak detector voltage is greater than the reference voltage. The captured time by the CCP will be the part of the round trip time traveled by the ultrasonic pulse.
Figure 5 depicts how this ultrasonic range detection system transmits and receives the ultrasonic pulse. The ultrasonic pulse train, which contains multiple pulses, is generated during the transmit period. This pulse train is reflected by an object of greater density than air, and some of the energy from the emitted pulse returns to the receiver. Before detecting the reflected pulse, a delay between transmit and receive is necessary to damp out the ringing produced during the transmission of ultrasonic pulse. Upon transmitting the ultrasonic pulse, the microcontroller starts its timer peripheral and captures the timer value when the reflected pulse is detected. When the signal has been detected, the round trip travel time is measured and converted to distance via the speed of sound.
PRODUCING 40 kHz WITH A PIC® MCU

Ultrasonic devices should be driven as close as possible to their specified frequency to maximize the output power. The 40 kHz drive signal of the ultrasonic devices can be easily created in a PIC microcontroller by dividing down its internal oscillator or by using Timer peripherals. Two I/O pins of a PIC MCU can be used to generate two differential 40 kHz signals that drive the ultrasonic transmitter while a Timer0 peripheral interrupt-on-overflow can be used to create the time base for the output signal. (See Appendix C: “Timer Flowchart Implementation for Ultrasonic Range Detection” for more details on how this is accomplished). However, this type of control will require a lot of firmware coding compared to the interconnection of CIPs that have no software overhead.

FIGURE 6: 40 kHz ULTRASONIC TRANSMITTER CONFIGURATION

Figure 6 shows the connection of peripherals to produce an ultrasonic pulse. The PWM output and CLC1 output are connected to DSM, whereas Timer0 and Timer3 are connected to the CLC1 inputs. The CLC1 sets the period of the ultrasonic pulse. This means that the duration of time that PWM pulses are output through the DSM output pin depends on the CLC1 state. This implies that the DSM effectively modulates the PWM pulses with CLC1 output. The CLC1 is periodically set by the Timer3 and reset by the Timer0 when the specified number of PWM pulses has been reached.

DEAD TIME BEFORE RECEIVING ULTRASONIC PULSE

After an ultrasonic signal is created and output from the ultrasonic transmitter, the next task is to create some dead time between transmitter and receiver. The dead time is necessary so that residual oscillations on the sending element do not generate a false signal on the receiving element. This is made possible by adding an HLT, configured as a one-shot timer, and by connecting and disconnecting the CMP1 positive input to ground (GND). The HLT one-shot timer period is the corresponding count of the minimum detectable distance traveled by the ultrasonic signal.

As shown in Figure 7, when the CLC1 is set, the CLC2 resets and the CMP1 positive input connects to GND. As the CLC1 transitions from high-to-low, the HLT is triggered. When the HLT one-shot times out, the CLC2 is set and the CMP1 positive input connects to the output pin. This output pin connects the CMP1 to the receiver circuit. By connecting and disconnecting the CMP1 positive input to GND, intermittent signal detection during the transmission of ultrasonic pulses is avoided.

FIGURE 7: DEAD TIME BETWEEN TRANSMITTER AND RECEIVER
RECEIVING AN ULTRASONIC PULSE

As the ultrasonic signal propagates in the air during the transmission, the intensity of the signal decreases with distance due to air absorption and beam spreading. Thus, the returning sound wave obtained by the receiver is significantly attenuated. To alleviate this problem, signal amplification is necessary to detect the returned signal. This amplification can be implemented using a single OPA inside the PIC microcontroller, configured to operate as a difference amplifier.

An example circuit for the difference amplifier is shown in Figure 8. This OPA circuit amplifies the voltage across the ultrasonic receiver connected between the two input pins. The common-mode noise at the difference amplifier output can be minimized by matching the input bias current through resistors R5 and R7 and resistors R4 and R9. The ultrasonic receiver acts like a tuned high Q filter, thus, the difference amplifier effectively amplifies the filtering effect of the receiver.

FIGURE 8: CIRCUIT FOR A DIFFERENCE AMPLIFIER

The output of the difference amplifier is tied to the peak detector to measure the maximum magnitude of a signal over a period of time. This peak detector is simply a diode D1 connected between the output of the difference amplifier (VAMP) and positive input of the CMP1 (VCMP+) with a parallel capacitor C5 and resistor R5, as shown in Figure 9. When the input signal on the peak detector is rising, D1 is forward-biased and C5 charges rapidly to the difference of VAMP and D1 voltage drop. When the input signal is falling, D1 becomes reverse-biased and C5 stored charge releases slowly through R6. The discharging of C5 continues until the VAMP becomes greater than the C5 charge. When VAMP is greater than the C5 voltage, D1 conducts again and the process is repeated.

FIGURE 9: PEAK DETECTOR
The output of the peak detector is compared to the receiver reference voltage using a comparator. The receiver reference voltage provides a falling ramp voltage so that the received signal reflected from long distances can be detected. This ramping voltage was produced by the PRG which is coupled to a unity gain OPA2 for its physical output. The PRG is configured to run as a falling ramp generator with its timing dependent on the CLC2 peripheral and its reference voltage input tied to the FVR peripheral. When the CLC2 output is reset, the PRG goes up to the FVR voltage, and when the CLC2 is set, the PRG outputs a falling ramp voltage. However, the falling ramp voltage produced by the PRG lasts for a very short period of time compared to the period needed for long distance detection. Due to this, external capacitors C9 and C10 are coupled to the OPA2 output. The combination value of C9 and C10 must be large enough to have a sufficient falling ramp voltage for better receiver sensitivity.

When the peak detector output reaches the receiver reference voltage, the CMP1 output will be set. This event triggers the capture mode of the CCP. The 16-bit CCP High Byte and Low Byte register pair (CCPRxH and CCPRxL) captures and stores the 16-bit value of the Timer1 Counter register pair (TMR1H and TMR1L), respectively. This captured value will be part of the round trip time in the form of counts traveled by the ultrasonic pulse.

CONVERTING COUNTS TO DISTANCE

After detecting the returning ultrasonic pulse, the value captured by the CCP and the HLT period will be converted to distance. This is made possible by dividing the CCP captured value by two, since the ultrasonic pulse travels back and forth, and then adding the HLT period. The resulting value is multiplied by the speed of sound and the inverse of the Timer clock input as described in Equation 1.

Example 1 shows the code snippet on how distance is calculated. The \texttt{SpeedOfSound\_at\_TimerInc} is a resulting constant from the product of the speed of sound and the time it takes for the Timer peripheral to increment. The variables used in the code are detailed in \textit{Appendix A: “Variables Used in the Firmware”}.

\textbf{EQUATION 1: \textit{DISTANCE FORMULA}}

\[
\text{Distance} = \left( \frac{\text{CCP captured value}}{2} + \text{HLT period} \right) \times \text{speed of sound} \times \frac{4}{F_{\text{OSC}}}
\]

\textbf{EXAMPLE 1: \textit{CODE EXAMPLE FOR DISTANCE COMPUTATION}}

\begin{verbatim}
Distance\_in\_Counts = (CCP1Capture - Timer1ReloadVal) \gg 1) + HLTPeriod;
Distance = (Distance\_in\_Counts \times SpeedOfSound\_at\_TimerInc);
\end{verbatim}
ULTRASONIC RANGING DETECTION FIRMWARE

Figure 10 shows the flowchart of the ultrasonic range detection firmware. During system start-up, the firmware initializes the peripherals and the connections between them. After the peripherals are initialized, the PRG ramp output is enabled and the interrupts are enabled. The firmware waits until a signal is detected or Timer3 times out. When a signal is detected, display_result is set and the distance traveled by the returned signal is calculated and displayed on the LCD. When Timer3 times out, if no signal has been detected, “No Detection” is displayed on the LCD.

Additionally, the firmware executes several Interrupt Service Routines (ISRs) that are automatically executed when certain peripheral criteria have been met.

1. Timer3 Interrupt – Executed periodically and dictates the maximum range detection of the ultrasonic pulse. When the Timer3 interrupt occurs, it reloads Timer0 and starts Timer4. Once Timer0 times out, CLC1 output will be set high, enabling Timer4 HLT. If no_detection flag is set during the Timer3 interrupt, “No Detection” will be displayed on the LCD.

2. CLC2 Interrupt – The rising interrupt of the CLC2 occurs when Timer4 HLT one-shot times out. CLC2 Interrupt will connect the CMP1 positive input to C1IN1+ output pin to enable the detection of the received signal. The interrupt will also reload Timer1, which keeps count of the received signal’s travel time.

3. CCP Interrupt – When the ultrasonic pulse is received, the CCP1 interrupts. When this happens, the CMP1 positive input is connected to GND to disable the detection. The display_result variable is set so that the main loop will calculate and display the measured distance on the LCD.
Notice that after the initialization, the calculation of the distance traveled by the returned ultrasonic pulse is dependent on the ISRs. This happens due to the CIPs, which, integrated to control the operation of the ultrasonic range detection, perform their tasks independently. As a result, the complexity of the firmware is reduced and the core is only required during the ISRs. All peripherals used in the firmware are configured and initialized using the MPLAB® Code Configurator (MCC). For the complete source code, refer to the code appended to the electronic version of this application note.

MAXIMUM DETECTABLE DISTANCE

There are two ways of increasing the maximum detectable distance in this application: increased transmission power and increased receiver sensitivity. Increasing the transmission power is made possible by connecting the DSM output to a Logic-Controlled Load driver. This logic-controlled load driver, which is one application use of a MOSFET driver, boosts the current and voltage supplied to the transmitter transducer. As a result, the ultrasonic pulse can travel longer distances and still be detectable by the receiver.

Another method of increasing the maximum detectable distance is by increasing the receiver sensitivity. In this application, the sensitivity of the receiver can be increased by changing the gain to a much higher value. Also, carefully controlling the offset value of the difference amplifier ensures that the smallest return pulse is positively detected.

LAYOUT CONSIDERATIONS

If a separate transmitter and receiver are used, they should both be aligned in the same direction. The transmitted signal and any subsequent ringing will leak through the PCB to the receiver circuitry. Placing more space or a cut-out between the devices on the board will help to minimize this leakage. Ultrasonic transducers are often mounted using rubber or silicon to limit the amount of leaked ultrasonic signal to/from the surrounding material.

PERFORMANCE

Table 1 summarizes the performance of the ultrasonic range detection. It can be observed that as the input voltage of the range finder increases, the measured range also increases.

<table>
<thead>
<tr>
<th>Input Voltage (V)</th>
<th>Maximum Range Measured (in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 V</td>
<td>207</td>
</tr>
<tr>
<td>12 V</td>
<td>246</td>
</tr>
<tr>
<td>15 V</td>
<td>260</td>
</tr>
<tr>
<td>18 V</td>
<td>273</td>
</tr>
</tbody>
</table>

CONCLUSION

This application note describes a PIC microcontroller-based solution for measuring the distance to an object ultrasonically with fewer external components. By utilizing the flexibility of the PIC16F176X family microcontroller, with its core independent peripherals, the traveled distance by the ultrasonic pulse is acquired with less firmware overhead.
APPENDIX A: VARIABLES USED IN THE FIRMWARE

The variables used in the firmware are defined in the define.h file. Below is the summary of these variables.

**TABLE A-1: FIRMWARE VARIABLES**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>display_result</td>
<td>This variable toggles for the distance result display</td>
<td>0 or 1</td>
</tr>
<tr>
<td>SpeedOfSound_at_TimerInc</td>
<td>Speed of sound multiplied by the inverse of clock input (i.e., 4/FOSC)</td>
<td>0.0135612 (in inches)</td>
</tr>
<tr>
<td>HLTPeriod</td>
<td>Set HLT period value</td>
<td>0xD8</td>
</tr>
<tr>
<td>no_detection</td>
<td>This variable toggles when no receive signal is detected after a certain period of time, determined by Timer3 period</td>
<td>0 or 1</td>
</tr>
</tbody>
</table>
FIGURE B-1: SCHEMATIC

APPENDIX B: ULTRASONIC PROXIMITY AND RANGE FINDER
APPENDIX C: TIMER FLOWCHART IMPLEMENTATION FOR ULTRASONIC RANGE DETECTION

FIGURE C-1: ULTRASONIC FLOWCHART FOR TIMER IMPLEMENTATION

- Power On
  - US_Init()
  - LCD_Init()
  - Math_Init()
- Enable Global Interrupts
  - Clear TMR1H:L
  - Turn on TMR1
- SendPulse()
  - SendDelay()
  - DetectReturnPulse()
- Increase number of output pulses to a max. of 16
  - TMR1 Overflowed 8 times?
    - NO
      - CountDistance()
      - AverageDistance()
      - WriteOutToLCD()
    - YES
      - Minimum distance detected 'numPulses' set to 0x01
- TMR1 Overflowed 8 times?
  - NO
    - Turn off Timer1
      - Store Timer1 count
      - Turnoff Comparator
      - Return
    - YES
      - Increment Overflow Count
      - Clear TMR1IF
      - Preload Timer0 with 0xF1
      - Initialize USdrive pin high one low
      - Clear Timer0 Interrupt Flag(T0IF)
      - Set Timer0 Interrupt Enable (T0IE)
      - Load TMR0 with 0xF3
      - Toggle USdrive pins
      - Decrement 'i'
        - 'i' = 0?
          - NO
            - Set output pins to ground
            - Return
          - YES
            - Clear T0IE
        - Return From Interrupt
  - YES
    - Copy 'numPulses' to variable 'i'
    - Double number of output pulses
    - Preload Timer0 with 0xF1
    - Initialize USdrive pin high one low
    - Clear Timer0 Interrupt Flag(T0IF)
    - Set Timer0 Interrupt Enable (T0IE)
    - Load TMR0 with 0xF3
    - Toggle USdrive pins
    - Decrement 'i'
      - 'i' = 0?
        - NO
          - Set output pins to ground
          - Return
        - YES
          - Clear T0IE
      - Return From Interrupt
Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip’s Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KEELOQ, KEELOQ logo, Kleer, LANCheck, LINK MD, maxStylus, maxTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNiC, SST, SST logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.


SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Silicon Storage Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.


ISBN: 978-1-5224-1880-1

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company’s quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KeeLoq® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM CERTIFIED BY DNV

ISO/TS 16949