To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

H8/300L

Ultrasonic Range Finder (Ultrange)

Introduction

This application note describes the implementation of an ultrasonic range finder using H8/38024 SLP MCU. A 40kHz square wave is generated by the MCU and transmitted through an ultrasonic sensor. The reflected ultrasound is received by another ultrasonic sensor. Computation of distance is then performed by the MCU. The effective range is from 6cm to 200cm.

Target Device

H8/38024F

Contents

1. Theory

1.1 Overview

The H8/38024F microcontroller is used as the target in this application note. The software for ultrasonic range finder is written in C for easy portability.

Ultrasounds refer to sound above the frequencies of audible sound and nominally include anything over 20 kHz. Frequencies used for medical diagnostic and imaging extend to 10 MHz and beyond. Higher frequencies have shorter wavelengths, which make them 'bounce' (reflect) from objects more readily. Unfortunately, extremely high frequencies are difficult to generate and measure. Detection and measurement of ultrasonic sound is mainly through the use of piezoelectric receivers.

Ultrasound is commonly applied in burglar alarms, motion detectors and range meters in cars. Other applications include medical diagnostic (imaging of human body), cleaning (removal of grease and dirt), flow meter (making use of Doppler effect), non-destructive testing (for detecting material imperfections), soldering etc.

1.2 Software Implementation

Distance is computed by measuring the time taken for the ultrasound to echo back to the ultrasonic sensor. Ideally, the object should have a large surface area and does not absorb the ultrasound.

The 38024F CPU Board is used in this application note. Figure 1 shows the working principles of the ultrasonic range finder. TMOFH (Pin 63) is used to transmit 40kHz ultrasound for 0.5msec and IRQ0 (Pin72) is used to detect the reflected wave. After transmission, Timer C is started to keep track of the number of counts in Timer Counter C (TCC) so as to compute the distance of the object.

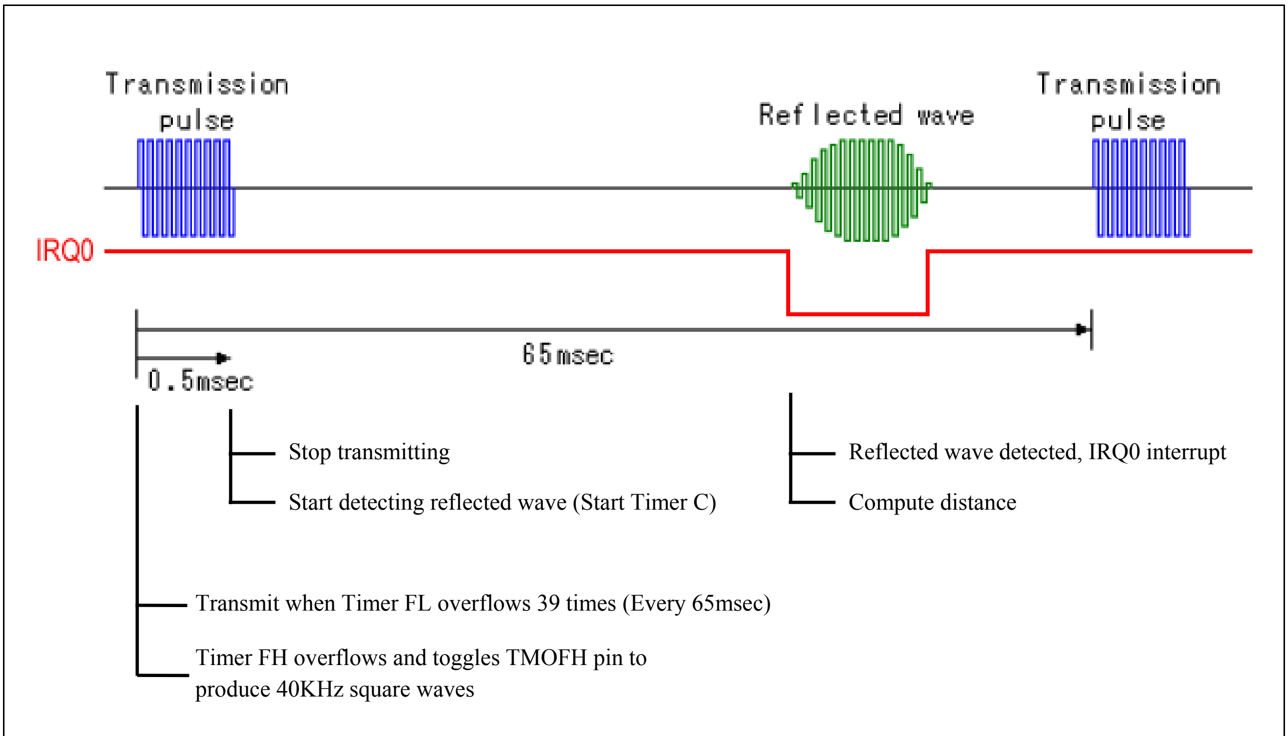


Figure 1. Working Principles of the Range Finder

1.2.1 Transmission of Ultrasound

Timer F is a 16-bit free running counter with a built-in output compare function. It can also be used as two separate 8-bit timers FH and FL.

In this AN, Timer F is used as two separate 8-bit timers. Timer FL is initialized to generate an interrupt whereas Timer FH toggles the output level of TMOFH when compare match occurs.

CKSH2/ CKSL2	CKSH1/ CKSL1	CKSH0/ CKSL0	Description
1	0	0	Internal clock: counting on ø/32
1	0	1	Internal clock: counting on ø/16
1	1	0	Internal clock: counting on ø/4
1	1	1	Internal clock: counting on øw/4

Table 1 Clock Selection for Timer F

For Timer FL, internal clock of ø/32 is selected. Output Compare Register FL (OCRFL) is loaded with H'FF.

Therefore, Timer FL generates an interrupt every 1.67msec as calculated below:

Ø = Crystal frequency / 2

$$\text{Timer FL Internal Clock frequency} = \frac{\text{Crystal frequency}}{2 \times 32} = \frac{9.8304\text{ MHz}}{64} = 153.6\text{kHz}$$

$$\text{Interrupt Period} = \frac{1}{153.6\text{kHz}} \times 256 = 1.67\text{msec}$$

To start transmitting ultrasound every 65msec, Timer FL needs to be interrupted for 39 times (65msec / 1.67msec = 39) before starting to transmit.

Bit 2 TMOFH	Description
0	Functions as P32 I/O pin (Initial value)
1	Functions as TMOFH output pin

Table 2 Function Selection of Port Mode Register 3 Bit 2

For Timer FH to generate a 40kHz signal, the output level of TMOFH is set to toggle when counter FH (TCFH) value matches that of the value in Output Compare Register FH (OCRFH). The Output Compare Register FH value is calculated as follows:

Timer FH, internal clock of ø/4 is selected.

$$\text{Timer FH Internal Clock period} = \frac{1}{\frac{\text{Crystal frequency}}{2 \times 4}} = \frac{8}{9.8304\text{MHz}} = 0.814\mu\text{sec}$$

For a signal of 40kHz, TMOFH needs to toggle very 12.5µsec: (1/40kHz)/2.

Output Compare Register FH, OCRFH = $\frac{12.5\mu sec}{0.814\mu sec}$ = 15.36 ≈ 15

Hence, OCRFH is loaded with H'0F.

A software delay is used to send the ultrasound for 0.5msec before switching Pin 63 to I/O port P32 to stop transmitting. Table 2 shows the setting for Port Mode Register 3 to select pin to function as an I/O pin or TMOFH output pin.

1.2.2 Initialization of Timer C

After transmitting the ultrasound, Timer C is turned on to count the time to receive back the reflected ultrasound. Timer C is set as an auto reload, up-counter with internal clock selected as ø/64. Table 3 shows the setting for Timer Mode Register C. The required settings are in bold.

Bit 7 TMC7	Description
0	Interval timer function selected (Initial value)
1	**Auto-reload function selected**

Bit 6 TMC6	Bit 5 TMC5	Description
0	**0**	**TCC is an up-counter**
0	1	TCC is a down-counter
1	* Don't care	Hardware control by UD pin input UD pin input high: Down-counter UD pin input low: Up-counter

Bit 2 TMC2	Bit 1 TMC1	Bit 0 TMC0	Description
0	0	0	Internal clock: ø/8192
0	0	1	Internal clock: ø/2048
0	1	0	Internal clock: ø/512
0	**1**	**1**	**Internal clock: ø/64**
1	0	0	Internal clock: ø/16
1	0	1	Internal clock: counting on ø/4
1	1	0	Internal clock: counting on øw/4
1	1	1	External event (TMIC): rising or falling

Table 3 Settings for Timer Mode Register C

Timer Load Register (TLC) is then loaded with H'00 to start counting from 0.
Timer C interrupt is enabled, IENTC = 1 in Interrupt Enable Register 2 (IENR2). If the count value in Timer Counter C (TCC) reaches H'FF, the next clock input causes timer C to overflow, generating an interrupt. In the Timer C overflow interrupt subroutine, OVERFLOW_COUNT is incremented to keep track of the number of overflows.

When the reflected ultrasound is received, IRQ0 voltage level drops low and generates IRQ0 interrupt. Timer C counter is stopped by setting '1' to TMC2 ~ TMC0 as no external clock is present to increment the counter. Then the value of TCC is read and used to compute the distance.

1.2.3 Computation of Distance

Having chosen the internal clock for Timer as ø/64, distance is calculated as follows:

$$\text{For Timer C, 1 count} = \frac{1}{\frac{\text{Crystal frequency}}{2 \times 64}} = \frac{128}{9.8304\text{MHz}} = 13.02\mu\text{sec}$$

Speed of sound = 343 m/sec = 34300 cm/sec

Therefore, time taken to travel 1cm = 1sec/34300cm = 29.15 μsec

By tracking the number of counts and overflows in Timer Counter C (TCC), the distance of the object can be calculated.

For example, a count of 55 and 1 overflow,

Total number of counts = (1x 256) + 55 = 311

Total time taken to receive reflected ultrasound (in μsec) = 311 x 13 = 4043

$$\text{Distance between sensors and object} = \frac{4043 / 2}{29} = 69.7 \approx 70 \text{ cm}$$

Divide by 2 due to reflection of ultrasound (distance traveled is twice the distance to the object)

1.3 Hardware Implementation

The schematic of the ultrasonic range finder is given in Section 4. Details of the ultrasonic transmitter and receiver circuits are discussed in the following sections.

1.3.1 Transmitter Circuit

The transmitter circuit is made up of several inverters and two transistors. The first inverter outputs the negative part of the ultrasonic wave. The transistors are to drive the CMOS inverters. The two inverters are connected in parallel to increase electric power transmission. The phase of the sensor is shifted by 180° between the positive and negative terminals of the sensor. The voltage applied on to the transmitter is twice the from the single inverter input (have a positive and negative peak to peak value).

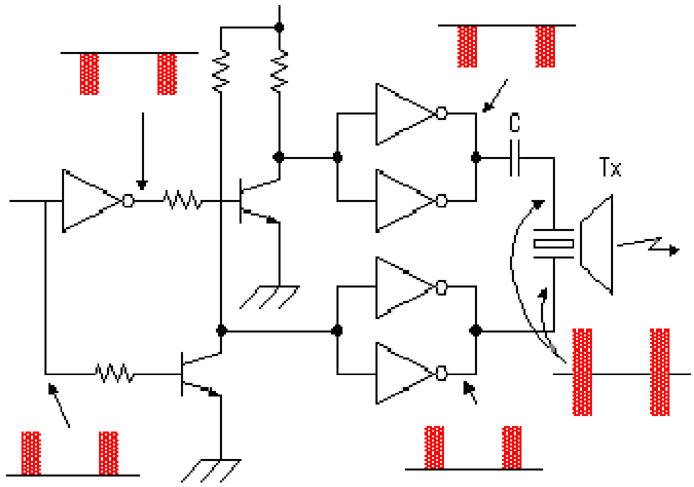


Figure 3. Transmitter Circuit

1.3.2 Receiver Circuit

The receiver circuit consists of two main parts, the signal amplification and detection circuitry.

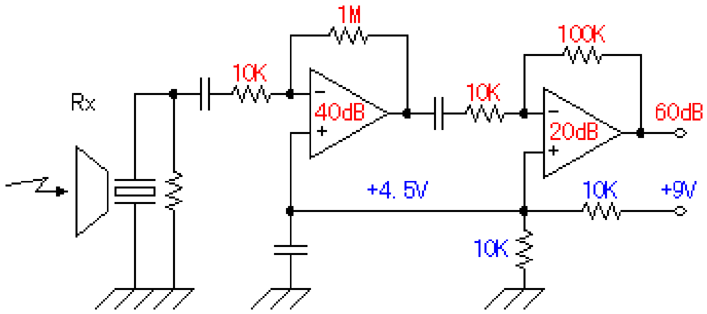


Figure 4. Signal Amplification Circuit

Upon receiving an ultrasonic signal through a reception sensor, the signal is amplified by 1000 times. The first stage will amplify the original signal by 100 times (40 dB) and the gain of second stage is 10 (20 dB).

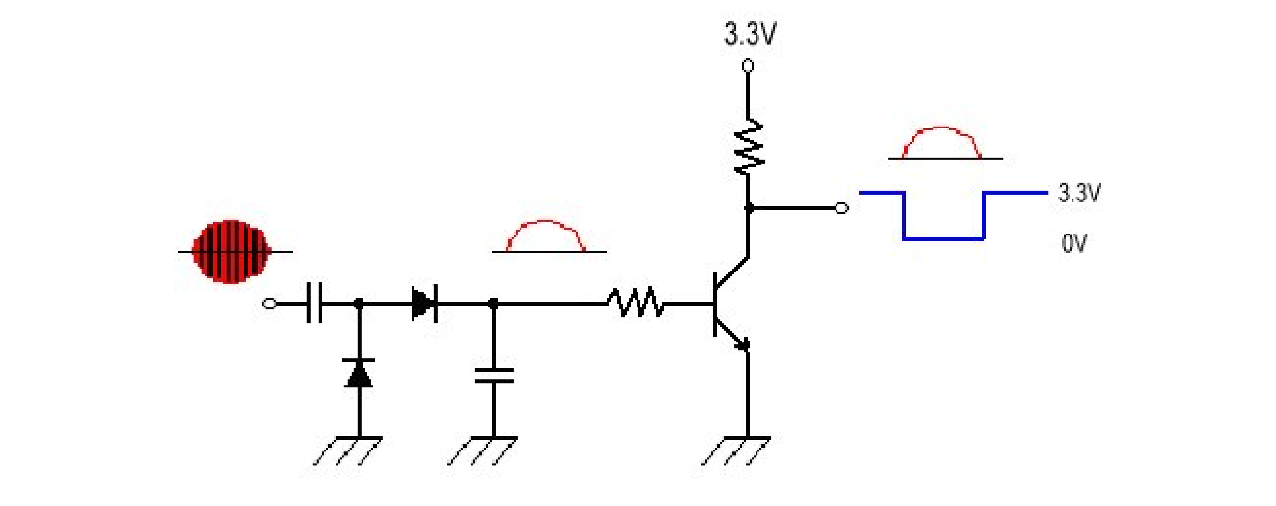


Figure 5. Signal Detection Circuit

After the amplification circuit, the signal will go through a detection circuit consisting of a half wave rectification circuit. This circuit is implemented by two 1CV5 diodes. The rectified signal goes to the transistor. When there is no signal, output is 3.3V (HIGH). If a signal is present, it will cause the output voltage to drop to 0V (LOW).

The output is fed into IRQ0 pin of the H8/38024 to generate an interrupt when a falling edge is detected.

1.3.3 Power Supply

Three voltage supplies are required:

Range Finder Board

- 9V input voltage – For amplifier LM833
- 3.3V – For inverter 74LS04 and transistors BC547

38024 CPU Board

- 5V input voltage – CPU Board
- 3.3V – For MCU

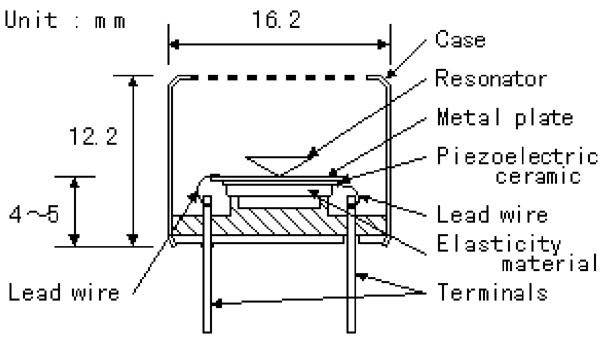
User has to provide 9V input voltage to the range finder board and 5V input voltage to the CPU Board.

1.3.4 Ultrasonic Sensor

Ultrasonic transmitter (T40-16) and receiver (R40-16) from Nippon Ceramic company is used in this Application Note. *T* indicates transmitter, *R* for the receiver and *40* refers to the resonant frequency of the ultrasonic.(40kHz).

The brief specification of the ultrasonic sensor is shown below.

Item		Spec
Frequency(kHz)		40
Sound pressure level (dB)		115 <
Sensitivity (dB)		-64 <
Size (mm)	Diameter	16.2
	Height	12.2
	Interval	10.0



2. Operation

The 38024F CPU Board is connected to the Ultrasonic Range Finder circuit as follows:

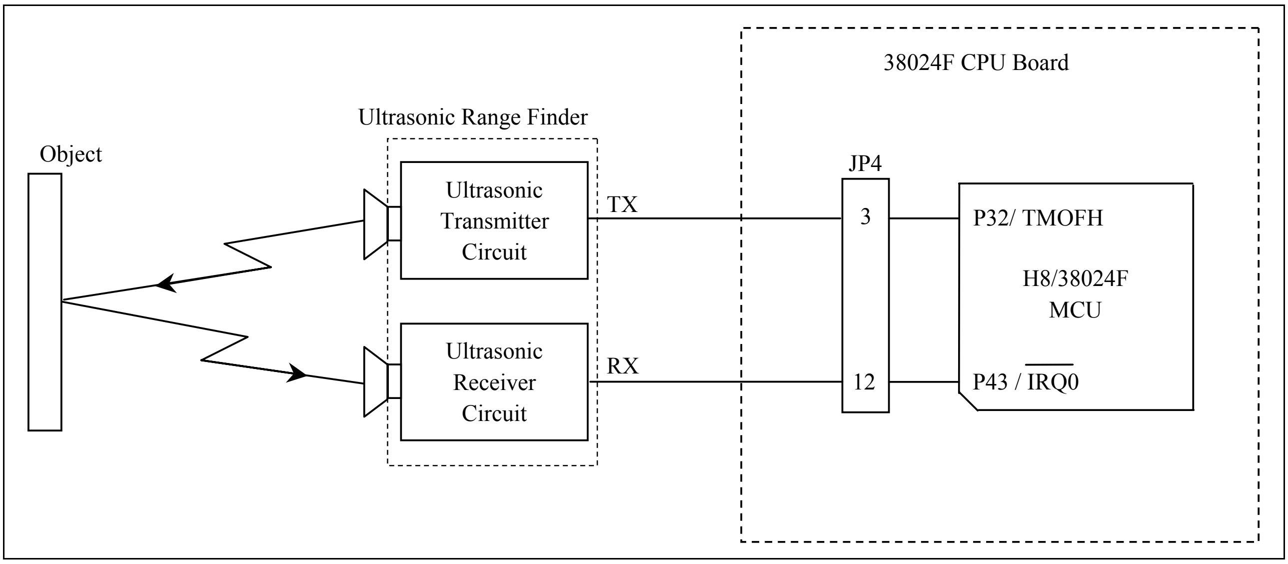


Figure 6 Microcontroller Setup with Ultrasonic Range Finder

TMOFH, which outputs the ultrasound, has to be connected to the TX pin of the Ultrasonic Transmitter Circuit. The detected signal is connected to IRQ0. Hence, connect pins 3 and 12 of JP4 on the 38024F CPU Board to the TX and RX pins of the Ultrasonic Range Finder respectively.

2.1 Hyper Terminal Setting

After completing the hardware setup, the user has to configure the HyperTerminal window to display the distance detected by the MCU. The COM port settings have to be made in accordance with the UART protocol and Baud Rate used in the program as shown in Figure 7.

From the start menu button, go to *Programs > Accessories > Communications > HyperTerminal*. Select *Properties* in the *File* menu of HyperTerminal window and click on *Configure…* to change the Port Settings.

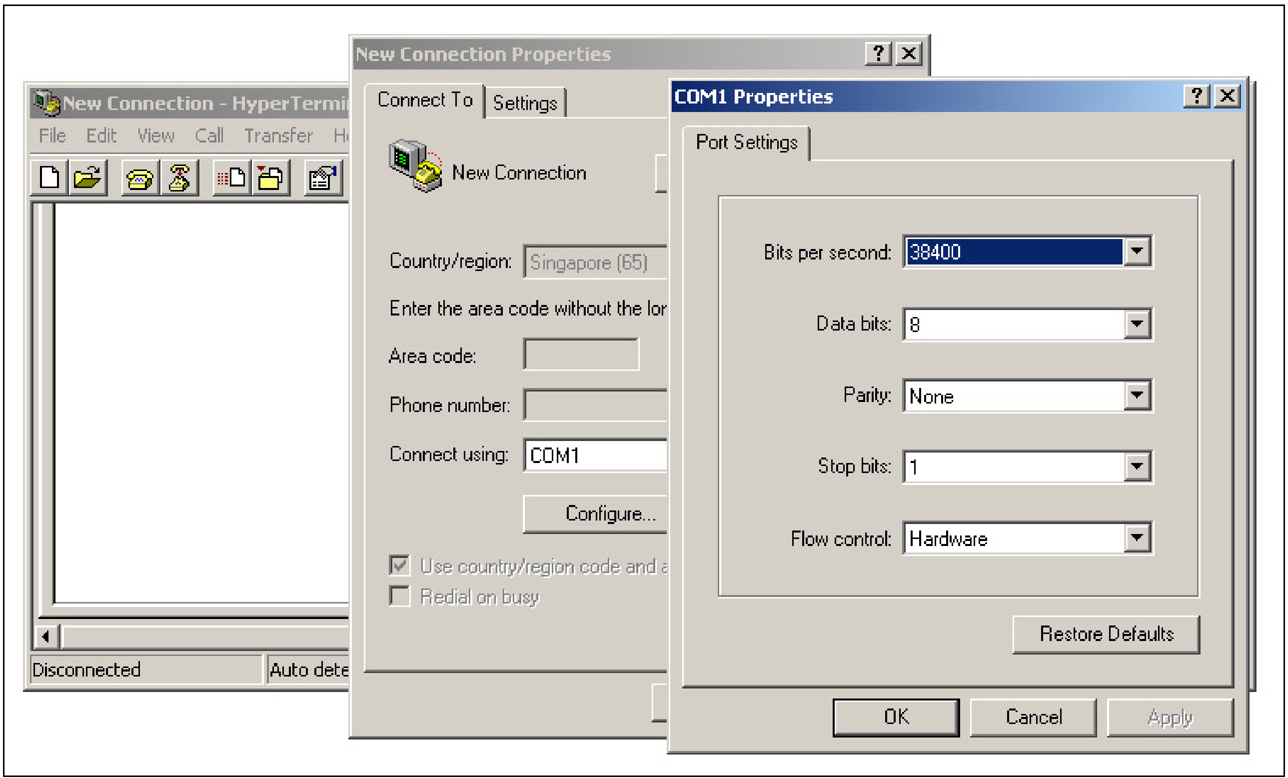


Figure 7 PC HyperTerminal settings

2.2 Results

First, flash the program into the MCU using FD. Then run the program by pressing reset button in user mode. Observe that LED D1 on the CPU Board is blinking continuously, indicating that the ultrasound is being transmitted.

By placing a large object that does not absorb the ultrasound in front of the sensors, the user will be able to see the detected distance in the HyperTerminal window as shown in Figure 9. Each time the object is detected (IRQ0 interrupt generated); a dot is displayed on the HyperTerminal window. After detecting 5 similar readings, the readings are averaged and the distance is calculated and displayed.

This ultrasonic range finder can only detect the object between 6cm to 200cm.

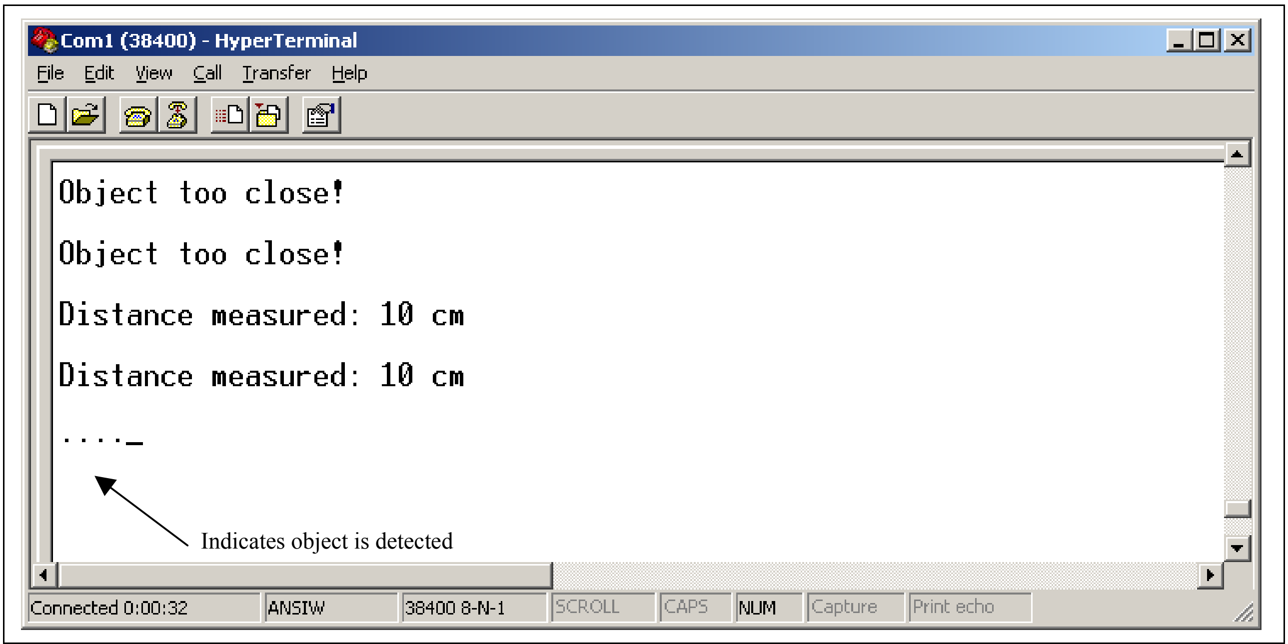


Figure 8 Results Displayed on PC HyperTerminal

2.3 Limitations

2.3.1 Distance Between Sensors

The main consideration for designing an ultrasonic range finder would be the positioning of the ultrasonic sensors. If the receiving ultrasonic sensor is placed far away from transmitting ultrasonic sensor, it would not be able to detect objects which are very close to it. This is illustrated below.

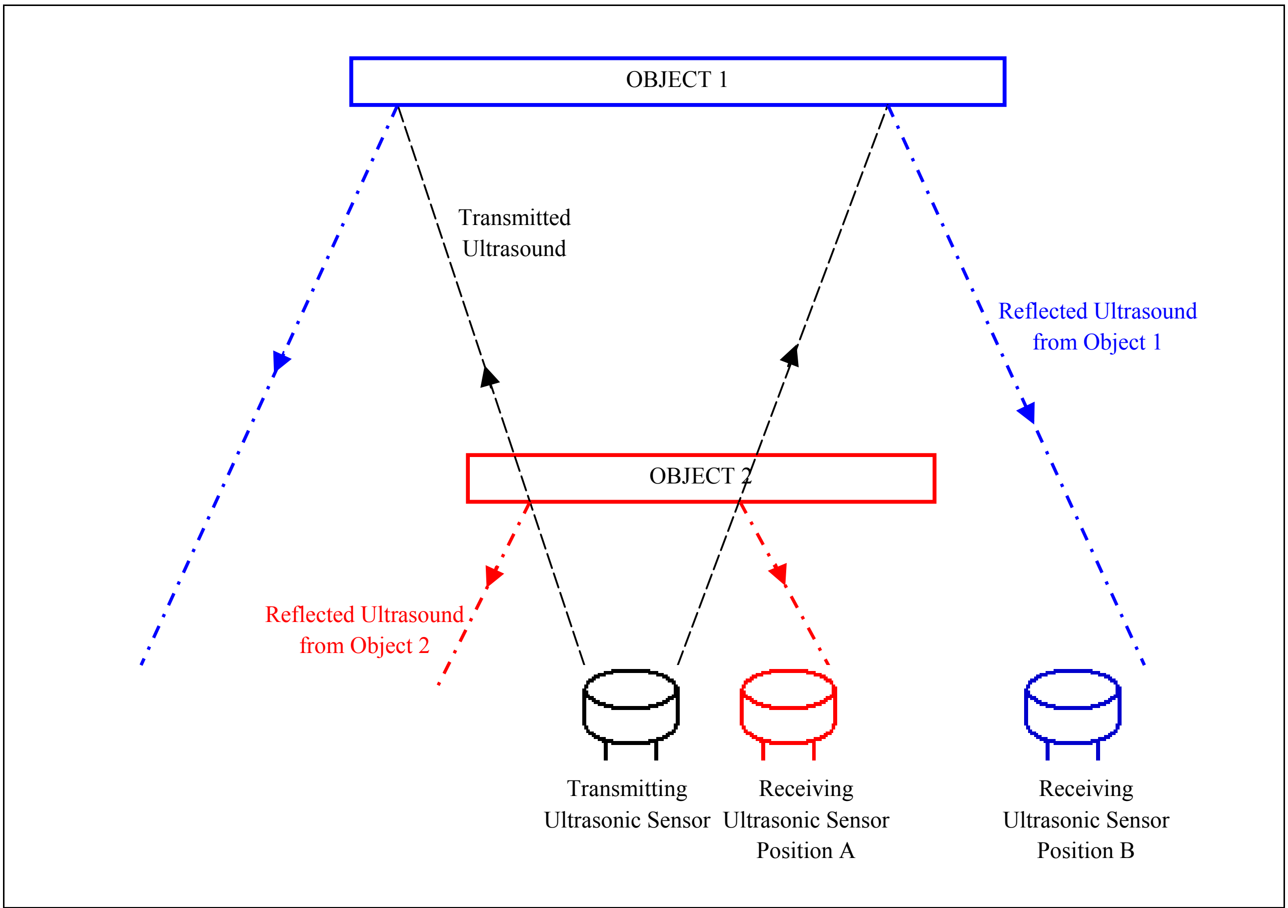


Figure 9 Illustration of Difference Between Distances of Sensors

For Object 1, which is further away, placing the receiving ultrasonic sensor at either Position A or B would not be a problem as the reflected ultrasound would reach both sensors.

However for Object 2, if the ultrasonic sensor is placed at Position B, the sensor would not detect the reflected ultrasound as it is too far apart.

For applications (e.g. micromouse), which require the measurement of short distances, the sensors would have to be placed close to each other or turned slightly to face each other.

In this Application Note, the sensors are placed 3.5cm apart.

2.3.2 Actual Distance Measured

The ultrasonic range finder measures distance by dividing the time when the echoed back ultrasound is received by half. However, the actual distance is the distance perpendicular to the ultrasonic sensors. This error would be more significant for near objects and negligible for objects that are far away, as shown in Figure 10.

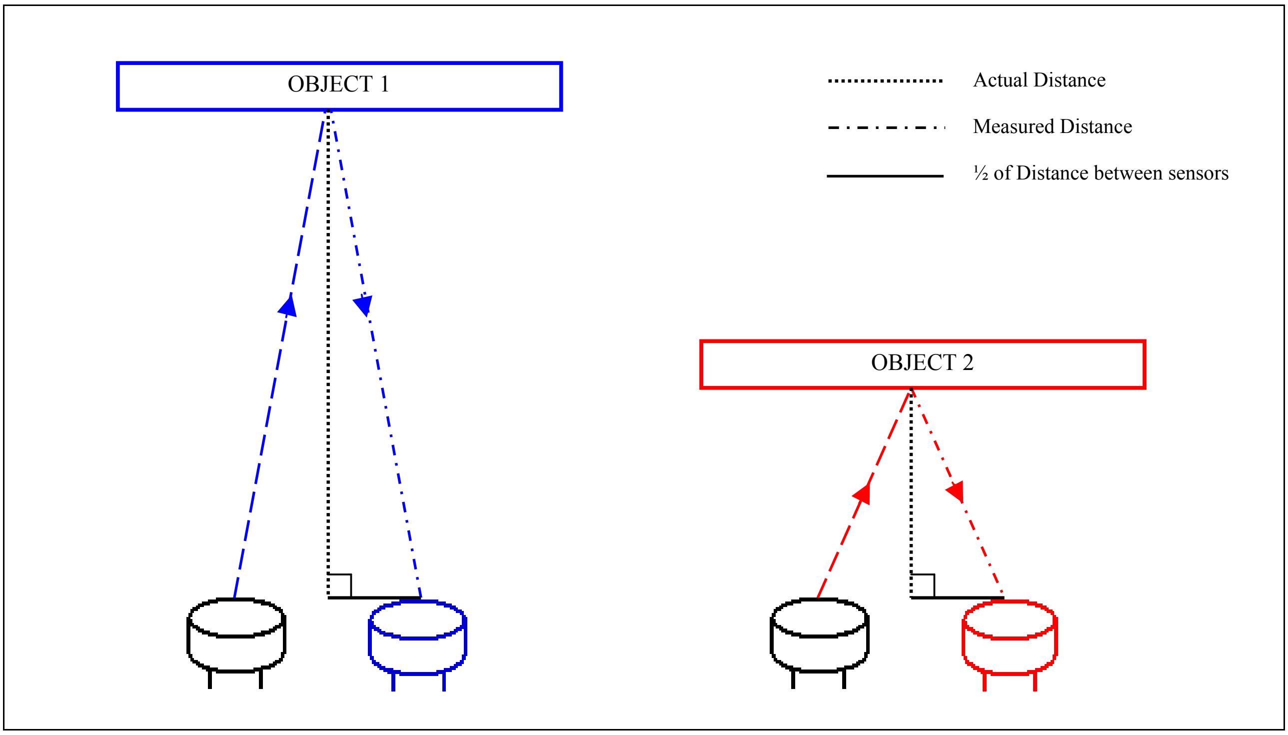


Figure 10 Illustration of Actual Distance Measured

Users may use the following formula to compute the distance to correct the error:

$$\text{Actual Distance} = \sqrt{(\text{Measured Distance})^2 - (\tfrac{1}{2}\text{ of Distance between sensors})^2}$$

2.3.3 Dead Zone

Ultrasonic sensors have a Dead Zone in which they cannot detect the target. This is the distance between the sensing face and the minimum sensing range. The Nippon Ceramic Company Ultrasonic sensor's Dead Zone is experimentally determined to be about 1cm.

2.3.4 Detectable Range

The minimum detectable range is due to the Dead Zone (refer to section 2.3.3) and limitations of the MCU as well as the response and layout of the circuitry. As distance is computed from the value of the Timer C counter, accuracy would depend on the time Timer C start and stop counting. Another limitation of the MCU would be the interrupt latency.

The minimum and maximum detectable range is experimentally determined to be 6cm and 200cm respectively.

Hence an offset value (DISTANCE + 5) is required in the program. Users should experimentally determine the minimum detectable distance of their circuits and add the offset value accordingly.

The maximum detectable range is also determined by the input voltage to the LM833 operational amplifier. The amplitude of the amplified output signal decreases with the input voltage, hence the maximum detectable range also decreases.

The minimum input voltage to the LM833 operational amplifier is +5V for this range finder circuit. This is due to the voltage drop across the two diodes. If voltage is below +5V, there would not be enough voltage to turn on the transistorQ3.

By reducing the input voltage to the LM833 operational amplifier to +5V, the maximum range drops to 150cm.

3. Codes

The following attached code for this Application Note is generated using HEW project generator targeting at H8/38024 micon. The toolchain used is the free H8 Tiny/SLP toolchain (version 1.0.0) for HEW Version 2.2 (Release 15).

Flowcharts are included to illustrate the main functionality and to give a better understanding for the user.

```
/***********************************************************************/
/*                                                                     */
/*   FILE        :subfunctions.h                                       */
/*   DATE        :Tue, Feb 17, 2004                                    */
/*   DESCRIPTION :Subfunctions and defined constants used              */
/*   CPU TYPE    :H8/38024F                                            */
/*                                                                     */
/***********************************************************************/

void initialize (void);
void char_put(char);
void PutStr(char *);
void display_decimal(unsigned int);

#define  COUNT_PERIOD  13    //(micro secconds) calculate as (2*64) / XTAL freq
                             //(internal clock source selected as phi/64)
#define  CONV_DIST    29     //Speed of sound = 343m/sec,
                             //so for 1cm, time taken is 29.15 usec
```

```
/**********************************************************************/
/*                                                                    */
/*  FILE        :Ultrasonic_sensor.c                                  */
/*  DATE        :Tue, Feb 17, 2004                                    */
/*  DESCRIPTION :Main Program                                         */
/*  CPU TYPE    :H8/38024F                                            */
/*                                                                    */
/*  This file is generated by Hitachi Project Generator (Ver.2.1).    */
/*                                                                    */
/**********************************************************************/
#include "iodefine.h"
#include "subfunctions.h"
#include <machine.h>
#include <_h_c_lib.h>

unsigned int TIME, DISTANCE, COUNT, PREVIOUS_COUNT;
unsigned int INPUT_CAPTURE, OVERFLOW_COUNT, MATCH, delay;
unsigned long ADD_COUNT;
signed int DIFFERENCE;

void main(void)
{
   delay = PREVIOUS_COUNT = MATCH = ADD_COUNT = 0;
   initialize();

   while (1);
}

//--------------------------------------------------------------------//
/* init_sci() : Initialize PORT 9, SCI3, IRQ0, Timer F               */
//--------------------------------------------------------------------//
void initialize (void)
{
// INITIALIZE Port 9
   P_IO.PMR9.BYTE = 0x00;          // Port 9 as output port
   P_IO.PDR9.BYTE = 0xFF;          // Off LEDs

// INITIALIZE SCI3
   //Serial Control Register
   //CKE1 = CKE0 = '0': SCK32 functions as I/O port
   P_SCI3.SCR3.BYTE &= 0x00; //clear TE & RE

   //Serial Mode Register
   //SMR : |COM|CHR|PE|PM|STOP|MP|CKS1|CKS0| : |0|0|0|0|0|0|0|0|
   //COM : Communication Mode  : 0 : asynchronous mode
   //CHR : Character Length    : 0 : character length = 8 bits
   //PE  : Parity Enable       : 0 : parity bit addition and checking disabled
   //PM  : Parity Mode         : 0 : even parity (no effect since parity is
   //                                 already disabled)
   //STOP: Stop Bit Length     : 0 : 1 stop bit
   //MP  : Multiprocessor Mode : 0 : multiprocessor communication function
   //                                 disabled
   //|CKS1|CKS0| : Clock Select: |0|0| : clock source for baud rate generator
   P_SCI3.SMR.BYTE = 0x00;
```

```
    //Bit Rate Register
    //For clk = 10MHz, bit rate = 38400 bps, n = 0, N = 3
    P_SCI3.BRR = 3;

    //minimum of 1-bit delay = 417ns
    nop();
    nop();
    nop();

    //SPCR : |---|---|SPC32|---|SCINV3|SCINV2|---|---| : |1|1|1|0|0|0|0|0|
    //SPC32 = 1 : P42 functions as TXD32 output pin
    //need to set TE bit in SCR3 after setting this bit to 1
    //SCINV3 = 0 : TXD32 output data is not inverted
    //SCINV2= 0 : RXD32 input data is not inverted
    //Bits 7 and 6 are reserved and always read as 1
    //Bits 4, 1 and 0 are reserved and only 0 can be written to these bits
    P_SCI3.SPCR.BYTE = 0xE0;

    P_SCI3.SCR3.BYTE |= 0x30; //Set TE & RE

// INITIALIZE IRQ0 INTERRUPT
    P_IO.PMR2.BIT.IRQ0 = 1;          // I/O pin used as input capture
    P_SYSCR.IEGR.BIT.IEG0 = 0;       // Interrupt at falling edge of IRQ0

    P_SYSCR.IENR1.BIT.IEN0 = 1;      // Enable IRQ0 interrupt

// INITIALIZE TIMER F
    //Timer Control Register F
    //TOLH = '1':  Initial output for TMOFH is high
    //CKSH2 = '1', CKSH1 = '1', CKSH0 = '0': 8-bit mode, phi/4 i.e.,
    //(9.8304MHz/2/4 = 1.2288MHz)
    //TOLL = '0': Initial output for TMOFL is low
    //CKSL2 = '1', CKSL1 = '0', CKSL0 = '0': 8-bit mode, phi/32 i.e.,
    //(9.8304MHz/2/32 = 153.6kHz)
    P_TMRF.TCRF.BYTE = 0xE4;

    //Timer Control/Status Register F
    //CCLRH = '1': in 8-bit mode, TCFH clearing by compare match is enabled
    //CCLRL = '1': in 8-bit mode, TCFL clearing by compare match is enabled
    P_TMRF.TCSRF.BYTE = 0x11;

    //Output Compare Register FL
    //OCRF = FF
    P_TMRF.OCRF.BYTE.L = 0xFF;
    //Output Compare Register FH
    //OCRF = 0F    (1.2288MHz/40kHz=30 , 30/2=15 0xF) .
    P_TMRF.OCRF.BYTE.H = 0x0F;

    P_SYSCR.IENR2.BIT.IENTFL = 1;// Enable Timer F L interrupt
}
//__________________________________________________________________________
```

```
//--------------------------------------------------------------------------//
/* char_put() : Transmits a character to the PC for debugging purposes.  */
//--------------------------------------------------------------------------//

void char_put(char OutputChar)              //Serial Port
{

   //TDRE : transmit data register empty
   while ((P_SCI3.SSR.BIT.TDRE) == 0);        //Wait for TDRE = 1

   P_SCI3.TDR = OutputChar;
   //TEND : transmit end
   while ((P_SCI3.SSR.BIT.TEND) == 0);        //Wait for TEND = 1

   P_SCI3.SSR.BIT.TEND = 0;
}
//__________________________________________________________________________

//--------------------------------------------------------------------------//
/* PutStr() : Transmits a string of characters to the PC                  */
//--------------------------------------------------------------------------//
void PutStr(char *str)
{
    while (*str != 0)  char_put(*str++);
}
//__________________________________________________________________________

//--------------------------------------------------------------------------//
/* display_decimal() : Transmit through SCI3 an interger value in decimal*/
//--------------------------------------------------------------------------//
void display_decimal(unsigned int display_data)
{
   unsigned char  first_digit, second_digit, third_digit, fourth_digit,
fifth_digit;

   first_digit  = (unsigned char)(display_data / 10000);
   second_digit = (unsigned char)((display_data - first_digit * 10000) /
1000);
   third_digit  = (unsigned char)(((display_data - first_digit * 10000) -
second_digit * 1000) /100);
   fourth_digit = (unsigned char)((((display_data - first_digit * 10000) -
second_digit * 1000) - third_digit * 100) / 10);
   fifth_digit  = (unsigned char)((((display_data - first_digit * 10000) -
second_digit * 1000) - third_digit * 100) - fourth_digit * 10);

   if (display_data >= 10000) char_put(first_digit + '0');
   if (display_data >= 1000) char_put(second_digit + '0');
   if (display_data >= 100) char_put(third_digit + '0');
   if (display_data >= 10) char_put(fourth_digit + '0');
   char_put(fifth_digit + '0');
}
//______________________________________________________________________
```

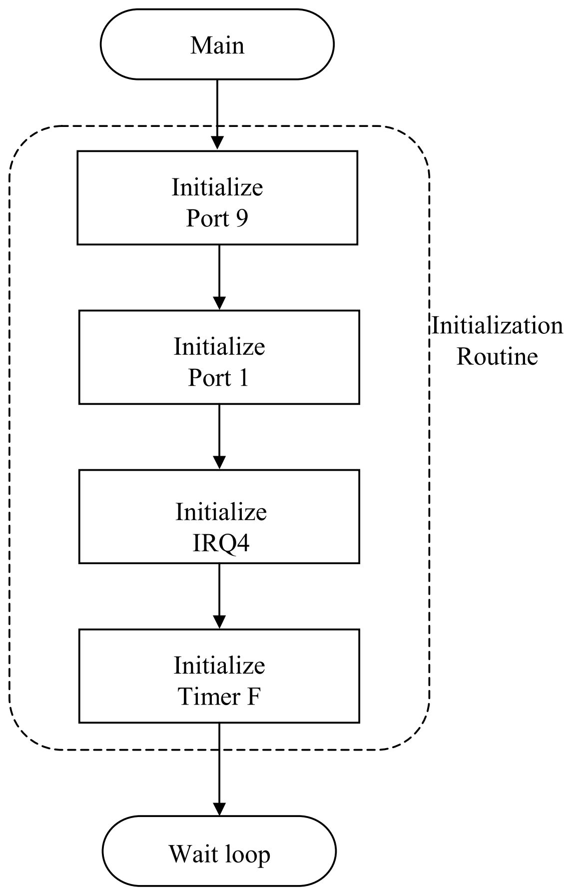


Figure 9 Main Program

```
/*********************************************************************/
/*                                                                   */
/*  FILE        :intprg.c                                            */
/*  DATE        :Tue, Feb 17, 2004                                   */
/*  DESCRIPTION :Interrupt Program                                   */
/*  CPU TYPE    :H8/38024F                                           */
/*                                                                   */
/*  This file is generated by Hitachi Project Generator (Ver.2.1).   */
/*                                                                   */
/*********************************************************************/
#include "iodefine.h"
#include "subfunctions.h"
#include <machine.h>

extern  unsigned int TIME, DISTANCE, COUNT, PREVIOUS_COUNT;
extern  unsigned int INPUT_CAPTURE, OVERFLOW_COUNT, MATCH, delay;
extern  unsigned long ADD_COUNT;
extern  signed int DIFFERENCE;

#pragma section IntPRG

// Vector 4 IRQ0
__interrupt(vect=4) void INT_IRQ0(void)
{
   int a;

   COUNT = TIME = DISTANCE = 0;

//STOP timer C counting as tigger selected as external source
   P_TMRC.TMC.BYTE = 0x9F;         // Auto reload, External Trigger (TMIC)

   INPUT_CAPTURE = (unsigned char) P_TMRC.TCCTLC;    // Read Timer C counter

   COUNT = (INPUT_CAPTURE + (OVERFLOW_COUNT * 256));
   PutStr(".");

   for (a = 0; a < 300; a ++);     // Short delay

   DIFFERENCE = COUNT - PREVIOUS_COUNT;

   if (DIFFERENCE > -4 && DIFFERENCE < 4)
   {
      MATCH ++;
      ADD_COUNT += COUNT;
   }

   PREVIOUS_COUNT = COUNT;

   if (MATCH == 5)    // Only display the distance detected after getting 5
                      //similar readings
   {
      TIME = (ADD_COUNT/5)* COUNT_PERIOD;

      DISTANCE = TIME / (CONV_DIST*2);
```

```
      if (DISTANCE == 0)
         PutStr("\rObject too close!\r\n\n");

      else
      {
         PutStr("\rDistance measured: ");
         display_decimal(DISTANCE + 5);  // Offset of 6cm required
         PutStr(" cm\r\n\n");
      }

      ADD_COUNT = 0;
      MATCH = 0;
   }

   OVERFLOW_COUNT = 0;              // Clear overflow counter
   P_SYSCR.IRR1.BIT.IRRI0 = 0;      // Clear interrupt request
}

// Vector 13 Timer C Overflow
__interrupt(vect=13) void INT_TimerC(void)
{
   OVERFLOW_COUNT++;   // Increment overflow counter when Timer C overflows
   P_SYSCR.IRR2.BIT.IRRTC = 0;      // Clear interrupt request
}

// Vector 14 Timer FL Overflow
__interrupt(vect=14) void INT_TimerFL(void)
{
   unsigned int i;

   P_TMRF.TCSRF.BIT.CMFL = 0;       // Clear overflow flag OVFL

   delay++;
   if (delay == 39)    // Transmit waveform every 65ms (1/153.6kHz) * 256 * 39
   {
      delay = 0;
      OVERFLOW_COUNT = 0;
      P_IO.PDR9.BIT.P92 ^= 1;       //Toggle LED D1 on P92

   // TRANSMIT
      P_IO.PMR3.BIT.TMOFH    = 1;  // P32 functions as TMOFH to output 40kHz
      for (i = 0; i < 312; i ++); // 0.5ms delay (transmit pulses for 0.5ms)
      P_IO.PMR3.BIT.TMOFH    = 0;  // P32 functions as I/O pin, pulses stopped

   // INITIALIZE TIMER C TO START DETECTION
      P_IO.PMRB.BIT.IRQ1 = 0;            // PB3 functions as IO pin
      P_TMRC.TMC.BYTE = 0x9B;            // Auto Reload, up-counter & internal
                                         //clk= phi/64
      P_TMRC.TCCTLC = 0x00;              // Set counting from 0 (TLC=0)
      P_SYSCR.IENR2.BIT.IENTC = 1;       // Enable Timer C interrupt
   }
   P_SYSCR.IRR2.BIT.IRRTFL = 0;    // Clear interrupt request
}
```

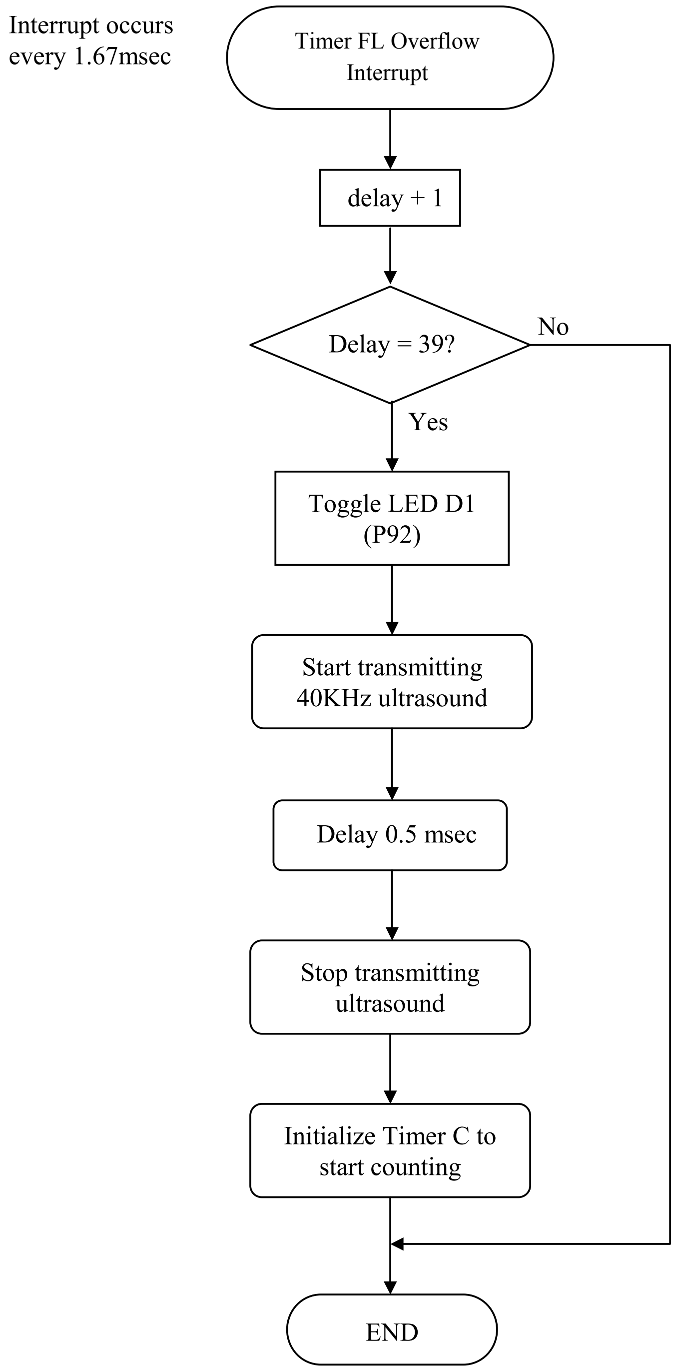


Figure 10 Timer FL Overflow Interrupt Service Routine

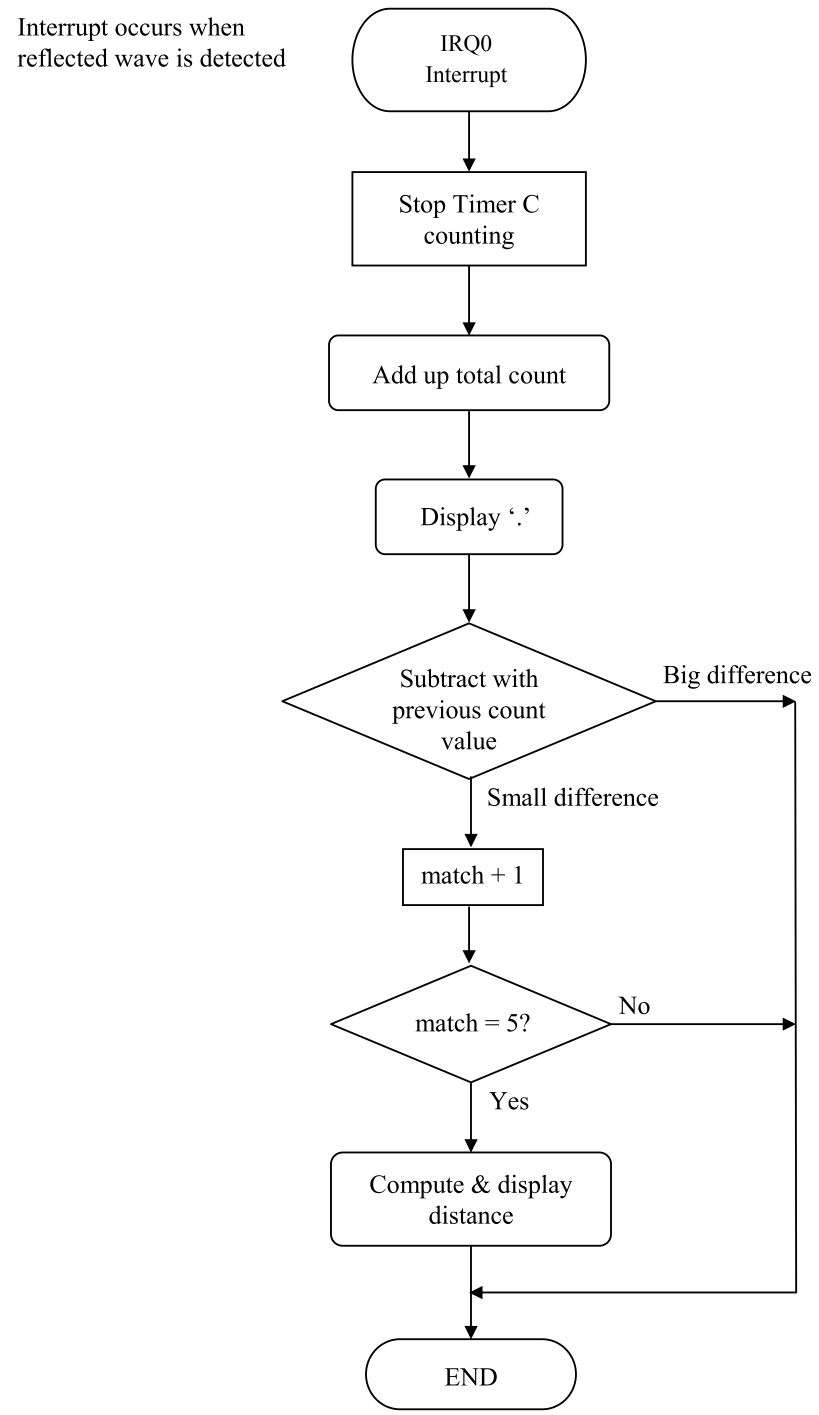


Figure 11 IRQ0 Interrupt Service Routine

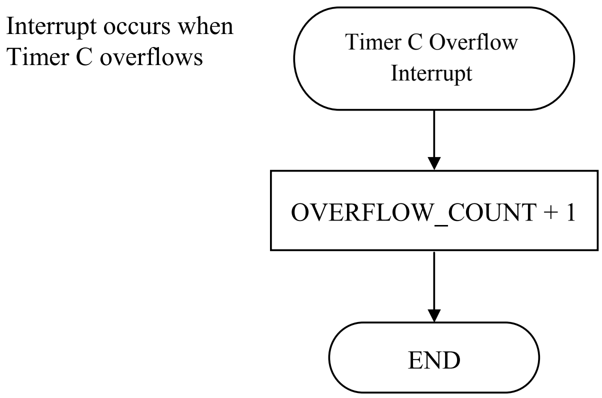
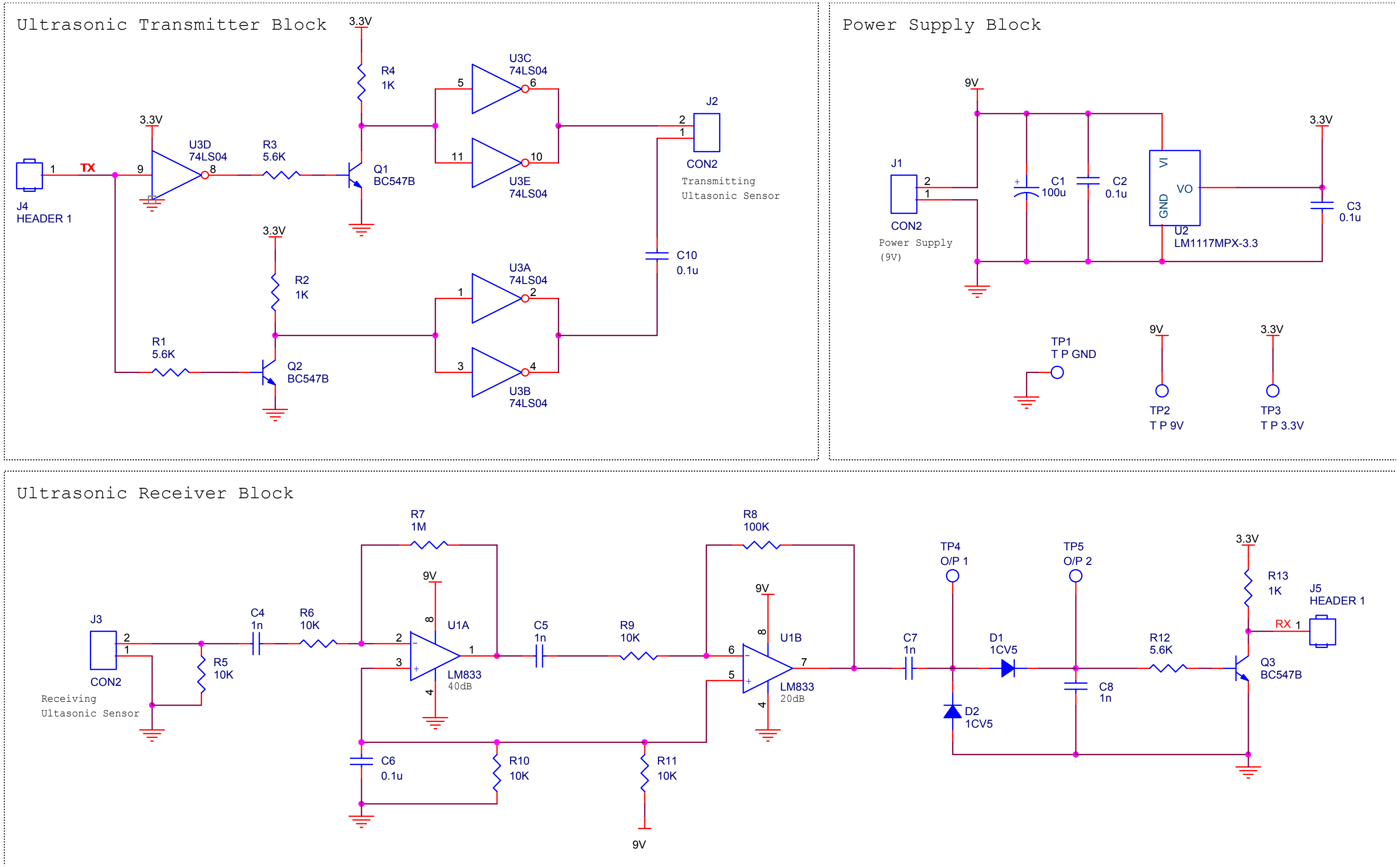


Figure 12 Timer C Overflow Interrupt Service Routine

4. Hardware Schematics



5. References

1. *H8/38024, H8/38024S, H8/38024F-ZTAT Group Hardware Manual*, Revision 4, 26 May 2003, Renesas Technology Corporation.
2. NIPPON CERAMIC CO., LTD. Open aperture Type - Air Transmission Ultrasonic Sensor
3. *LM1117/LM1117I 800mA Low-Dropout Linear Regulator*, 2002, National Semiconductor Corporation.
4. *LM833* Dual Audio Operational Amplifier, 2002, National Semiconductor Corporation.

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	March 2004	—	First edition issued

Keep safety first in your circuit designs!

1. Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
2. Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.