

**Northumbria University**

15

**EN0627 Report**

**Microcontroller Based Distance Measurement System**

**Adam Brown 10015647**

**Mitchell Smith 10017351**

Contents

[Introduction 1](#_Toc410310902)

[Aims 1](#_Toc410310903)

[Background 1](#_Toc410310904)

[Specification 1](#_Toc410310905)

[Team Contributions 1](#_Toc410310906)

[Theory 2](#_Toc410310907)

[Polaroid 6500 2](#_Toc410310908)

[Startup 2](#_Toc410310909)

[Operation 2](#_Toc410310910)

[Pulse Generation Time 2](#_Toc410310911)

[Time of Response between INIT and ECHO 2](#_Toc410310912)

[Radius of Curvature 3](#_Toc410310913)

[Calculating Result 3](#_Toc410310914)

[Passband and Tolerance 4](#_Toc410310915)

[Hardware 5](#_Toc410310916)

[Circuit Design 5](#_Toc410310917)

[Power Supply 5](#_Toc410310918)

[PIC 6](#_Toc410310919)

[Display 7](#_Toc410310920)

[LEDs 7](#_Toc410310921)

[Buttons 7](#_Toc410310922)

[External Connector 7](#_Toc410310923)

[PCB 8](#_Toc410310924)

[Enclosure 8](#_Toc410310925)

[Power Testing 9](#_Toc410310926)

[Costings 9](#_Toc410310927)

[Software 10](#_Toc410310928)

[Clock Speed 10](#_Toc410310929)

[Hardware Assignment 10](#_Toc410310930)

[PORTA 10](#_Toc410310931)

[PORTB 10](#_Toc410310932)

[Tools 10](#_Toc410310933)

[Techniques 11](#_Toc410310934)

[Handling Numbers 11](#_Toc410310935)

[Directives 11](#_Toc410310936)

[Macros 11](#_Toc410310937)

[Functions 12](#_Toc410310938)

[Startup 12](#_Toc410310939)

[Menu 13](#_Toc410310940)

[LCD 14](#_Toc410310941)

[Scan 16](#_Toc410310942)

[Numerical Processing 17](#_Toc410310943)

[Testing 17](#_Toc410310944)

[Software Debugging 17](#_Toc410310945)

[Stimulus 18](#_Toc410310946)

[Experiment 18](#_Toc410310947)

[Oscilloscope Traces 20](#_Toc410310948)

[Analysis of Results 21](#_Toc410310949)

[Conclusions 21](#_Toc410310950)

[Further Development 21](#_Toc410310951)

[Debouncing 21](#_Toc410310952)

[Error Checking 22](#_Toc410310953)

[Saving Results to EEPROM 22](#_Toc410310954)

[Device-PC Interface 22](#_Toc410310955)

[Bibliography 22](#_Toc410310956)

[Appendix A: Build Output 1](#_Toc410310957)

[Appendix B: Program Listing 1](#_Toc410310958)

[Appendix C: .map File 1](#_Toc410310959)

# Introduction

## Aims

The aim of the project was to design, prototype, and develop an ultrasonic sensor that met the specification with software written entirely in assembly.

## Background

TRISK, a Sunderland-based company, design and manufacture infra-red paint curing equipment for the automotive industry. They are currently replacing old control electronics and mechanical timers, and switches, with PIC microcontrollers. One upgrade under consideration is a distance indicator, since the distance of the infra-red heaters is critical for performance.

## Specification

* Must use a DC battery
* The PIC must use the sleep function
* Maximum power consumption:
* Maximum cost: £12
* Maximum enclosure size:
* Radius of curvature of panel detected:
* Dynamic passband range:
* Accuracy: 97.5%
* PCB design to industrial guidelines
* Unit must withstand overspray
* Unit must indicate when close to passband
* Unit must indicate when infra-red heater is too close

## Team Contributions

The project was divided into tasks for each team member. The responsibility matrix is below.

Table 1 (Continues on next page)

|  |  |  |
| --- | --- | --- |
|  | **Adam Brown** | **Mitchell Smith** |
| **Hardware Development** | **Primary** | **Secondary** |
| Circuit Design | Primary | Secondary |
| Circuit Testing | Primary | Secondary |
| PCB design | Primary | Secondary |
| PCB construction | Primary | Secondary |
| Component selection | Primary | Secondary |
| Costing | Primary | Secondary |
| **Software Development** | **Secondary** | **Primary** |
| Software design | Secondary | Primary |
| Software debugging | Secondary | Primary |
| Startup | Secondary | Primary |
| Menu | Secondary | Primary |
| Scan | Secondary | Primary |
| Numerical Processing | Secondary | Primary |
| LCD Control | Primary | Secondary |

# Theory

## Polaroid 6500

### Startup

### Operation

1. The PIC sets INIT
2. The Polaroid 6500 holds ECHO low and generates 16 ultrasonic pulses
3. The Polaroid 6500 waits for those pulses to return
4. The Polaroid 6500 releases ECHO (goes high again)

The distance between INIT being set and ECHO going high again is the distance between the pulses and an object

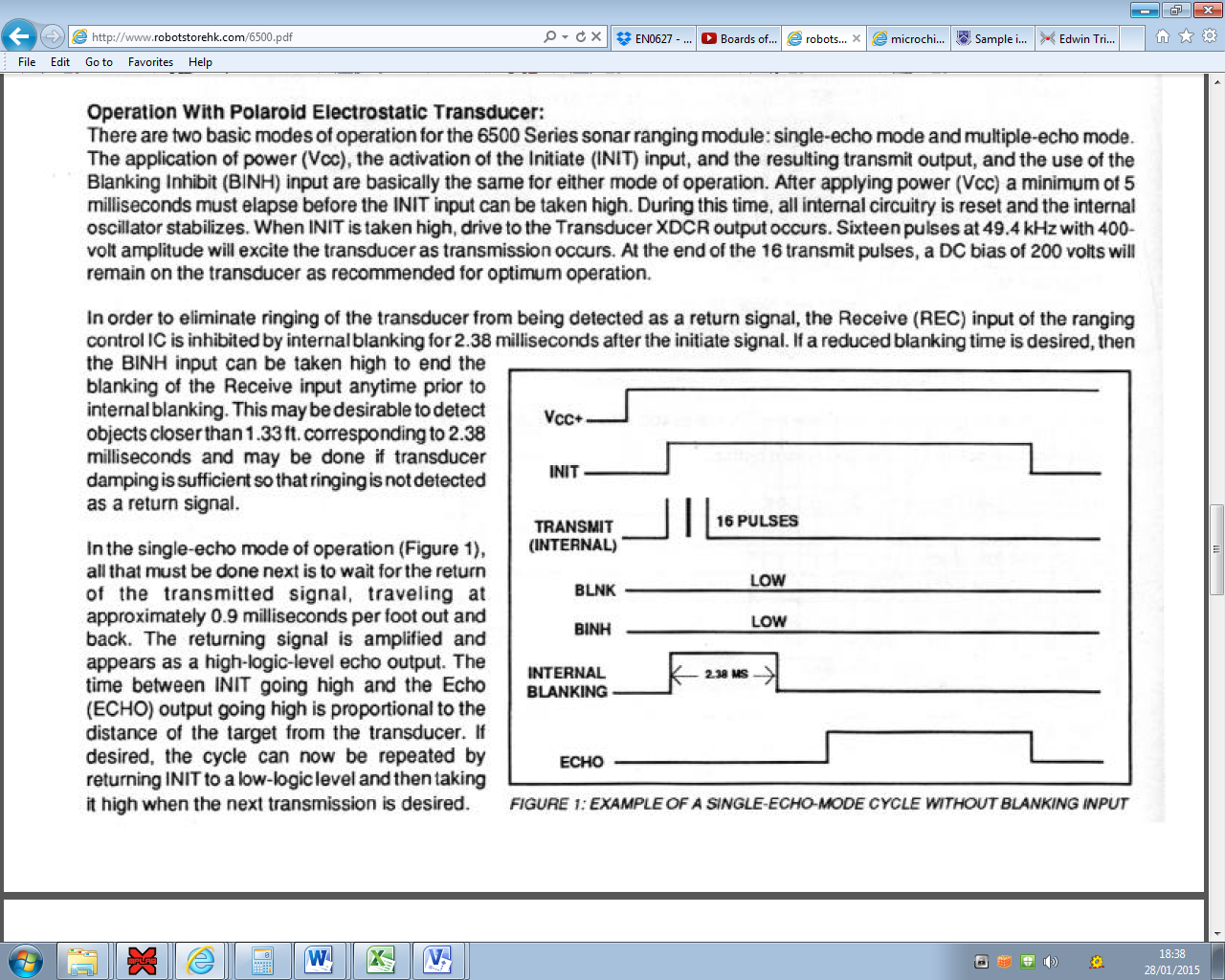


Figure 1 (Polaroid, p. 3)

### Pulse Generation Time

When INIT is set, the Polaroid 6500 generates 16 pulses at equivalent to . Equation 1 shows how this was calculated.

Equation 1

### Time of Response between INIT and ECHO

The ultrasonic pulses travel at to an object and back (Polaroid, p. 3) to the object and back. There are. Therefore, there are forward and back. Equation 2 shows how this was calculated and Equation 3 shows.

Equation 2

Equation 3

### Radius of Curvature

When an ultrasonic pulse hits a surface, it reflects back away from it. For an object with a surface at , the pulses are reflected straight back at the Polaroid 6500.

Figure 2 is a diagram showing how ultrasonic pulses are reflected off a surface at . It is clear that each of the pulses will return.

Figure 3 is a diagram showing how ultrasonic pulses are reflected off a perfect circle. The pulses are reflected off the object at different angles so it is not guaranteed that they will return.

Polaroid 6500

Ultrasonic pulses

Figure 2

Polaroid 6500

Figure 3

## Calculating Result

The result is calculated in 4 steps. These are Equation 4Equation 7.

|  |  |
| --- | --- |
| **Equation** | **Description** |
| Equation 4 | Subtract PULSEGEN (calibration error in terms of us) from result |
| Equation 5 | Multiply by distance per us. Distance per us is a fraction, so it is scaled by , to become 173 |
| Equation 6 | Divide result by to eliminate scaling factor |
| Equation 7 | Subtract LATENCY (calibration error in terms of mm) |

Equation 4

Equation 5

Equation 6

Equation 7

## Passband and Tolerance

The passband is the range in which the Green LED is lit. This corresponds to the ideal distance between a car and the infra-red heater. The tolerance band corresponds to distances “close” to the passband, but not within it, and lights the Amber LED.

[FIGURE] below is a diagram showing how the passband and tolerance bands work.

Pass band

Tolerance band

Fail

Fail

Figure 4

# Hardware

## Circuit Design

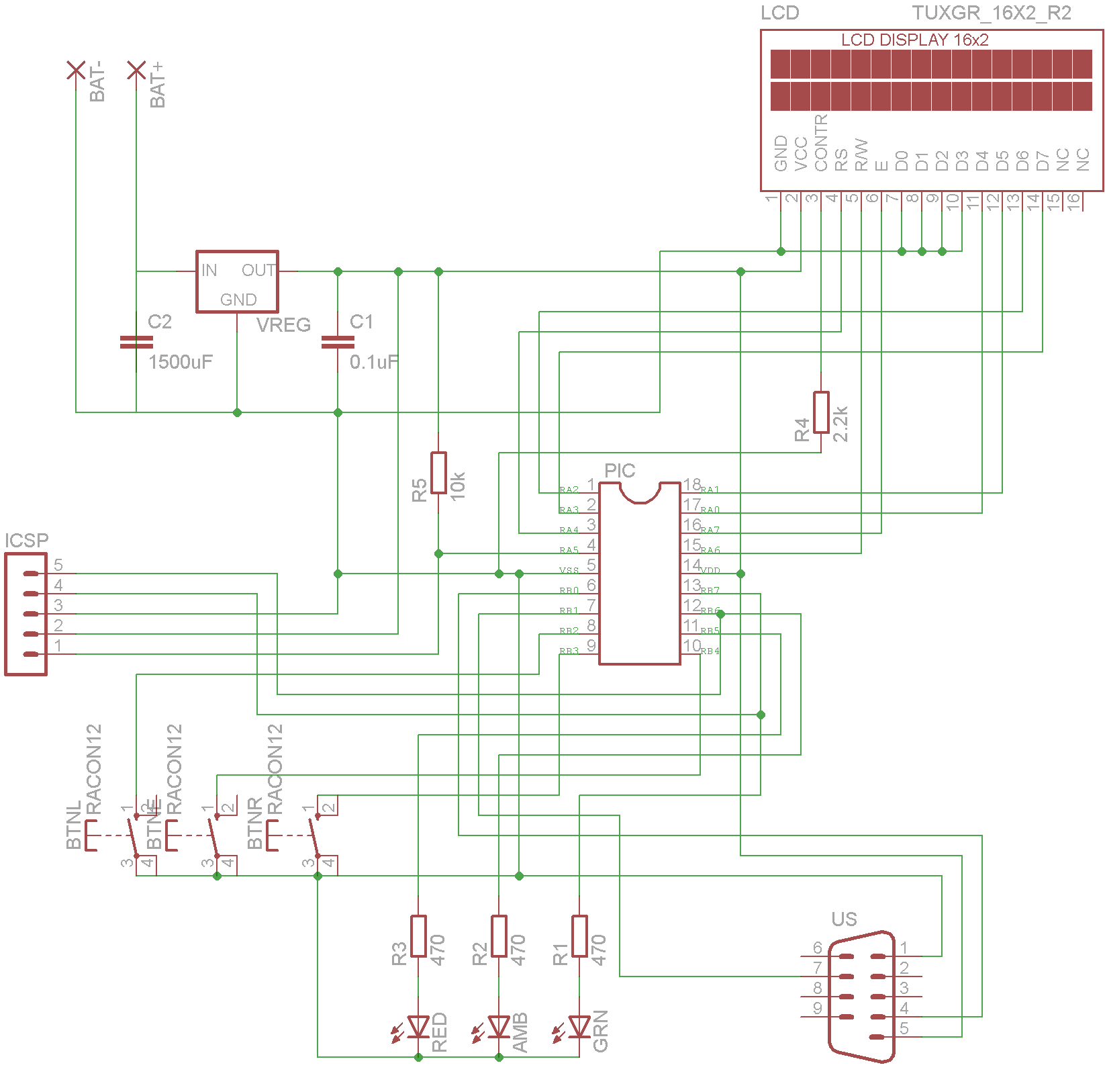


Figure 5

### Power Supply

The device power is supplied by a 9V PP3 size battery, this battery is held in place inside the casing by metal clip attached to the casing. Power is supplied through a battery clip wired in series with a switch; this is the main power on/off switch for the device. The power supply wires are soldered on to the PCB through holes; this allows the battery to clip into the case beneath the PCB.

The 9V power from the battery supplies a 5V 7805 series linear voltage regulator that provides power for the devices mounted on the PCB, there is a 100nF decoupling capacitor connected in parallel across the output as this is required by the regulator for proper operation. The specific voltage regulator used has a minimum voltage specification of 7V, this is sufficient to ensure the device will be usable throughout the normal voltage range of a typical 9V alkaline battery.

Power is supplied through the 9 pin connector to the ultrasound unit. The unit draws 100mA of current when not transmitting and 2000mA while transmitting at a voltage of 5V. As the current drawn when transmitting exceeds the current capabilities of the battery, an input capacitor is needed to meet peak current demands. The unit transmits 16 pulses at a frequency of 49.2kHz, giving a total transmit time of 325µs. The minimum voltage required by the ultrasound is 4.5V, so an input capacitor must not let the voltage drop more than 0.5V. 2A at 5V gives an input resistance of 2.5Ω.

Given this information, a required capacitance value can be calculated.

A capacitor with a capacitance of at least 1172µF is needed, as the next biggest commonly available capacitance is 1500 µF, this value was chosen for the input capacitor after researching costs.

### PIC

The microcontroller chosen was the Microchip PIC 16F1847, this is an 18 pin 8 bit PIC with 16 I/O pins available. This PIC was chosen as it had the features needed for the device, could be obtained at a good price, and could be programmed using the hardware available.

Initial research into the model of PIC to be used identified the PIC 16F1707 as the ideal PIC for the device due to its cost and features suited to the design. However as this PIC has only been recently released, the programming boards available did not support this model, and the PIC16F1847 was chosen as it was the second best in performance for price identified through research. (Microchip Technology, 2013b)

The 16F1847 has 16 I/O pins split over two ports (A and B) consisting of 15 pins configurable as inputs or outputs and one input only pin on port A. Pin allocation was decided based on the available features on both ports, as the Capture and Compare pins are located on port B, this port was reserved for the ultrasound interface. As the LCD display requires 7 pins for its operation, port A was reserved for this purpose as it has 7 bidirectional I/O pins. (Microchip Technology, 2013a)

To ensure easy removal and replacement of the PIC for external programming or in case it should develop a fault, the PIC was mounted on an 18 pin DIL socket. This allows the PIC to be removed without the need to de-solder which could result in damage to the rest of the board.

An In Circuit Serial Programming (ISCP) header is provided on the board for programming the PIC in situ. These pins interface with the appropriate pins on the pic to allow the use of an external programmer following the ISCP specification (Microchip Technology, 1997). The pins used for programming are the power supply pins (VDD and VSS) the programming voltage pin (VPP) and the ICSP Clock and Data lines (RB6 and RB7). The VPP pin is held high by a 10kΩ resistor.

### Display

The device makes use of an 8 character wide, 2 line LCD display to display information to the user. This information includes the current distance setting, options to increment and decrement the distance, measured distance and if the measured distance lies within the set tolerance.

The display used is the Powertip PC1602ARU, this display is a reflective Liquid Crystal Display (LCD) with no backlight. The decision to use a non-backlit display was made as backlit LCDs are more expensive and use considerably more power. The LCD controller is powered by a 5V supply and the LCD display by a voltage up to 5V, varying this voltage alters the contrast of the display and is achieved by connecting a resistor in series to ground from the VE (Contrast Voltage) pin. A resistance value of 2.2kΩ was found to provide an acceptable contrast level.

The LCD chosen uses the Hitachi HD44780 LCD controller which provides a standardised set of instructions for the LCD, the inputs consist of eight data lines (DB0 – DB7) and three control lines (RS, R/W and EN). The data lines are configured as outputs when R/W is held high, this is used to read a busy flag output on DB7 to indicate when the LCD controller can accept data or commands. Due to the number of I/O lines available on the microcontroller, the LCD controller was configured in four bit mode using only data lines DB4 – DB7, this is set during the initialisation of the controller.

The unused data lines (DB0 – DB3) are connected to ground to ensure they do not float and cause unintended behaviour of the LCD controller. The LCD board is connected to the main circuit board by header pins.

The LCD has an initialisation delay at power on which is slower than the controlling PIC so a delay has been introduced in the PIC initialisation to ensure the LCD is ready to accept commands when the PIC starts transmitting.

### LEDs

The device includes red, amber and green LEDs to indicate if an object is within an acceptable range of the device. The red LED indicates it is out of the set range, the amber that it is close to the set range, and the green that it is in range.

The anodes of the LEDs are connected to output pins on the microcontroller and the cathodes to ground through 470Ω current limiting resistors. The resistors limit the maximum current drawn per LED to 11mA which is below the microcontrollers maximum rated current per pin of 25mA. The measured forward bias voltages for the LEDs were 2.1V, 2.3V and 2.2V for the red, amber and green LEDs respectively; this gives a power level of 23 – 25 mW per LED which provides an acceptable level of brightness for the LEDs to act as indicators.

### Buttons

The device uses buttons to navigate the menus and set options, three buttons are used, for left, right and enter functions. The buttons are SPST push-to-make type, and are connected from the PIC input pins to ground. The internal pull up functionality of port B is used as this eliminates the need for external pull up resistors.

### External Connector

The connection to the ultrasound unit is made through a 9 pin DB9 size connector. The connector on the ultrasound is of the female type, so the decision was made to include a male type connector on the device, this will protrude from the side of the enclosure. The pins map directly so a straight-through type cable can be used.

The external connector carries power for the ultrasound unit so that no external power connections are required.

## PCB

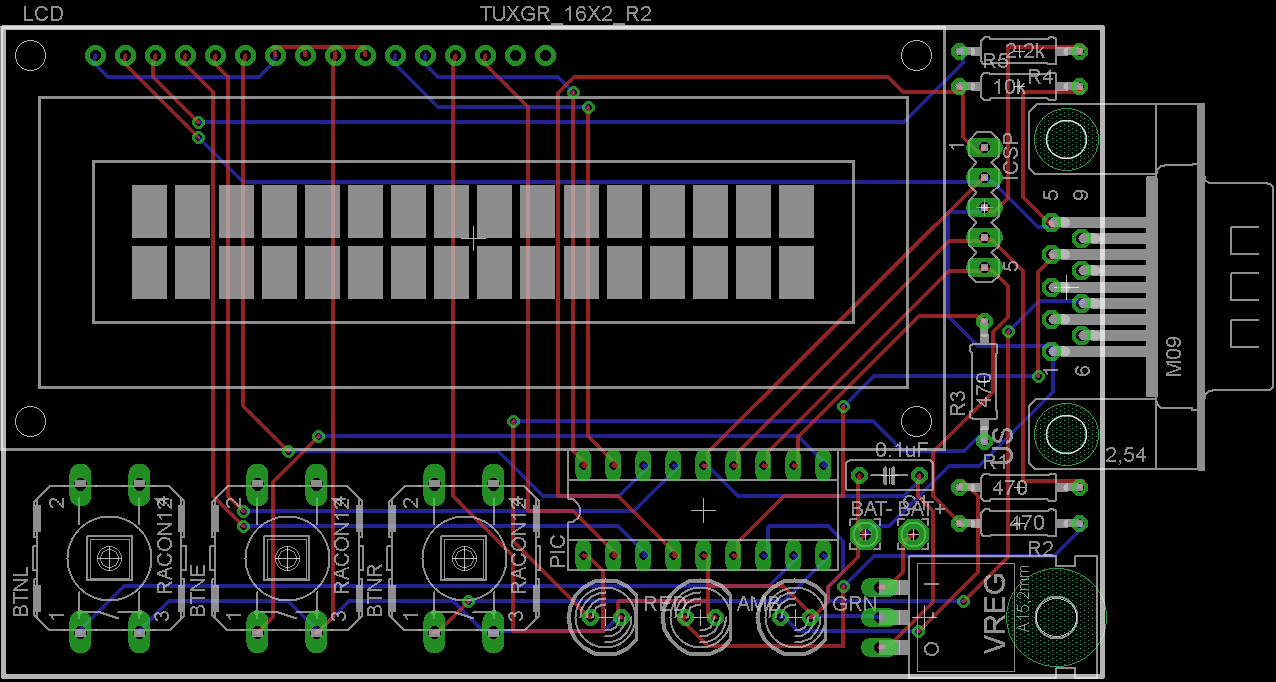


Figure 6

The circuit diagram was used to design a PCB for all the components. The designed PCB is a two layer, through hole design with top and bottom layers. The components were arranged to fit and then the tracks were routed to create connections, because of the dimensions of the LCD display, the rest of the circuitry has limited space and so is designed to be compact.

Holes are drilled into the PCB to mount components; these holes also provide electrical connections on both sides of the PCB which the components will be soldered into. As the PCB is a two layer design, special via holes are used to allow a track to cross from one side to the other, these must be drilled and filled with solder to function.

The PCB is sized to fit exactly into the enclosure chosen taking into account the space taken up by the lid screw mounts, this ensures all the available space is used. The 9 pin connector overhangs the right side of the PCB as it will protrude from the enclosure to enable the cable to be connected.

## Enclosure

The enclosure used for the device is an ABS plastic type enclosure with a lid secured with four screws. This type of enclosure was chosen due to its low cost and the ease in which it can be machined due to the type of plastic.

The amount an enclosure protects against solid and liquid ingress is measured by its IP rating, with the first digit corresponding to solid ingress and the second to liquid. The enclosure chosen is IP54 rated, this means that the enclosure is sealed with limited dust protection and protection from water spray from any direction. This is important as the device must be protected from overspray and be able to be wiped clean.

As the enclosure will be modified to allow for components to protrude, any gaps resulting will be sealed using hot melt glue to preserve the ingress protection.

## Power Testing

The power consumption of the board with and without the ultrasound unit attached is given in the table below.

Table 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Configuration** | **Voltage (V)** | **Current (mA)** | **Power (mW)** |
| **Board Only (idle)** | 4.931 | 1.340 | 6.608 |
| **Board with Ultrasound (idle)** | 4.931 | 39.50 | 194.8 |

These results show that the majority of the power consumption is due to the ultrasound device. From these figures, a typical fully charged 9V PP3 battery with a capacity of 565 mAh will last for approximately 422 hours with the ultrasound disconnected and approximately 13.3 hours with the ultrasound connected. (Duracell)

## Costings

Components were sourced online from Farnell, prices for a single unit and bulk purchase for 10 units are given. The highest cost item was the LCD display; this has been justified as it enhances the usability of the devices by displaying information clearly. The cost of the device is under the maximum cost set in the specification of £12.

Table 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component** | **Quantity** | **Model** | **Price (Single)** | **Price (10)** |
| LCD | 1 | POWERTIP PC1602ARU | £4.3300 | £3.7400 |
| PIC | 1 | [PIC16F1847](http://uk.farnell.com/microchip/pic16f1847-i-p/mcu-8bit-pic16-32mhz-dip-18/dp/1971869) | £1.8200 | £1.1700 |
| Enclosure | 1 | [Deltron Enclosure](http://uk.farnell.com/deltron-enclosures/479-0020-1/enclosure-multipurpose-abs-black/dp/2393701) | £1.0400 | £0.9580 |
| Connector | 1 | [D-SUB, PLUG, THT, R/A, 9WAY](http://uk.farnell.com/te-connectivity-amp/1734351-1/connector-d-sub-plug-tht-r-a-9way/dp/1653975) | £0.8500 | £0.8500 |
| Buttons | 3 | [SWITCH, SPST, 0.05A, 12VDC, THT](http://uk.farnell.com/multicomp/mcdts2-4n/switch-spst-0-05a-12vdc-tht/dp/9471634) | £0.5190 | £0.5190 |
| Voltage Regulator | 1 | [MC7805ACTG](http://uk.farnell.com/on-semiconductor/mc7805actg/ic-v-reg-5-0v-7805-to-220-3/dp/9664505) | £0.4780 | £0.3330 |
| On Switch | 1 | [SWITCH, ROCKER, DPST, ON-OFF](http://uk.farnell.com/multicomp/r13-244a-02-bb-3a/switch-rocker-dpst-on-off/dp/1634650) | £0.4590 | £0.4590 |
| LCD Header | 1 | [HEADER, 16WAY, 1ROW](http://uk.farnell.com/fci/68001-416hlf/board-board-conn-header-16way/dp/1834975) | £0.4240 | £0.3820 |
| Battery Leads | 1 | [Battery Strap](http://uk.farnell.com/multicomp-formerly-from-spc/8459-0673/battery-strap-9v-wire-lead/dp/1153006) | £0.2560 | £0.2560 |
| Power Capacitor | 1 | [CAP, ALU ELEC, 1500UF, 16V, RAD](http://uk.farnell.com/panasonic-electronic-components/eeufr1c152l/cap-alu-elec-1500uf-16v-rad/dp/1800640) | £0.1960 | £0.1960 |
| Battery Holder | 1 | [Battery Retainer](http://uk.farnell.com/keystone/71/battery-retainer/dp/1650708) | £0.1630 | £0.1630 |
| ICSP Header | 1 | [PIN HEADER, 5POS, 5MM](http://uk.farnell.com/multicomp/md022-50005/pin-header-5pos-5mm/dp/1708274) | £0.1520 | £0.1520 |
| DIP Socket | 1 | [DIP Socket](http://uk.farnell.com/multicomp/spc15500/dip-socket-18pos-through-hole/dp/1183575) | £0.1270 | £0.1270 |
| Switch Cap | 3 | [CAP, ROUND, ORANGE](http://uk.farnell.com/multicomp/ktsc-22s/cap-round-orange/dp/9561498) | £0.1233 | £0.1233 |
| Amber LED | 1 | [5MM, YELLOW, 30MCD, 588NM](http://uk.farnell.com/kingbright/l-1503yd/led-5mm-yellow-30mcd-588nm/dp/2335727) | £0.0643 | £0.0578 |
| Green LED | 1 | [5MM, GREEN, 100MCD, 568NM](http://uk.farnell.com/kingbright/l-1503gd/led-5mm-green-100mcd-568nm/dp/2335726) | £0.0643 | £0.0578 |
| Red LED | 1 | [5MM, RED, 40MCD, 617NM](http://uk.farnell.com/kingbright/l-1503id/led-5mm-red-40mcd-617nm/dp/2335725) | £0.0643 | £0.0578 |
| Supply Capacitor | 1 | [0.1uF Cap](http://uk.farnell.com/multicomp/mc0805b104k500a5-08mm/cap-mlcc-0-1uf-50v-x7r-rad/dp/2395773) | £0.0225 | £0.0192 |
| LED Resistors | 3 | [CARBON , 470R, 0.25W, 5%](http://uk.farnell.com/te-connectivity/cfr16j470r/resistor-carbon-470r-0-25w-5/dp/2329531) | £0.0135 | £0.0135 |
| **Total (Per Unit)** |  |  | **£11.1662** | **£9.6344** |

# Software

The software was written entirely in PIC assembly. This helps to ensure that the code is efficient and the developer has very precise control of the software’s wall-clock time (time it takes to execute a function).

## Clock Speed

The PIC uses a system clock. Operating at 5V, the typical current drawn by the PIC at 5V is (Microchip Technology Inc, p. 315), compared to the current drawn by the Polaroid 6500, this is very small, and guarantees that the program will operate quickly and can accurately measure the time between setting INIT and seeing a rising edge on ECHO.

## Hardware Assignment

### PORTA

The LCD uses 7 pins; 4 data pins (DB 4 – 7), Enable, Register Select and Read/Write; these were assigned to PORTA because implementing the algorithms necessary to communicate with the LCD were much more straightforward assuming all the pins were on a single port.

RA5 was not used because it is input only.

### PORTB

Because ECHO is connected as the CCP1 pin, it was necessary to place it on PORTB. It was assigned to PORTB.0, but it could have also been assigned to PORTB.3

Because the PIC sleeps when idle, the buttons needed to trigger an interrupt when pressed; for this reason they were assigned to PORTB, since this offers access to the Interrupt on Change feature.

The LEDs were assigned to leftover ports.

## Tools

#### IDE

We used the MPLABX development environment, and the MPASM toolkit to build the project.

The version information for the copy of MPLABX used is in Figure 7.

**Product Version:** MPLAB X IDE v2.05

**Java:** 1.7.0\_25; Java HotSpot(TM) 64-Bit Server VM 23.25-b01

**System:** Windows 7 version 6.1 running on amd64; Cp1252; en\_GB (mplab)

Figure 7

## Techniques

### Handling Numbers

The PIC is little-endian, this means that the most significant value of a number is represented by the largest bit in a register. This is crucial because it will affect how mathematical functions are developed, eg a quick divide is possible when the denominator is a power of, because this is equivalent to a bit shift. In a little-endian system the register must be shifted right places, but in a big-endian system the register would be shifted places to the left.

Little-endianness is illustrated by Table 4.

Table 4

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Decimal** | **Binary** | **Register (memory location = k)** | | | | | | | |
| 10 | 0000 1010 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

#### 16-Bit Numbers

16-bit numbers place the upper byte of data into the largest memory location, and the lower byte of data into the smallest memory location.

This is illustrated by Table 5

Table 5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Decimal** |  | **Binary** | **Upper register ( memory location = k + 1)** | | | | | | | | | | | |
| **40970** | Upper byte | 1010 0000 | b7 | b6 | b5 | b4 | | b3 | b2 | | b1 | | b0 | |
| 1 | 0 | 1 | 0 | | 0 | 0 | | 0 | | 0 | |
| Lower byte | 0000 1111 | Lower register (memory location = k) | | | | | | | | | | | |
| b7 | b6 | b5 | b4 | b3 | | | b2 | | b1 | | b0 |
| 0 | 0 | 0 | 0 | 1 | | | 1 | | 1 | | 1 |

It was essential to standardise the handling of 16-bit files to ensure that all files are handled correctly by the 16-bit numerical processing functions

### Directives

Directives are labels that the assembler recognises and substitutes for lines of code. They help make code human-readable.

### Macros

Macros generate code in locations where their respective labels are used. There is a tradeoff between using macros or using the call function and indirect registers: macros require more program memory, but call functions would mean passing files to and fro using the indirect registers (eg FSR0, FSR1, X, Y)

The principal advantage of using macros is improving the human-readability of the code; an action that may require five lines of code can be substituted for one. The point is illustrated in Figure 8.

|  |  |
| --- | --- |
| **Program Listing (pre-assembly)** | **Deassembly (post-assembly)** |
| **movff MACRO f1,f2**  **banksel f1**  **movf f1,W**  **banksel f2**  **movwf f2**  **ENDM**    **org 0x00**  **movff fA, fB**  **end** | org 0x00  banksel fA  movf fA,W  banksel fB  movwf fB  end |

Figure 8

Table 6

|  |  |
| --- | --- |
| **Macro** | **Description** |
| **movlf** | Moves 8-bit literal k into file f1 |
| **movlf16** | Moves 16-bit literal k into file f1 |
| **movff** | Copies 8-bit file f1 into 8-bit file f2 |
| **movff16** | Copies 16-bit file f1 into 16-bit file f2 |
| **clear** | Sets the contents of 8-bit file f1 to 0x00 |
| **movfi** | Copies the contents of 8-bit file f1 into indirect register fsrn |
| **movif16** | Copies the contents of fsrn and fsrn+1 into 16-bit register f1 |
| **movli** | Moves literal l1 into indirect register fsrn |
| **movli16** | Moves 16-bit literal l1 into the register pointed to by the contents of fsrn and fsrn+1 |
| **movif16** | Moves the 16-bit value contained in the address located in fsrn and fsrn+1 into 16-bit register f1 |
| **get\_ascii\_ln** | Converts the lower nibble (b3-b0)of the argument into an ASCII number between “0” and “9”. |
| **get\_ascii\_un** | Converts the upper nibble (b7-b4) of the argument into an ASCII character between “0” and “9” |

## Functions

### Startup

Startup initialises registers and the pinout. Includes a 5ms delay to ensure that the Polaroid 6500 is initialised after powerup

[Table] below shows the functions Startup offers

Table 7

|  |  |
| --- | --- |
| **Label** | **Description of code** |
| **Setup** | Initialises the PIC and includes 5ms delay to ensure Polaroid 6500 is initialised before it can be operated |

### Menu

The menu controls which function is called when ENTER is pressed, and cycling through these functions.

The menu set of functions includes those for altering the size of the passband and tolerance band.

Pressing enter makes the PIC jump to the location stored in menu\_pointer. When the user moves to a different menu, the location stored in menu\_pointer is updated.

[TABLE] below shows the functions Menu offers

Table 8

|  |  |
| --- | --- |
| **Label** | **Description of code** |
| **cycle\_left** | Cycle left increments the menu\_selected register and then error checks to ensure it does not overflow. If it does then the minimum value of menu\_selected for that menu is copied into it. |
| **cycle\_right** | Cycle left decrements the menu\_selected register and then error checks to ensure it does not underflow. If it does then the mmaximum value of menu\_selected for that menu is copied into it. |
| **start\_menu** | Copies the value of menu\_selected to W and uses this to branch to the option that the user selected. |
| **upper\_passband\_menu** | Copies the value of menu\_selected to W and uses this to branch to the option that the user selected. |
| **lower\_passband\_menu** | Copies the value of menu\_selected to W and uses this to branch to the option that the user selected. |
| **upper\_tolerance\_menu** | Copies the value of menu\_selected to W and uses this to branch to the option that the user selected. |
| **lower\_tolerance\_menu** | Copies the value of menu\_selected to W and uses this to branch to the option that the user selected. |
| **add\_100** | Adds 100 to the location stored in current\_value\_pointer. Then error checks to ensure that value does not exceed 2000mm. If it does then the value is set to 2000. |
| **add\_10** | Adds 10 to the location stored in current\_value\_pointer. Then error checks to ensure that value does not exceed 2000mm. If it does then the value is set to 2000. |
| **add\_1** | Adds 1 to the location stored in current\_value\_pointer. Then error checks to ensure that value does not exceed 2000mm. If it does then the value is set to 2000. |
| **sub\_100** | Subtracts 100 from the location stored in current\_value\_pointer. Then error checks to ensure that value does not go below 500mm. If it does then the value is set to 500. |
| **sub\_10** | Subtracts 10 from the location stored in current\_value\_pointer. Then error checks to ensure that value does not go below 500mm. If it does then the value is set to 500. |
| **sub\_1** | Subtracts 1 from the location stored in current\_value\_pointer. Then error checks to ensure that value does not go below 500mm. If it does then the value is set to 500. |

### LCD

[TABLE] below shows the functions LCD offers

Table 9

|  |  |
| --- | --- |
| **Label** | **Description of Code** |
| **lcd\_setup** | Initialises the LCD |
| **lcdbusy** | Prints the contents of FSR0 to the LCD, increments FSR0 and loops round until it reaches a null character, then it returns. |
|  |  |
| **lcdcmd** | Passes a specified command to the LCD |
| **set\_lcd\_top\_line** | Sets the LCD to the top line |
| **set\_lcd\_bottom\_line** | Sets the LCD to the bottom line |
| **lcd\_clear** | Clears the LCD screen |
| **lcdoff** | Switches the LCD off |
| **lcdon** | Switches the LCD on |
| **lcd\_print** |  |
| **update\_lcd\_start\_menu** | Copies menu\_selected to W and branches to the print function for that option. |
| **update\_lcd\_upper\_passband\_menu** | Copies menu\_selected to W and branches to the print function for that option. |
| **update\_lcd\_lower\_passband\_menu** | Copies menu\_selected to W and branches to the print function for that option. |
| **update\_lcd\_upper\_tolerance\_menu** | Copies menu\_selected to W and branches to the print function for that option. |
| **update\_lcd\_lower\_tolerance\_menu** | Copies menu\_selected to W and branches to the print function for that option. |
| **print\_start\_string** | Clears the LCD and then prints start\_string to the top line of the LCD |
| **print\_scan\_string** | Clears the LCD and then prints scan\_string to the top line of the LCD |
| **print\_pass\_string** | Converts scan result into the equivalent ASCII characters. Clears the LCD and then prints scan\_string to the top line of the LCD, then passes pass\_string, the result of the scan and mm\_string to the bottom line of the lcd. |
| **print\_tolerance\_string** | Converts scan result into the equivalent ASCII characters. Clears the LCD and then prints scan\_string to the top line of the LCD, then passes tolerance\_string, the result of the scan and mm\_string to the bottom line of the lcd. |
| **print\_fail\_string** | Converts scan result into the equivalent ASCII characters. Clears the LCD and then prints scan\_string to the top line of the LCD, then passes fail\_string, the result of the scan and mm\_string to the bottom line of the lcd. |
| **print\_timeout\_string** | Clears the LCD and then prints scan\_string to the top line of the LCD, then passes timeout\_string, to the bottom line of the lcd. |
| **print\_upper\_passband\_string** | Clears the LCD and then prints upper\_passband\_string to the top line of the LCD |
| **print\_lower\_passband\_string** | Clears the LCD and then prints lower\_passband\_string to the top line of the LCD |
| **print\_upper\_tolerance\_string** | Clears the LCD and then prints lower\_tolerance\_string to the top line of the LCD |
| **print\_lower\_tolerance\_string** | Clears the LCD and then prints lower\_tolerance\_string to the top line of the LCD |
| **print\_add\_100\_string** | Converts contents of register pointed to by current\_value\_pointer into equivalent ASCII characters. Clears the LCD and then prints add\_100\_string to the top line of the LCD, then passes the ASCII to the bottom line |
| **print\_add\_10\_string** | Converts contents of register pointed to by current\_value\_pointer into equivalent ASCII characters. Clears the LCD and then prints add\_10\_string to the top line of the LCD, then passes the ASCII to the bottom line |
| **print\_add\_1\_string** | Converts contents of register pointed to by current\_value\_pointer into equivalent ASCII characters. Clears the LCD and then prints add\_1\_string to the top line of the LCD, then passes the ASCII to the bottom line |
| **print\_sub\_100\_string** | Converts contents of register pointed to by current\_value\_pointer into equivalent ASCII characters. Clears the LCD and then prints sub\_100\_string to the top line of the LCD, then passes the ASCII to the bottom line |
| **print\_sub\_10\_string** | Converts contents of register pointed to by current\_value\_pointer into equivalent ASCII characters. Clears the LCD and then prints sub\_10\_string to the top line of the LCD, then passes the ASCII to the bottom line |
| **print\_sub\_1\_string** | Converts contents of register pointed to by current\_value\_pointer into equivalent ASCII characters. Clears the LCD and then prints sub\_1\_string to the top line of the LCD, then passes the ASCII to the bottom line |
| **hex16\_2\_ascii** | Converts a 16 bit file, f1 into a 4 character wide ASCII number. |

### Scan

Scan calls the Ultrasound function, which acquires a result and copies it into the X register. If a result was received, the number is processed and compared to the passband and tolerances. If there was a timeout, then the PIC jumps straight back to the end of Scan.

Scan converts the result from time into millimetres, and subtracts the pulse generation and electronic latency from the result.

Scan uses two assembler constants, PULSEGEN and LATENCY. These are fiddle factors that can correct unusual delays in the time between INIT and ECHO.

Conversion is performed using numerical processing functions (eg multiply, add, subtract). One alternative was to use a lookup table, with distances corresponding to specific values of CCP1. Although this would have been faster, it would also mean reserving a large block of program memory, and would have to be rewritten every time it is used with a sensor that is calibrated differently.

[TABLE] shows the functions Scan offers.

Table 10

|  |  |
| --- | --- |
| **Label** | **Description of code** |
| **scan** | Calls ultrasound and decides to move to scan\_pass or scan\_timeout depending on whether PASS or FAIL bit of usresult set |
| **scan\_pass** | Convert result from CCP1 to distance in MM  Checks if within passband, if above move to above\_passband, if below move to below\_passband, if within, sets PASS bit of usresult and moves to scan\_end |
| **below\_passband** | Checks if result is within lower tolerance, if within then set TOLERANCE bit of usresult and move to scan\_end, otherwise move to scan\_fail |
| **above\_passband** | Checks if result is within upper tolerance, if within then set TOLERANCE bit of usresult and move to scan\_end, otherwise move to scan\_fail |
| **scan\_fail** | Sets FAIL bit of usresult and moves to scan\_end |
| **scan\_timeout** | Sets TIMEOUT bit of usresult and moves to scan\_end |
| **scan\_end** | Returns |
| **ultrasound** | Gets result from Polaroid 6500 and returns with PASS or FAIL bit of usresult set depending on whether acquired result or timed out |
| **wait\_loop** | Wait in loop until rising edge on ECHO pin, or TMR1 overflows |
| **timeout** | Sets the FAIL bit of usresult and moves to got\_result |
| **success** | Sets PASS bit of usresult and moves to got\_result |
| **got\_result** | Clears init, CCP1CON and interrupt flags. Includes an 80ms delay using TMR0 so that the ultrasound is definitely cycled before the next scan |
| **ults\_end** | Disable and clear TMR0 setup |

#### Ultrasound

Ultrasound uses TMR1 and CCP1 to time the response from ECHO immediately after INIT has been set. It includes an recycle period delay[source] after acquiring the result. If TMR1 overflows ( then PIC sets usresult.TIMEOUT and returns to Scan

The function initiates a scan by setting INIT, and is cleared immediately after receiving a result or TMR1 overflow.

CCP1 is the best peripheral for capturing a result because it automatically copies the contents of TMR1 the cycle after a rising edge is seen on ECHO. This reduces the amount of code because the programmer does not have to copy the contents of TMR1 or subtract any offset between the rising edge of ECHO and copying TMR1.

### Numerical Processing

[TABLE] below shows the functions in this section.

Table 11

|  |  |
| --- | --- |
| **Label** | **Description of code** |
| **mult16x16** | Multiplies the 16-bit file X by 16-bit register Y, and the places result into 40-bit file result |
| **add16x16** | Adds the 16-bit file X by 16-bit register Y, and the places result into 40-bit file result |
| **subt16x16** | Subtracts the 16-bit file Y from 16-bit register X, and the places result into 40-bit file result |
| **div40bin** | Divides the 40-bit file result by 1,24. This is implemented by bit shifting right 10 places. |

# Testing

## Software Debugging

Before downloading the code to a PIC, it can be simulated in MPLABX. It is possible to see the contents of the data and program memory, and as such is very useful for ironing out software bugs.

Figure 9 shows the data memory displayed during simulation. It is also possible to see the program memory, SFRs and stack.

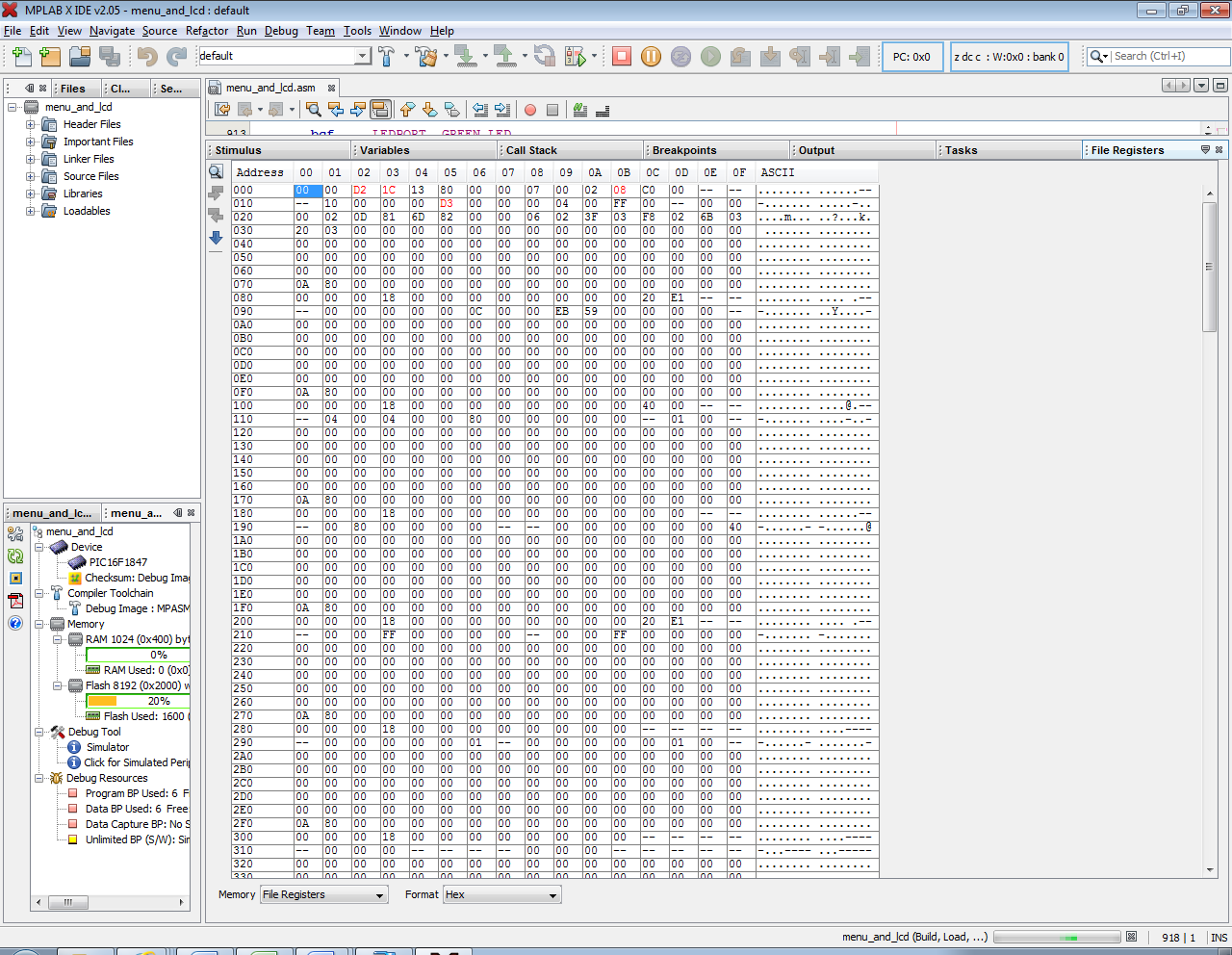


Figure 9

### Stimulus

The stimulus is a feature that emulates signals on pins. It is useful for testing inputs, such as the buttons and the ECHO pin. Using this meant that the software could be tested independently of the hardware.

Figure 10 is a picture of the stimulus window.

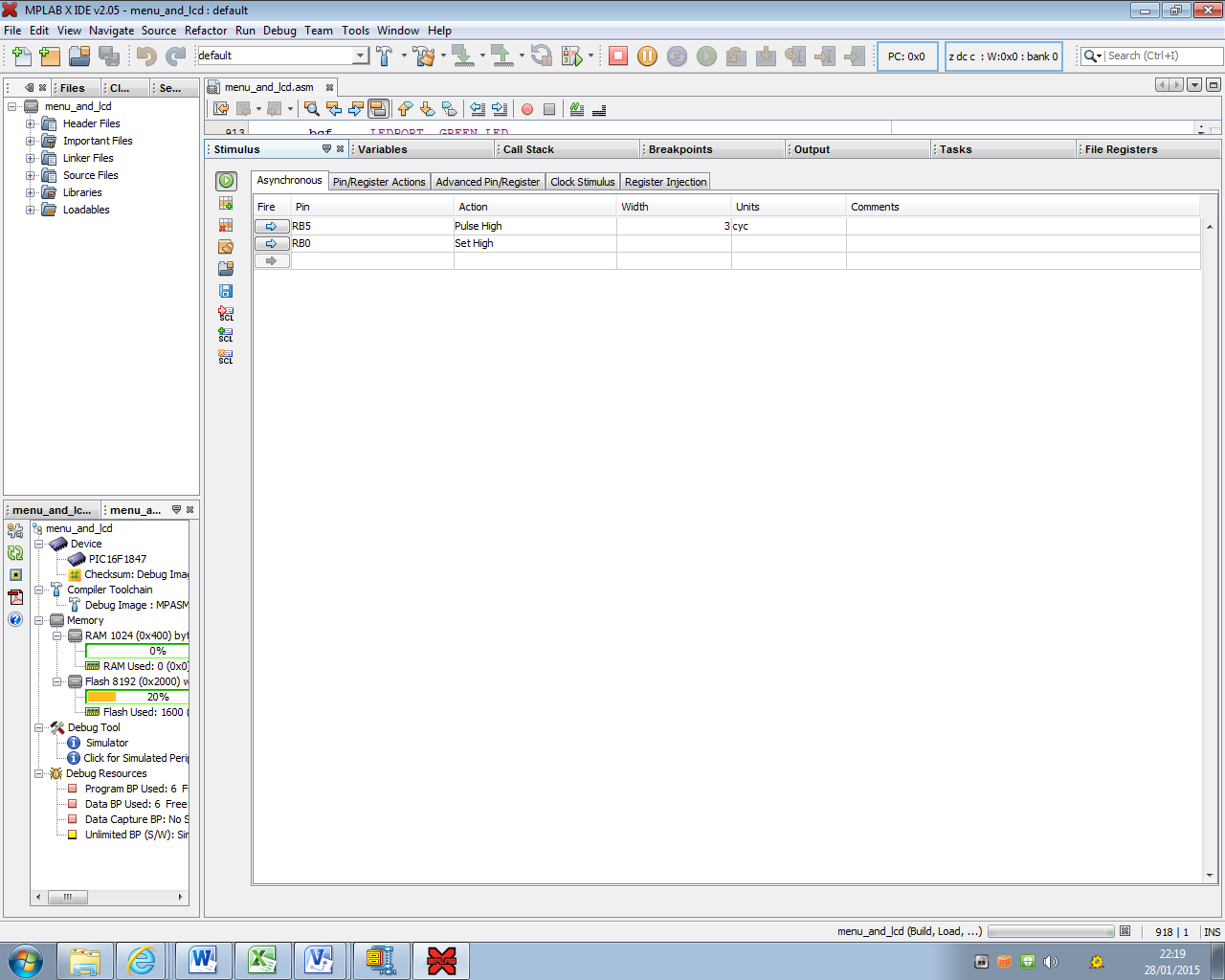


Figure 10

## Experiment

The ultrasound ranging device has been tested across its operational range of to , the results indicate that the ranging device meets the specification and that the human-device interface operates as expected.

The ultrasound module was tested using default passband and tolerance band, shown in [TABLE]

Table 12

|  |  |
| --- | --- |
| **Band** | **Values** |
| **Passband** | 875 mm – 800 mm |
| **Tolerance Band** | 919 mm – 760 mm |

[EQUATION] shows how the percentage accuracy was calculated.

Equation 8

[TABLE] below displays the results.

Actual results were determined using an uncalibrated meter rules. Measured values are those presented on the LCD screen after each test.

Table 13

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Actual** | **Measured** | **LED Lit** | **String Passed** | **Correct Behaviour?** | **% Accuracy** |
| 500 | 500 | Red | "FAIL" | Y | 100.00% |
| 550 | 550 | Red | "FAIL" | Y | 100.00% |
| 600 | 600 | Red | "FAIL" | Y | 100.00% |
| 650 | 650 | Red | "FAIL" | Y | 100.00% |
| 700 | 700 | Red | "FAIL" | Y | 100.00% |
| 750 | 750 | Red | "FAIL" | Y | 100.00% |
| 760 | 760 | Amber | "FAIL" | Y | 100.00% |
| 770 | 770 | Amber | "FAIL" | Y | 100.00% |
| 780 | 780 | Amber | "FAIL" | Y | 100.00% |
| 790 | 790 | Amber | "FAIL" | Y | 100.00% |
| 800 | 800 | Green | "PASS" | Y | 100.00% |
| 810 | 810 | Green | "PASS" | Y | 100.00% |
| 820 | 820 | Green | "PASS" | Y | 100.00% |
| 830 | 830 | Green | "PASS" | Y | 100.00% |
| 840 | 840 | Green | "PASS" | Y | 100.00% |
| 850 | 850 | Green | "PASS" | Y | 100.00% |
| 860 | 860 | Green | "PASS" | Y | 100.00% |
| 870 | 870 | Green | "PASS" | Y | 100.00% |
| 880 | 880 | Amber | "FAIL" | Y | 100.00% |
| 890 | 890 | Amber | "FAIL" | Y | 100.00% |
| 900 | 900 | Amber | "FAIL" | Y | 100.00% |
| 910 | 910 | Amber | "FAIL" | Y | 100.00% |
| 920 | 920 | Red | "FAIL" | Y | 100.00% |
| 930 | 930 | Red | "FAIL" | Y | 100.00% |
| 940 | 940 | Red | "FAIL" | Y | 100.00% |
| 950 | 951 | Red | "FAIL" | Y | 99.89% |
| 1000 | 1001 | Red | "FAIL" | Y | 99.90% |
| 1050 | 1051 | Red | "FAIL" | Y | 99.90% |
| 1100 | 1101 | Red | "FAIL" | Y | 99.91% |
| 1150 | 1151 | Red | "FAIL" | Y | 99.91% |
| 1200 | 1201 | Red | "FAIL" | Y | 99.92% |
| 1250 | 1252 | Red | "FAIL" | Y | 99.84% |
| 1300 | 1302 | Red | "FAIL" | Y | 99.85% |

### Oscilloscope Traces

The below traces show the relationship between the INIT signal (yellow) sent to the ultrasound unit and the ECHO signal (green) received back.

The traces in figure 11 show the very beginning of the INIT pulse sent to the ultrasound unit, the interference clearly shows the ultrasound sending pulses and its effect on the voltage supply.

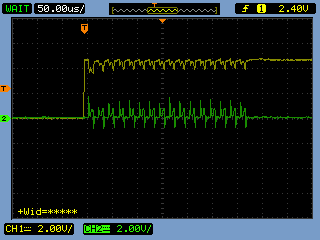


Figure 11

The traces in figure 12 show the end of the INIT pulse when an ECHO pulse is received from the ultrasound. The INIT pulse is shown to be pulled low after the PIC detects the ECHO pulse, there is an approximately 15µS delay which is within the accuracy specifications.

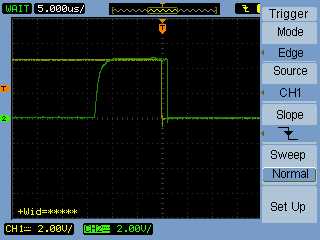


Figure 12

The traces in figure 13 show the whole INIT pulse and the EHO reply pulse. This trace was for a distance of 500mm which the measured pulse width of 2.95ms supports.

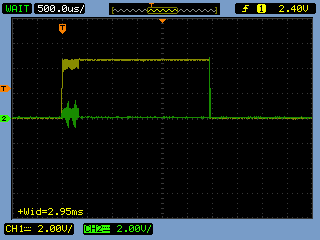


Figure 13

### Analysis of Results

The results show that the rangefinder correctly interprets results with a minimum of 99.85% accuracy. It may not be practical to use 1mm precision when testing at longer distances.

[FIGURE] is a graph of the results, and clearly displays the linearity of results.

(mm)

(mm)

Figure 14

# Conclusions

## Further Development

### Debouncing

The buttons occasionally bounce. This is a nuisance for the user. One solution would be to include a 1 second delay between a button being pushed and a button push being registered again.

### Error Checking

Comparing the pass and tolerance bands in the software would reduce rates of human error.

At present the PIC does not compare passband and tolerance bands, therefore it is possible for the passband to be wider than the tolerance band. The user must make sure that this is not the case.

### Saving Results to EEPROM

At the moment, all settings are stored in data memory, this means that when the power is cut, the settings are lost and must be reset to default during power up.

The EEPROM is non-volatile data memory, and would make it possible to store the settings before the device is switched off, and recover them during initialisation when the power is switched on.

### Device-PC Interface

At the moment the user has to be physically close to the device, to press buttons. One future development would be to add serial over USB support so that a user could control the sensor from a workstation. It would also be possible to log data to a file on the workstation.

This could increase the number of applications of the device.

# Bibliography

Harrington, P., & Dai, D. (2014, 12 18). Email conversation. (M. Smith, Interviewer)

Microchip Technology Inc. (n.d.). Retrieved 01 29, 2015, from Microchip Technology Inc Website: http://ww1.microchip.com/downloads/en/DeviceDoc/41453B.pdf

Northumbria University. (n.d.). *Coursework Assessment Specification: EN0627 2014-2015.*

Polaroid. (n.d.). Retrieved 01 29, 2015, from Robot Store (HK) Website: http://www.robotstorehk.com/6500.pdf

DURACELL. MN1604. Available: http://ww2.duracell.com/media/en-US/pdf/gtcl/Product\_Data\_Sheet/NA\_DATASHEETS/MN1604\_6LR61\_US\_CT.pdf.

MICROCHIP TECHNOLOGY. 1997. In-Circuit Serial Programming™ (ICSP™).

MICROCHIP TECHNOLOGY. 2013a. PIC16(L)F1847 Datasheet.

MICROCHIP TECHNOLOGY. 2013b. *Recently Released Products - Microchip Technology Inc.* [Online]. Available: <http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1013> [Accessed 28/01/2015.

# Appendix A: Build Output

make -f nbproject/Makefile-default.mk SUBPROJECTS= .build-conf

make[1]: Entering directory `D:/en0627\_assignment.X

make -f nbproject/Makefile-default.mk dist/default/production/en0627\_assignment.X.production.hex

make[2]: Entering directory `D:/en0627\_assignment.X

"C:\Program Files (x86)\Microchip\MPLABX\mpasmx\mpasmx.exe" -q -p16f1847 -l"build/default/production/en0627\_assignment.lst" -e"build/default/production/en0627\_assignment.err" -o"build/default/production/en0627\_assignment.o" "en0627\_assignment.asm"

Warning[210] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 104 : Extra ","

Warning[210] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 105 : Extra ","

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 123 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 123 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 158 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 158 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 279 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 280 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 281 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 282 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 283 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 284 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 285 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 286 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 290 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 158 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 292 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 293 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 294 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 295 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 301 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 302 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 303 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 158 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 347 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 350 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 353 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 158 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 138 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 138 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 1245 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 1246 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 1260 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 123 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 158 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 158 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 123 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 123 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 1312 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 123 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[305] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 1367 : Using default destination of 1 (file).

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 158 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 158 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 123 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 158 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 1448 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 158 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 1640 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 1641 : Register in operand not in bank 0. Ensure that bank bits are correct.

Message[302] D:\EN0627\_ASSIGNMENT.X\EN0627\_ASSIGNMENT.ASM 1642 : Register in operand not in bank 0. Ensure that bank bits are correct.

"C:\Program Files (x86)\Microchip\MPLABX\mpasmx\mplink.exe" -p16f1847 -w -m"dist/default/production/en0627\_assignment.X.production.map" -z\_\_MPLAB\_BUILD=1 -odist/default/production/en0627\_assignment.X.production.cof build/default/production/en0627\_assignment.o

MPLINK 5.01, LINKER

Device Database Version 1.18

Copyright (c) 1998-2011 Microchip Technology Inc.

Errors : 0

MP2HEX 5.01, COFF to HEX File Converter

Copyright (c) 1998-2011 Microchip Technology Inc.

Errors : 0

make[2]: Leaving directory `D:/en0627\_assignment.X

make[1]: Leaving directory `D:/en0627\_assignment.X

BUILD SUCCESSFUL (total time: 1s)

Loading code from D:/en0627\_assignment.X/dist/default/production/en0627\_assignment.X.production.hex...

Loading symbols from D:/en0627\_assignment.X/dist/default/production/en0627\_assignment.X.production.cof...

Loading completed

# Appendix B: Program Listing

|  |
| --- |
| D:\en0627\_assignment.X\en0627\_assignment.asm |

1 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 ; en0627\_assignment.asm

3 ; Authors....: Mitchell Smith (w10017351), Adam Brown (w10015647)

4 ; Revision...: 1.01

5 ; Date.......: 2015-01-28

6 ;

7 ; Description

8 ;

9 ; Designed to meet the specification given.

10 ; Calculates distance using result from Polaroid 6500 ultrasonic sensor

11 ; then determines if within acceptable limits and lights LEDs depending on

12 ; result

13 ;

14 ; Includes a menu that is printed to LCD for human-device interface

15 ;

16 ; Pinout

17 ; \_\_\_\_\_\_\_\_\_

18 ; RA2 / DB6 -| 16f1847 |- DB5 / RA1

19 ; RA3 / DB7 -| |- DB4 / RA0

20 ; RA4 / RS -| |- EN / RA7

21 ; RA5 / N/A -| |- RW / RA6

22 ; Vss -| |- Vdd

23 ; RB0 / ECHO -| |- RIGHT / RB7

24 ; RB1 / INIT -| |- LEFT / RB6

25 ; RB2 / RED\_LED -| |- ENTER / RB5

26 ; RB3 / AMBER\_LED -|\_\_\_\_\_\_\_\_\_|- GREEN\_LED / RB4

27 ;

28

29

30

31 list p=16F1847

32 #include <p16F1847.inc>

33

34 ; +++ CONSTANTS +++

35 DIST\_MAX EQU .1300 ; The maximum distance we calculate is 2000mm, 0x7D0

36 DIST\_MIN EQU .0500 ; The mimimum distance we calculate is 400mm, 0x190

37 PASSBAND\_MAX EQU .0875 ; 0x032A

38 PASSBAND\_MIN EQU .0800 ; 0x02DF

39 TOLERANCE\_MAX EQU .0919 ; PASSBAND\_MIX + 5% 0x0397

40 TOLERANCE\_MIN EQU .0760 ; PASSBAND\_MIN - 5% 0x02F8

41

42 MM\_PER\_US EQU 0x00AE ; (mm per second \* tinstruction \* 1024) = .174

43 PULSEGEN EQU 0x0000 ;0x01B8;\ The time it takes for ultrasound to generate pulses

44 LATENCY EQU 0x0000 ;0x00C8 ;\ 200mm

45 FAIL EQU 0

46 PASS EQU 1

47 TOLERANCE EQU 2

48 TIMEOUT EQU 3

49

50 ; The maximum value of menu\_selected. Must be equal to

51 ; limit = n +1 (eg options 1...3 .'. n = 3 + 1 = 4)

52 START\_MENU\_MAX EQU .6

53 UPPER\_PASSBAND\_MENU\_MAX EQU .8

54 LOWER\_PASSBAND\_MENU\_MAX EQU .8

55 UPPER\_TOLERANCE\_MENU\_MAX EQU .8

56 LOWER\_TOLERANCE\_MENU\_MAX EQU .8

57

58 ; The minimum value of menu\_selected for menus.

59 START\_MENU\_MIN EQU .2

60 LIMIT\_MENU\_MIN EQU .2

61 UPPER\_PASSBAND\_MENU\_MIN EQU .2

62 LOWER\_PASSBAND\_MENU\_MIN EQU .2

63 TOLERANCE\_MENU\_MIN EQU .2

64 UPPER\_TOLERANCE\_MENU\_MIN EQU .2

65 LOWER\_TOLERANCE\_MENU\_MIN EQU .2

66

67 ; Hardware Assignment

68 LCDPORT EQU PORTA

69 LCDDIR EQU TRISA

70 DB4 EQU .0

71 DB5 EQU .1

72 DB6 EQU .2

73 DB7 EQU .3

74 RS EQU .4

75 RW EQU .6

76 EN EQU .7

77

78 USPORT EQU PORTB

79 USDIR EQU TRISB

80 USPU EQU WPUB

81 ECHO EQU .0 ; RB.0 (Input, internal pullup)

82 INIT EQU .1 ; RB.1 (Output)

83

84 LEDPORT EQU PORTB

85 LEDDIR EQU PORTB

86 RED\_LED EQU .2 ; RB.5

87 AMBER\_LED EQU .3 ; RB.6

88 GREEN\_LED EQU .4 ; RB.7

89

90 BUTTONPORT EQU PORTB

91 BUTTONDIR EQU TRISB

92 BUTTONPU EQU WPUB

93 ENTER EQU .5 ; RB.4 (Input, internal pullup)

94 LEFT EQU .6 ; RB.2 (Input, internal pullup)

95 RIGHT EQU .7 ; RB.3 (Input, internal pullup)

96

97 ; LCD

98 #define LCDMASK 0x0F

99 #define STRH 0x00

100 #define STRL 0x20

101

102 ; Data Memory

103 cblock 0x20

104 **current\_menu:**1, **menu\_selected:**1, **menu\_pointer:**2, **lcd\_pointer:**2, **distance:**2,

105 **menu\_max:**1, **menu\_min:**1, **upper\_tolerance:**2, **lower\_tolerance:**2,

106 **upper\_passband:**2

107 endc

108 cblock 0x30

109 **lower\_passband:**2, **X:**2, **Y:**2, **BCD:**4 ,**result:**5, **count:**1

110 endc

111 cblock 0x40

112 **temp:**1,**usresult:**1,**ascii:**6, **current\_value\_pointer:**1

113 endc

114 cblock 0x70

115 **deltime:**1, **cmdbyte:**1

116 endc

117

118 ; Macros

119 ; Moves l into f

120 movlf MACRO ell,eff

121 banksel eff

122 **movlw** ell

123 **movwf** eff

124 ENDM

125

126 ; Moves 16bit l into 16bit f

127 movlf16 MACRO ell,eff

128 banksel eff

129 **movlw** low ell

130 **movwf** eff

131 **movlw** high ell

132 **movwf** eff+1

133 ENDM

134

135 ; Moves 8bit f1 into 8bit f2

136 movff MACRO f1,f2

137 banksel f1

138 **movf** f1,W

139 banksel f2

140 **movwf** f2

141 ENDM

142

143 ; Moves 16bit f1 into 16bit f2

144 movff16 MACRO f1,f2

145 banksel f1

146 **movf** f1,W

147 banksel f2

148 **movwf** f2

149 banksel f1+1

150 **movf** f1+1,W

151 banksel f2

152 **movwf** f2+1

153 ENDM

154

155 ; Selects and clears f1

156 clear MACRO f1

157 banksel f1

158 **clrf** f1

159 ENDM

160

161 ; Moves f into FSRn

162 movfi MACRO f1,fsrn

163 banksel f1

164 **movf** f1,W

165 **movwi** 0[fsrn]

166 ENDM

167

168 ; Loads fsrn and fsrn+1 into a 16-bit file

169 movif16 MACRO fsrn,f1

170 **moviw** 0[fsrn]

171 banksel f1

172 **movwf** f1

173 **moviw** 1[fsrn]

174 **movwf** f1+1

175 ENDM

176

177 ; Moves l into fsrn

178 movli MACRO ell,fsrn

179 **movlw** ell

180 **movwi** 0[fsrn]

181 ENDM

182

183 ; Moves 16-bit l into fsrn and fsrn+1

184 movli16 MACRO ell,fsrn

185 **movlw** low ell

186 **movwi** 0[fsrn]

187 **movlw** high ell

188 **movwi** 1[fsrn]

189 ENDM

190

191 ; Moves 0[FSR0] and 1[FSR0] into f1 and f1+1

192 moviw16 MACRO fsrn,f1

193 **moviw** 0[fsrn]

194 banksel f1

195 **movwf** f1

196

197 **moviw** 1[fsrn]

198 **movwf** f1+1

199 ENDM

200

201 ; Returns the lower nibble of a file as ASCII character using lookup table

202 get\_ascii\_ln MACRO f1,f2

203 banksel f1

204 **movf** f1,W

205 **andlw** 0x0F

206 **call** get\_ascii

207 banksel f2

208 **movwf** f2

209 ENDM

210

211 ; Returns the upper nibble of a file as ASCII chaarcter using lookup table

212 get\_ascii\_un MACRO f1,f2

213 banksel f1

214 **swapf** f1,W

215 **andlw** 0x0F

216 **call** get\_ascii

217 banksel f2

218 **movwf** f2

219 ENDM

220

221 ; Subtract l from f, place in w

222 subfl\_to\_w MACRO l1,f1

223 banksel f1

224 **movlw** l1

225 **subwf** f1,w

226 ENDM

227

228 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

229 ; Program

230 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

231

232 org 0x00

233 **goto** setup

234 org 0x04

235 ISR

236 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

237 ; Strings

238 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

239 start\_string dt "START MENU\0"

240 scan\_string dt "SCAN\0"

241 testing\_string dt "TESTING\0"

242 upper\_passband\_string dt "UPPER PSBND\0"

243 lower\_passband\_string dt "LOWER PSBND\0"

244 upper\_tolerance\_string dt "UPPER TOL\0"

245 lower\_tolerance\_string dt "LOWER TOL\0"

246 pass\_string dt "PASS: \0"

247 fail\_string dt "FAIL: \0"

248 timeout\_string dt "USOUND TIMEOUT\0"

249 mm\_string dt " MM\0"

250 add\_100\_string dt "ADD 100\0"

251 add\_10\_string dt "ADD 10\0"

252 add\_1\_string dt "ADD 1\0"

253 sub\_100\_string dt "SUB 100\0"

254 sub\_10\_string dt "SUB 10\0"

255 sub\_1\_string dt "SUB 1\0"

256

257 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

258 ; Setup

259 ;

260 ; Contains code for initialisation of PIC and ultrasound

261 ; Only ever runs once, when PIC started up

262 ;

263 ; Initialises the system clock, pinout, LCD and waits 5ms to ensure Polaroid

264 ; 6500 intialised.

265 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

266 setup

267 ; use 4Mhz fosc

268 movlf 0xEB,OSCCON

269

270 movlf 0xF0,TMR0

271 **bsf** INTCON,TMR0IE ; Set TMR0 to count 5ms w/ pre of 4 and use Finst

272 movlf 0x01,OPTION\_REG

273

274 ; Make sure no PORTB pins are analogue pins

275 clear ANSELB

276

277 ; Initialise pin direction

278 clear TRISB ;\ Set buttons as inputs

279 **bsf** TRISB,LEFT ;|

280 **bsf** TRISB,RIGHT ;|

281 **bsf** TRISB,ENTER ;/

282 **bcf** TRISB,GREEN\_LED ;\ Set LED pins as outputs

283 **bcf** TRISB,AMBER\_LED ;|

284 **bcf** TRISB,RED\_LED ;/

285 **bcf** TRISB,INIT ; Set INIT as output

286 **bsf** TRISB,ECHO ; Set ECHO as input

287

288 ; Switch CCP1 from RB3 to RB0

289 banksel APFCON0

290 **bsf** APFCON0,CCP1SEL

291 clear WPUB ;\Set pullups for buttons, will use

292 **bsf** WPUB,LEFT ;| -ive edge triggering

293 **bsf** WPUB,RIGHT ;|

294 **bsf** WPUB,ENTER ;/

295 **bsf** WPUB,ECHO

296 ; Initialise the LCD

297 **call** lcd\_setup

298

299 ; Enable -ive edge interrupts on buttons

300 banksel IOCBN

301 **bsf** IOCBN,LEFT

302 **bsf** IOCBN,RIGHT

303 **bsf** IOCBN,ENTER

304

305 ; Wait 5ms so that Polaroid 6500 is definitely initialised

306 **btfss** INTCON,TMR0IF

307 **goto** $-1

308

309 ; Disable TMR0 and clear IF

310 **bcf** INTCON,TMR0IE

311 **bcf** INTCON,TMR0IF

312 clear OPTION\_REG

313

314 ; Initialise variables

315 movlf16 TOLERANCE\_MIN,lower\_tolerance

316 movlf16 TOLERANCE\_MAX, upper\_tolerance

317 movlf16 PASSBAND\_MIN, lower\_passband

318 movlf16 PASSBAND\_MAX, upper\_passband

319

320 ; Initialise the menu

321 **goto** set\_start

322

323 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

324 ; Main

325 ;

326 ; Program loop

327 ; Default state is sleep, woken when a button is pressed

328 ;

329 ; Will then move to perform action appropriate for the button pressed

330 ;

331 ; When that action is finished, PIC moves into update\_lcd,

332 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

333 main

334 **bsf** INTCON,IOCIE

335 **sleep** ; Wait for input

336

337 ; TMR0 sometimes overflows, not sure why yet but this stops it from

338 ; waking the PIC up

339 **btfsc** INTCON,TMR0IF

340 **goto** TMR0\_reset

341

342 ; Disable IOC pins

343 clear PIR1

344 **bcf** INTCON,IOCIE

345

346 banksel IOCBF

347 **btfsc** IOCBF,LEFT

348 **goto** cycle\_left

349

350 **btfsc** IOCBF,RIGHT

351 **goto** cycle\_right

352

353 **btfsc** IOCBF,ENTER

354 **goto** goto\_menu

355

356 TMR0\_reset

357 **bcf** INTCON,TMR0IF

358 **goto** main

359

360 ; Jump to location stored in menu\_pointer

361 goto\_menu

362 banksel menu\_pointer

363 movff menu\_pointer+1,PCLATH

364 **bsf** PCLATH,7

365 movff menu\_pointer,PCL

366

367 ; Cycles "left" through menu, error checks to make sure never go above max limit

368 cycle\_left

369 banksel menu\_selected

370 **incf** menu\_selected,F

371 **call** error\_check\_upper

372 **goto** update\_lcd

373

374 ; Cycles "right" thru menu, error checks to make sure never go above max limit

375 cycle\_right

376 banksel menu\_selected

377 **decf** menu\_selected,F

378 **call** error\_check\_lower

379 **goto** update\_lcd

380

381 ; Jump to location stored in lcd\_pointer

382 update\_lcd

383 movff lcd\_pointer+1,PCLATH

384 movff lcd\_pointer,PCL

385

386 ; Get ready to jump back to main

387 fin

388 clear IOCBF

389 **goto** main

390

391 ; Check if menu\_option is less than minimum value for current menu, if so then

392 ; copies minimum value into menu\_option

393 error\_check\_lower

394 banksel menu\_min

395 **movf** menu\_min,W

396 **subwf** menu\_selected,W

397 **movf** menu\_max,W

398 **btfss** STATUS,C

399 **movwf** menu\_selected

400 **return**

401

402 ; Check if menu\_option is grater than maximum value for current menu, if so then

403 ; copies maximum value into menu\_option

404 error\_check\_upper

405 banksel menu\_max

406 **movf** menu\_max,W

407 **addlw** 0x01

408 **subwf** menu\_selected,W

409 **movf** menu\_min,W

410 **btfsc** STATUS,C

411 **movwf** menu\_selected

412 **return**

413

414 ; Jump to here from goto\_menu when in start menu and enter button pushed. Branch

415 ; to option currently selected

416 start\_menu

417 **movf** menu\_selected,W

418 **brw**

419 **goto** software\_error ; W = 0

420 **goto** software\_error ; W = 1

421 **goto** perform\_scan ; W = 2

422 **goto** set\_upper\_passband ; W = 3

423 **goto** set\_lower\_passband ; W = 4

424 **goto** set\_upper\_tolerance; W = 5

425 **goto** set\_lower\_tolerance; W = 6

426

427 ; Jump to here from goto\_menu when in upper passband menu and enter button

428 ; pushed. Branch to option currently selected

429 upper\_passband\_menu

430 **movf** menu\_selected,W

431 **brw**

432 **NOP** ;error

433 **NOP** ; error()

434 **goto** add\_100 ; add 100 to value

435 **goto** add\_10 ; add 10 from value

436 **goto** add\_1 ; add 1 from value

437 **goto** sub\_100 ; subtract 100 from value

438 **goto** sub\_10 ; subtract 10 from value

439 **goto** sub\_1 ; subtract 1 from value

440 **goto** set\_start

441

442 ; Jump to here from goto\_menu when in lower passband menu and enter button

443 ; pushed. Branch to option currently selected

444 lower\_passband\_menu

445 **movf** menu\_selected,W

446 **brw**

447 **NOP** ;error

448 **NOP** ; error()

449 **goto** add\_100 ; add 100 to value

450 **goto** add\_10 ; add 10 from value

451 **goto** add\_1 ; add 1 from value

452 **goto** sub\_100 ; subtract 100 from value

453 **goto** sub\_10 ; subtract 10 from value

454 **goto** sub\_1 ; subtract 1 from value

455 **goto** set\_start

456

457 ; Jump to here from goto\_menu when in upper tolerance menu and enter button

458 ; pushed. Branch to option currently selected

459 upper\_tolerance\_menu

460 **movf** menu\_selected,W

461 **brw**

462 **NOP** ;error

463 **NOP** ; error()

464 **goto** add\_100 ; add 100 to value

465 **goto** add\_10 ; add 10 from value

466 **goto** add\_1 ; add 1 from value

467 **goto** sub\_100 ; subtract 100 from value

468 **goto** sub\_10 ; subtract 10 from value

469 **goto** sub\_1 ; subtract 1 from value

470 **goto** set\_start

471

472 ; Jump to here from goto\_menu when in lower tolerance menu and enter button

473 ; pushed. Branch to option currently selected

474 lower\_tolerance\_menu

475 **movf** menu\_selected,W

476 **brw**

477 **NOP** ;error

478 **NOP** ; error()

479 **goto** add\_100 ; add 100 to value

480 **goto** add\_10 ; add 10 from value

481 **goto** add\_1 ; add 1 from value

482 **goto** sub\_100 ; subtract 100 from value

483 **goto** sub\_10 ; subtract 10 from value

484 **goto** sub\_1 ; subtract 1 from value

485 **goto** set\_start

486

487 ; Jump to here from goto\_menu when enter button pushed and "add 100" selected

488 ; Adds 100 the value located at the address stored in current\_value\_pointer, then error

489 ; checks to make sure it never goes above the absolute limit

490 add\_100

491 movff16 current\_value\_pointer,FSR0

492

493 ; Add 100 to lower byte of current\_value

494 **MOVIW** 0[FSR0]

495 **addlw** .100 ; 0x64

496 **MOVWI** 0[FSR0]

497

498 ; If there was an overflow, add 1 to upper byte of current\_value

499 **MOVIW** 1[FSR0]

500 **btfsc** STATUS,C

501 **addlw** .1

502 **MOVWI** 1[FSR0]

503

504

505 **call** add\_error\_check

506 **btfsc** WREG,FAIL

507 **call** set\_max\_value

508

509 **goto** update\_lcd ; write the new distance to the lcd

510

511 ; Jump to here from goto\_menu when enter button pushed and "add 10" selected

512 ; Adds 10 to the value located at the address stored in current\_value\_pointer, then error

513 ; checks to make sure it never goes above the absolute maximum limit

514 add\_10

515 movff16 current\_value\_pointer,FSR0

516

517 ; Add 10 to lower byte of current\_value

518 **MOVIW** 0[FSR0] ; move distance into w

519 **addlw** .10 ; 0x0A

520 **MOVWI** 0[FSR0]

521

522 ; If there was an overflow, add 1 to upper byte of current\_value

523 **MOVIW** 1[FSR0]

524 **btfsc** STATUS,C

525 **addlw** .1

526 **MOVWI** 1[FSR0]

527

528 ; Check if result > MAX\_DIST, if so set current value to MAX\_DIST

529 **call** add\_error\_check; error check: within limit?

530 **btfsc** WREG,FAIL ; if without limit, set to max value

531 **call** set\_max\_value

532

533 **goto** update\_lcd ; write the new distance to the lcd

534

535 ; Jump to here from goto\_menu when enter button pushed and "add 1" selected

536 ; Adds 1 to the value located at the address stored in current\_value\_pointer, then

537 ; error checks to make sure it never goes above the absolute maximum limit

538 add\_1

539 movff16 current\_value\_pointer,FSR0

540

541 ; Add 1 to lower byte of current\_value

542 **MOVIW** 0[FSR0] ; move distance into w

543 **addlw** .1 ; 0x01

544 **MOVWI** 0[FSR0]

545

546 ; If there was an overflow, add 1 to upper byte of current\_value

547 **MOVIW** 1[FSR0]

548 **btfsc** STATUS,C

549 **addlw** .1

550 **MOVWI** 1[FSR0]

551

552 ; Check if result > MAX\_DIST, if so set current value to MAX\_DIST

553 **call** add\_error\_check; error check: within limit?

554 **btfsc** WREG,FAIL ; if without limit, set to max value

555 **call** set\_max\_value

556

557 **goto** update\_lcd ; write the new distance to the lcd

558

559 ; Jump to here from goto\_menu when enter button pushed and "sub 100" selected

560 ; Subtracts 100 from the value located at the address stored in

561 ; current\_value\_pointer, then error checks to make sure it never goes below

562 ; the absolute minimum limit

563 sub\_100

564 movff16 current\_value\_pointer,FSR0

565 banksel temp

566

567 ; Move lower byte of current\_value to temp, then subtract 100 from it and

568 ; move back into current\_value

569 **moviw** 0[FSR0]

570 **movwf** temp

571 **movlw** .100 ; 0x64

572 **subwf** temp,W

573 **movwi** 0[FSR0]

574

575 ; If lower byte underflowed, subtract 1 from upper byte of current\_value

576 **moviw** 1[FSR0]

577 **movwf** temp

578 **movlw** .1

579 **btfss** STATUS,C

580 **subwf** temp,F

581 **movf** temp,W

582 **movwi** 1[FSR0]

583

584 ; Check if result < MIN\_DIST, if so set current value to MIN\_DIST

585 **call** sub\_error\_check ; error check: within limit?

586 **btfsc** WREG,FAIL ; if without limit, set to max value

587 **call** set\_min\_value

588

589 **goto** update\_lcd ; write the new distance to the lcd

590

591 ; Jump to here from goto\_menu when enter button pushed and "sub 10" selected

592 ; Subtracts 10 from the value located at the address stored in

593 ; current\_value\_pointer, then error checks to make sure it never goes below

594 ; the absolute minimum limit

595 sub\_10

596 movff16 current\_value\_pointer,FSR0

597 banksel temp

598

599 ; Move lower byte of current\_value to temp, then subtract 10 from it and

600 ; move back into current\_value

601 **moviw** 0[FSR0]

602 **movwf** temp

603 **movlw** .10 ; 0x64

604 **subwf** temp,W

605 **movwi** 0[FSR0]

606

607 ; If lower byte underflowed, subtract 1 from upper byte of current\_value

608 **moviw** 1[FSR0]

609 **movwf** temp

610 **movlw** .1

611 **btfss** STATUS,C

612 **subwf** temp,F

613 **movf** temp,W

614 **movwi** 1[FSR0]

615

616 ; Check if result < MIN\_DIST, if so set current value to MIN\_DIST

617 **call** sub\_error\_check ; error check: within limit?

618 **btfsc** WREG,FAIL ; if without limit, set to max value

619 **call** set\_min\_value

620

621 **goto** update\_lcd ; write the new distance to the lcd

622

623 ; Jump to here from goto\_menu when enter button pushed and "sub 1" selected

624 ; Subtracts 1 from the value located at the address stored in

625 ; current\_value\_pointer, then error checks to make sure it never goes below

626 ; the absolute minimum limit

627 sub\_1

628 movff16 current\_value\_pointer,FSR0

629 banksel temp

630

631 ; Move lower byte of current\_value to temp, then subtract 1 from it and

632 ; move back into current\_value

633 **moviw** 0[FSR0]

634 **movwf** temp

635 **movlw** .1 ; 0x64

636 **subwf** temp,W

637 **movwi** 0[FSR0]

638

639 ; If lower byte underflowed, subtract 1 from upper byte of current\_value

640 **moviw** 1[FSR0]

641 **movwf** temp

642 **movlw** .1

643 **btfss** STATUS,C

644 **subwf** temp,F

645 **movf** temp,W

646 **movwi** 1[FSR0]

647

648 **call** sub\_error\_check ; error check: within limit?

649 **btfsc** WREG,FAIL ; if without limit, set to max value

650 **call** set\_min\_value

651

652 ; Check if result < MIN\_DIST, if so set current value to MIN\_DIST

653 **goto** update\_lcd ; write the new distance to the lcd

654

655 set\_max\_value

656 movli16 DIST\_MAX,FSR0

657 **return**

658

659 set\_min\_value

660 movli16 DIST\_MIN,FSR0

661 **return**

662

663 ; See if current\_value > DIST\_MAX by seeing if (current\_value - DIST\_MAX)

664 ; underflows and sets the carry bit

665 add\_error\_check

666 **MOVIW** 0[FSR0]

667 **movwf** X

668 **MOVIW** 1[FSR0]

669 **movwf** X+1

670

671 movlf16 DIST\_MAX,Y

672

673 **call** subt16x16

674

675 **btfss** STATUS,C

676 **goto** add\_error\_check\_fail

677

678 **goto** add\_error\_check\_pass

679 add\_error\_check\_fail

680 **retlw** FAIL

681 add\_error\_check\_pass

682 **retlw** PASS

683

684 ; See if current\_value < DIST\_MIN by seeing if (DIST\_MIN - current\_value)

685 ; underflows and sets the carry bit

686 sub\_error\_check

687 **MOVIW** 0[FSR0]

688 **movwf** Y

689 **MOVIW** 1[FSR0]

690 **movwf** Y+1

691

692 movlf16 DIST\_MIN,X

693

694 **call** subt16x16

695

696 **btfss** STATUS,C

697 **goto** sub\_error\_check\_fail

698

699 **goto** sub\_error\_check\_pass

700 sub\_error\_check\_fail

701 **retlw** FAIL

702 sub\_error\_check\_pass

703 **retlw** PASS

704

705 ; Call scan, get result (ie passed, within tolerance, failed, or timeout

706 ; Then choose goto to print appropriate string

707 perform\_scan

708 **call** scan

709

710 **btfsc** usresult,PASS

711 **goto** print\_pass\_string

712

713 **btfsc** usresult,TOLERANCE

714 **goto** print\_tolerance\_string

715

716 **btfsc** usresult,FAIL

717 **goto** print\_fail\_string

718

719 **btfsc** usresult,TIMEOUT

720 **goto** print\_timeout\_string

721

722 ; Prepare for Start Menu to become current menu

723 set\_start

724 ; Reset menu\_selected

725 movlf START\_MENU\_MIN,menu\_selected

726

727 ; Set minimum possible menu

728 movlf START\_MENU\_MIN,menu\_min

729

730 ; Set maximum possible menu

731 movlf START\_MENU\_MAX,menu\_max

732

733 ; Copy location of start menu to menu\_pointer

734 movlf16 start\_menu,menu\_pointer

735

736 ; Copy location of info for printing lcd for the start meny

737 movlf16 update\_lcd\_start\_menu,lcd\_pointer

738

739 **goto** update\_lcd

740

741 ; Prepare for Upper Tolerance Menu to become current menu

742 set\_upper\_tolerance

743 movlf UPPER\_TOLERANCE\_MENU\_MIN,menu\_selected

744 movlf UPPER\_TOLERANCE\_MENU\_MIN,menu\_min

745 movlf UPPER\_TOLERANCE\_MENU\_MAX,menu\_max

746 movlf16 upper\_tolerance,current\_value\_pointer

747 movlf16 upper\_tolerance\_menu,menu\_pointer

748 movlf16 update\_lcd\_upper\_tolerance\_menu,lcd\_pointer

749 **goto** update\_lcd

750

751 ; Prepare for Lower Tolerance Menu to become current menu

752 set\_lower\_tolerance

753 movlf LOWER\_TOLERANCE\_MENU\_MIN,menu\_selected

754 movlf LOWER\_TOLERANCE\_MENU\_MIN,menu\_min

755 movlf LOWER\_TOLERANCE\_MENU\_MAX,menu\_max

756 movlf lower\_tolerance,current\_value\_pointer

757 movlf16 lower\_tolerance\_menu,menu\_pointer

758 movlf16 update\_lcd\_lower\_tolerance\_menu,lcd\_pointer

759 **goto** update\_lcd

760

761 ; Prepare for Upper Passband Menu to become current menu

762 set\_upper\_passband

763 movlf UPPER\_PASSBAND\_MENU\_MIN,menu\_selected

764 movlf UPPER\_PASSBAND\_MENU\_MIN,menu\_min

765 movlf UPPER\_PASSBAND\_MENU\_MAX,menu\_max

766 movlf16 upper\_passband,current\_value\_pointer

767 movlf16 upper\_passband\_menu,menu\_pointer

768 movlf16 update\_lcd\_upper\_passband\_menu,lcd\_pointer

769 **goto** update\_lcd

770

771 ; Prepare for Lower Passband to become current menu

772 set\_lower\_passband

773 movlf LOWER\_PASSBAND\_MENU\_MIN,menu\_selected

774 movlf LOWER\_PASSBAND\_MENU\_MIN,menu\_min

775 movlf LOWER\_PASSBAND\_MENU\_MAX,menu\_max

776 movlf16 lower\_passband,current\_value\_pointer

777 movlf16 lower\_passband\_menu,menu\_pointer

778 movlf16 update\_lcd\_lower\_passband\_menu,lcd\_pointer

779 **goto** update\_lcd

780

781 ; Jump to the appropriate print function for Start Menu

782 update\_lcd\_start\_menu

783 **movf** menu\_selected,W

784 **brw**

785 **NOP** ;error

786 **NOP** ;error

787 **goto** print\_scan\_string

788 **goto** print\_upper\_passband\_string

789 **goto** print\_lower\_passband\_string

790 **goto** print\_upper\_tolerance\_string

791 **goto** print\_lower\_tolerance\_string

792

793 ; Jump to the appropriate print function for Upper Passband Menu

794 update\_lcd\_upper\_passband\_menu

795 **movf** menu\_selected,W

796 **brw**

797 **NOP** ;error

798 **NOP** ;error

799 **goto** print\_add\_100\_string

800 **goto** print\_add\_10\_string

801 **goto** print\_add\_1\_string

802 **goto** print\_sub\_100\_string

803 **goto** print\_sub\_10\_string

804 **goto** print\_sub\_1\_string

805 **goto** print\_start\_string

806

807 ; Jump to the appropriate print function for Lower Passband Menu

808 update\_lcd\_lower\_passband\_menu

809 **movf** menu\_selected,W

810 **brw**

811 **NOP** ;error

812 **NOP** ;error

813 **goto** print\_add\_100\_string

814 **goto** print\_add\_10\_string

815 **goto** print\_add\_1\_string

816 **goto** print\_sub\_100\_string

817 **goto** print\_sub\_10\_string

818 **goto** print\_sub\_1\_string

819 **goto** print\_start\_string

820

821 ; Jump to the appropriate print function for Upper Tolerance Menu

822 update\_lcd\_upper\_tolerance\_menu

823 **movf** menu\_selected,W

824 **brw**

825 **NOP** ;error

826 **NOP** ;error

827 **goto** print\_add\_100\_string

828 **goto** print\_add\_10\_string

829 **goto** print\_add\_1\_string

830 **goto** print\_sub\_100\_string

831 **goto** print\_sub\_10\_string

832 **goto** print\_sub\_1\_string

833 **goto** print\_start\_string

834

835 ; Jump to the appropriate print function for Lower Tolerance Menu

836 update\_lcd\_lower\_tolerance\_menu

837 **movf** menu\_selected,W

838 **brw**

839 **NOP** ;error

840 **NOP** ;error

841 **goto** print\_add\_100\_string

842 **goto** print\_add\_10\_string

843 **goto** print\_add\_1\_string

844 **goto** print\_sub\_100\_string

845 **goto** print\_sub\_10\_string

846 **goto** print\_sub\_1\_string

847 **goto** print\_start\_string

848

849 ; Print "START" to top line of the LCD

850 print\_start\_string

851 **call** lcd\_clear ; clear the lcd

852 **call** set\_lcd\_top\_line ; set lcd to write on top line

853 movlf16 start\_string,FSR0

854 **call** lcd\_print ; write scan\_string to lcd\_print

855 **goto** fin

856

857 ; Print "SCAN" to top line of the LCD

858 print\_scan\_string

859 **call** lcd\_clear ; clear the lcd

860 **call** set\_lcd\_top\_line ; set lcd to write on top line

861 movlf16 scan\_string,FSR0

862 **call** lcd\_print ; write scan\_string to lcd\_print

863 **goto** fin

864

865 ; Print "SCAN" to top line of the LCD, and "PASS: XXXX CM" to bottom line of lcd

866 ; where XXXX is the four-digit result from scan. Sets GREEN\_LED, and clears

867 ; AMBER\_LED and RED\_LED

868 print\_pass\_string

869 **call** hex16\_2\_ascii

870

871 **call** lcd\_clear

872 **call** set\_lcd\_top\_line

873 movlf16 scan\_string, FSR0

874 **call** lcd\_print

875

876 **call** set\_lcd\_bottom\_line

877 movlf16 pass\_string, FSR0

878 **call** lcd\_print

879 movlf16 ascii, FSR0

880 **call** lcd\_print

881 movlf16 mm\_string, FSR0

882 **call** lcd\_print

883

884 banksel LEDPORT

885 **bsf** LEDPORT, GREEN\_LED

886 **bcf** LEDPORT, AMBER\_LED

887 **bcf** LEDPORT, RED\_LED

888

889 clear usresult

890

891 **goto** fin

892

893 ; Print "SCAN" to top line of the LCD, and "FAIL: XXXX CM" to bottom line of lcd

894 ; where XXXX is the four-digit result from scan. Sets AMBER\_LED, and clears

895 ; GREEN\_LED and RED\_LED

896 print\_tolerance\_string

897 **call** hex16\_2\_ascii

898

899 **call** lcd\_clear ; clear the lcd

900 **call** set\_lcd\_top\_line ; set lcd to write on top line

901 movlf16 scan\_string, FSR0

902 **call** lcd\_print

903

904 **call** set\_lcd\_bottom\_line

905 movlf16 fail\_string, FSR0

906 **call** lcd\_print

907 movlf16 ascii, FSR0

908 **call** lcd\_print

909 movlf16 mm\_string, FSR0

910 **call** lcd\_print

911

912 banksel LEDPORT

913 **bcf** LEDPORT, GREEN\_LED

914 **bsf** LEDPORT, AMBER\_LED

915 **bcf** LEDPORT, RED\_LED

916

917 clear usresult

918

919 **goto** fin

920

921 ; Print "SCAN" to top line of the LCD, and "FAIL: XXXX CM" to bottom line of lcd

922 ; where XXXX is the four-digit result from scan. Sets RED\_LED, and clears

923 ; GREEN\_LED and AMBER\_LED

924 print\_fail\_string

925 ; value should already be in X

926 **call** hex16\_2\_ascii

927

928 **call** lcd\_clear ; clear the lcd

929 **call** set\_lcd\_top\_line ; set lcd to write on top line

930 movlf16 scan\_string, FSR0

931 **call** lcd\_print

932

933 **call** set\_lcd\_bottom\_line

934 movlf16 fail\_string, FSR0

935 **call** lcd\_print

936 movlf16 ascii, FSR0

937 **call** lcd\_print

938 movlf16 mm\_string, FSR0

939 **call** lcd\_print

940

941 banksel LEDPORT

942 **bcf** LEDPORT, GREEN\_LED

943 **bcf** LEDPORT, AMBER\_LED

944 **bsf** LEDPORT, RED\_LED

945

946 clear usresult

947

948 **goto** fin

949

950 ; Print "SCAN" to top line of the LCD, and "TIMEOUT" to bottom line of lcd

951 ; Clears all three LEDs

952 print\_timeout\_string

953 **call** lcd\_clear ; clear the lcd

954 **call** set\_lcd\_top\_line ; set lcd to write on top line

955 movlf16 scan\_string, FSR0

956 **call** lcd\_print

957

958 **call** set\_lcd\_bottom\_line

959 movlf16 timeout\_string, FSR0

960 **call** lcd\_print

961

962 banksel LEDPORT

963 **bCf** LEDPORT, GREEN\_LED

964 **bcf** LEDPORT, AMBER\_LED

965 **bsf** LEDPORT, RED\_LED

966

967 **bcf** LEDPORT, GREEN\_LED

968 **bcf** LEDPORT, AMBER\_LED

969 **bcf** LEDPORT, RED\_LED

970

971 clear usresult

972

973 **goto** fin

974

975 ; Print "UPPER PASSBAND" to top line of the LCD

976 print\_upper\_passband\_string

977 **call** lcd\_clear

978 **call** set\_lcd\_top\_line

979 movlf upper\_passband\_string,FSR0

980 **call** lcd\_print

981 **goto** fin

982

983 ; Print "LOWER PASSBAND" to top line of the LCD

984 print\_lower\_passband\_string

985 **call** lcd\_clear

986 **call** set\_lcd\_top\_line

987 movlf16 lower\_passband\_string,FSR0

988 **call** lcd\_print

989 **goto** fin

990

991 ; Print "UPPER TOLERANCE" to top line of the LCD

992 print\_upper\_tolerance\_string

993 **call** lcd\_clear

994 **call** set\_lcd\_top\_line

995 movlf16 upper\_tolerance\_string,FSR0

996 **call** lcd\_print

997 **goto** fin

998

999 ; Print "LOWER TOLERANCE" to top line of the LCD

1000 print\_lower\_tolerance\_string

1001 **call** lcd\_clear

1002 **call** set\_lcd\_top\_line

1003 movlf16 lower\_tolerance\_string,FSR0

1004 **call** lcd\_print

1005 **goto** fin

1006

1007 ; Print "ADD 100" to top line of the LCD, and "XXXX MM" to bottom line of lcd

1008 ; where XXXX is the four-digit value of (current\_value + 100)

1009 print\_add\_100\_string

1010 movff16 current\_value\_pointer,FSR0

1011 moviw16 FSR0,X

1012 **call** hex16\_2\_ascii

1013

1014 **call** lcd\_clear

1015 **call** set\_lcd\_top\_line

1016 movlf16 add\_100\_string,FSR0

1017 **call** lcd\_print

1018

1019 **call** set\_lcd\_bottom\_line

1020 movlf ascii,FSR0

1021 **bcf** FSR0H,7

1022 **call** lcd\_print

1023 movlf mm\_string,FSR0

1024 **call** lcd\_print

1025

1026 **goto** fin

1027

1028 ; Print "ADD 10" to top line of the LCD, and "XXXX MM" to bottom line of lcd

1029 ; where XXXX is the four-digit resut of (current\_value + 10)

1030 print\_add\_10\_string

1031 movff16 current\_value\_pointer,FSR0

1032 moviw16 FSR0,X

1033 **call** hex16\_2\_ascii

1034

1035 **call** lcd\_clear

1036 **call** set\_lcd\_top\_line

1037 movlf16 add\_10\_string,FSR0

1038 **call** lcd\_print

1039

1040 **call** set\_lcd\_bottom\_line

1041 movlf ascii,FSR0

1042 ; FSR0H.7 indicates location is in program memory, clear it just in case

1043 ; it hasn't been cleared, since ascii is in data memory

1044 **bcf** FSR0H,7

1045 **call** lcd\_print

1046 movlf mm\_string,FSR0

1047 **call** lcd\_print

1048

1049 **goto** fin

1050

1051 ; Print "ADD 1" to top line of the LCD, and "XXXX MM" to bottom line of lcd

1052 ; where XXXX is the four-digit value of (current\_value + 1)

1053 print\_add\_1\_string

1054 movff16 current\_value\_pointer,FSR0

1055 moviw16 FSR0,X

1056 **call** hex16\_2\_ascii

1057

1058 **call** lcd\_clear

1059 **call** set\_lcd\_top\_line

1060 movlf16 add\_1\_string,FSR0

1061 **call** lcd\_print

1062

1063 **call** set\_lcd\_bottom\_line

1064 movlf ascii,FSR0

1065 ; FSR0H.7 indicates location is in program memory, clear it just in case

1066 ; it hasn't been cleared, since ascii is in data memory

1067 **bcf** FSR0H,7

1068 **call** lcd\_print

1069 movlf mm\_string,FSR0

1070 **call** lcd\_print

1071

1072 **goto** fin

1073

1074 ; Print "SUB 100" to top line of the LCD, and "XXXX MM" to bottom line of lcd

1075 ; where XXXX is the four-digit value of (current\_value - 1)

1076 print\_sub\_100\_string

1077 movff16 current\_value\_pointer,FSR0

1078 moviw16 FSR0,X

1079 **call** hex16\_2\_ascii

1080

1081 **call** lcd\_clear

1082 **call** set\_lcd\_top\_line

1083 movlf16 sub\_100\_string,FSR0

1084 **call** lcd\_print

1085

1086 **call** set\_lcd\_bottom\_line

1087 movlf ascii,FSR0

1088 ; FSR0H.7 indicates location is in program memory, clear it just in case

1089 ; it hasn't been cleared, since ascii is in data memory

1090 **bcf** FSR0H,7

1091 **call** lcd\_print

1092 movlf mm\_string,FSR0

1093 **call** lcd\_print

1094

1095 **goto** fin

1096

1097 ; Print "SUB 10" to top line of the LCD, and "XXXX MM" to bottom line of lcd

1098 ; where XXXX is the four-digit value of (current\_value - 10)

1099 print\_sub\_10\_string

1100 movff16 current\_value\_pointer,FSR0

1101 moviw16 FSR0,X

1102 **call** hex16\_2\_ascii

1103

1104 **call** lcd\_clear

1105 **call** set\_lcd\_top\_line

1106 movlf16 sub\_10\_string,FSR0

1107 **call** lcd\_print

1108

1109 **call** set\_lcd\_bottom\_line

1110 movlf ascii,FSR0

1111 ; FSR0H.7 indicates location is in program memory, clear it just in case

1112 ; it hasn't been cleared, since ascii is in data memory

1113 **bcf** FSR0H,7

1114 **call** lcd\_print

1115 movlf mm\_string,FSR0

1116 **call** lcd\_print

1117

1118 **goto** fin

1119

1120 ; Print "SUB 1" to top line of the LCD, and "XXXX MM" to bottom line of lcd

1121 ; where XXXX is the four-digit value of (current\_value - 1)

1122 print\_sub\_1\_string

1123 movff16 current\_value\_pointer,FSR0

1124 moviw16 FSR0,X

1125 **call** hex16\_2\_ascii

1126

1127 **call** lcd\_clear

1128 **call** set\_lcd\_top\_line

1129 movlf16 sub\_1\_string,FSR0

1130 **call** lcd\_print

1131

1132 **call** set\_lcd\_bottom\_line

1133 movlf ascii,FSR0

1134 ; FSR0H.7 indicates location is in program memory, clear it just in case

1135 ; it hasn't been cleared, since ascii is in data memory

1136 **bcf** FSR0H,7

1137 **call** lcd\_print

1138 movlf mm\_string,FSR0

1139 **call** lcd\_print

1140

1141 **goto** fin

1142

1143 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1144 ; SCAN

1145 ;

1146 ; Gets a result from the Polaroid 6500, in units of microseconds

1147 ; Processes it, and then returns a value indicating whether passed, failed,

1148 ; within tolerance or timed out

1149 ;

1150 ; Starts with a 16-bit result from the Polaroid 6500 in microseconds

1151 ;

1152 ; The result is processed as follows:

1153 ;

1154 ; 1. (((result from Polaroid 6500) - PULSEGEN) \* MM\_PER\_US)/1024) - LATENCY

1155 ; 2. The above is then compared to passband, if within the condition is PASS

1156 ; 3. Then compared to tolerance, if within condition is TOLERANCE

1157 ; 4. Must be outside passband or tolerance, therefore condition is FAIL

1158 ;

1159 ; If the PIC doesn't detect a reponse from ECHO before TMR1 overflow, then

1160 ; condition is TIMEOUT

1161 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1162 scan

1163 **call** ultrasound

1164

1165 banksel usresult

1166 **btfsc** usresult,PASS

1167 **goto** scan\_pass

1168 **btfsc** usresult,FAIL

1169 **goto** scan\_timeout

1170 scan\_pass clear usresult

1171 ; Recover result from CCPR1

1172 movff CCPR1L,X

1173 movff CCPR1H,X+1

1174 ;movlf16 0x1D60,X

1175 ; subtract ultrasound clock generation delay

1176 movlf16 PULSEGEN,Y

1177 **call** subt16x16

1178 ; multiply by (mm per instruction clock cycle)\*1024

1179 movff result,X

1180 movff result+1,X+1

1181 movlf16 MM\_PER\_US,Y

1182 **call** mult16x16 ;

1183 ; divide by 1024 to get (mm per instruction clock cycle)

1184 movlf .10,Y

1185 **call** div40bin

1186 ; subtract 20cm from result, fiddle factor

1187 movff16 result,X

1188 movlf16 LATENCY,Y

1189 **call** subt16x16

1190 ; check if the result is less than the lower limit

1191 movff16 result,X ; expect a 16 bit result so this is ok

1192 movff16 lower\_passband,Y

1193 **call** subt16x16

1194 **btfss** STATUS,C ;|\ If Z set, then either X = Y, or

1195 **goto** below\_passband ;// X > Y If Z clear, then X < Y

1196 **movwf** temp ;

1197 ; check if result is greater than the upper limit

1198 movff16 upper\_passband,Y

1199 **call** subt16x16

1200 **btfsc** STATUS,C

1201 **goto** above\_passband

1202 ; if you get here then hooray, the value is in the passband

1203 **bsf** usresult,PASS

1204 **goto** scan\_end

1205

1206 below\_passband

1207 ; test if within tolerance

1208 movff16 lower\_tolerance,Y

1209 **call** subt16x16

1210 **btfss** STATUS,C

1211 **goto** scan\_fail

1212 ; if you get here then the value is within tolerance

1213 **bsf** usresult, TOLERANCE

1214 **goto** scan\_end

1215

1216 above\_passband

1217 movff16 upper\_tolerance,Y

1218 **call** subt16x16

1219 **btfsc** STATUS,C

1220 **goto** scan\_fail

1221 ; test if within tolerance

1222 ; if you get here then within tolerance

1223 **bsf** usresult, TOLERANCE

1224 **goto** scan\_end

1225

1226 scan\_fail

1227 **bsf** usresult, FAIL

1228 **goto** scan\_end

1229 scan\_timeout

1230 clear usresult

1231 **bsf** usresult,TIMEOUT

1232 **goto** scan\_end

1233 scan\_end

1234 **return**

1235

1236 ; Communicate with Polaroid 6500 and get a result in terms of time

1237 ultrasound

1238 ; Clear GIE because we don't want to use the ISR. Enable PEIE so

1239 ; that Interrupt Flags can operate

1240 **clrf** INTCON

1241 **bcf** INTCON,GIE

1242 **bsf** INTCON,PEIE

1243 ; Set TMR1IE and CCP1IE because we will use these modules

1244 banksel PIE1

1245 **bsf** PIE1,TMR1IE

1246 **bsf** PIE1,CCP1IE

1247 ; Use instruction clock as Timer 1 input, 1MHz -> 1us per

1248 ; increment

1249 movlf 0x05,T1CON

1250 ; Set INIT to begin scan

1251 banksel PORTB

1252 **bsf** PORTB,INIT

1253 ; Clean TMR1 out just in case it isn't empty

1254 banksel TMR1H

1255 **clrf** TMR1H

1256 **clrf** TMR1L

1257 **bcf** INTCON,TMR0IF

1258 ; Reset TMR0 configuration

1259 banksel OPTION\_REG

1260 **bsf** OPTION\_REG,TMR0CS

1261 movlf 0x05,CCP1CON

1262 ; Poll IFs in infinite loop until we have a result or timeout

1263 banksel PIR1

1264 wait\_loop

1265 ; If TMR1IF set, then overflow ergo ultrasound must not be

1266 ; responding

1267 **btfsc** PIR1,TMR1IF

1268 **goto** timeout

1269 ; If CCP1IF set, then we must have got a result from the

1270 ; Polaroid 6500

1271 **btfsc** PIR1,CCP1IF

1272 **goto** success

1273 ; If neither IF's are set, loop back until one is

1274 **goto** wait\_loop

1275 timeout

1276 banksel usresult

1277 **bsf** usresult,FAIL

1278 **goto** got\_result

1279 success

1280 banksel usresult

1281 **bsf** usresult,PASS

1282 **goto** got\_result

1283

1284 got\_result

1285 ; Clear INIT to stop the Polaroid 6500 repeating the test

1286 banksel PORTB

1287 **bcf** PORTB,INIT

1288 ; Switch peripheral interrupts off because we don't need them

1289 ; right now

1290 **bcf** INTCON,PEIE

1291 ; Disable CCP1, TMR1, and PIR1

1292 clear CCP1CON

1293 clear T1CON

1294 clear PIR1

1295 ; Clear Interrupt Flags

1296 clear PIE1

1297 ; Set TMR0 to overflow after 80ms elapsed

1298 movlf 0x12,OSCCON

1299 **bcf** INTCON,TMR0IF

1300 movlf 0x62,TMR0

1301 ; Set TMR0 up with a prescaler of 4 and to use instruction clock

1302 ; (1Mhz)

1303 movlf 0x01,OPTION\_REG

1304 ults\_end

1305 ; Wait until 80ms elapsed before moving on

1306 **btfss** INTCON,TMR0IF

1307 **goto** $-1

1308 ; Disable TMR0

1309 **bcf** INTCON,TMR0IF

1310 ; Reset TMR0 configuration

1311 banksel OPTION\_REG

1312 **bsf** OPTION\_REG,TMR0CS

1313 ; Reset System Clock to 4MHz

1314 movlf 0xEB,OSCCON

1315 **return**

1316

1317 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1318 ; MATHS FUNCTIONS

1319 ;

1320 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1321

1322 ; Multiply together 16-bit integer X, and 16-bit integer Y and place into 40-bit

1323 ; Result register

1324 mulxy MACRO byte,bit

1325 **btfsc** byte,bit

1326 **call** add\_y

1327 **rrf** result+4,F

1328 **rrf** result+3,F

1329 **rrf** result+2,F

1330 **rrf** result+1,F

1331 **rrf** result,F

1332 ENDM

1333 mult16x16

1334 **bcf** STATUS,C ; Clear Carry bit before we use it

1335 banksel X

1336 **clrf** result+4 ; \ Clear registers that hold result

1337 **clrf** result+3 ; |

1338 **clrf** result+2 ; |

1339 **clrf** result+1 ; |

1340 **clrf** result ; /

1341 mulxy X,0 ; \ Perform long multiplication on X and Y, make the

1342 mulxy X,1 ; |

1343 mulxy X,2 ; |

1344 mulxy X,3 ; |

1345 mulxy X,4 ; |

1346 mulxy X,5 ; |

1347 mulxy X,6 ; |

1348 mulxy X,7 ; |

1349 mulxy X+1,0 ; |

1350 mulxy X+1,1 ; |

1351 mulxy X+1,2 ; |

1352 mulxy X+1,3 ; |

1353 mulxy X+1,4 ; |

1354 mulxy X+1,5 ; |

1355 mulxy X+1,6 ; |

1356 mulxy X+1,7 ; /

1357 **return**

1358 add\_y

1359 banksel Y

1360 **movf** Y,W ;\ Add least significant byte of Y to 2nd most

1361 **addwf** result+2,F ;/ sig. byte of result, if overflow set carry bit

1362 **movf** Y+1,W ;\ Add most significant byte of Y and C to most

1363 **addwfc** result+3,F ;/ sig. byte of result, if overflow set carry bit

1364 **btfss** STATUS,C ;\ Carry bit set -> result+3 oflows into result+4

1365 **goto** fin\_y ;| .'. add carry bit to result+4

1366 **movlw** 0x00 ;|

1367 **addwfc** result+4 ;/

1368 fin\_y **return**

1369

1370 ; Add 16-bit integer Y to 16-bit integer X, and place into result register

1371 add16x16

1372 **movf** Y,W ;\ Add lower byte of Y to lower byte of X

1373 **addwf** X,W ;|

1374 **movwf** result ;/

1375 **movf** Y+1,W ;\ Add upper byte of Y to upper byte of X

1376 **addwfc** X+1,W ;|

1377 **movwf** result+1 ;/

1378 **movwf** 0x00 ;\ Add carry to overflow

1379 **addwfc** result+2,F ;/

1380 **return** ;

1381

1382 ; Subtract a 16-bit integer Y from 16-bit integer X, place into result register

1383 subt16x16

1384 banksel Y

1385 **movf** Y,W

1386 **subwf** X,W

1387 **movwf** result

1388 ;

1389 **movf** Y+1,W

1390 **btfss** STATUS,C

1391 **addlw** 0x01

1392 **subwf** X+1,W

1393 **movwf** result+1

1394 ;

1395 **return**

1396

1397 ; Divide 40-bit wide result register by 1024 by bit shifting 10 places

1398 div40bin

1399 div **bcf** STATUS,C ;|

1400 **rrf** result+4,f ;|\ Shift right from MSreg to LSreg

1401 **rrf** result+3,f ;||

1402 **rrf** result+2,f ;||

1403 **rrf** result+1,f ;||

1404 **rrf** result,f ;|/

1405 **decfsz** Y,f ;| Decrement count, exit loop when zero

1406 **goto** div ;/

1407 **return**

1408

1409 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1410 ; LCD FUNCTIONS

1411 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1412

1413 lcd\_setup

1414 clear ANSELA

1415 clear LCDDIR

1416 clear LCDPORT

1417 ; Set WDT for 1s

1418 movlf 0x15,WDTCON

1419

1420 **sleep**

1421

1422 clear WDTCON

1423

1424 movlf .10,deltime

1425

1426 **movlw** 0x02 ;Set 4 bit mode

1427 **call** enable

1428 **call** delay

1429

1430 **movlw** 0x02 ;Set 4 bit mode

1431 **call** enable

1432 **call** delay

1433

1434 **movlw** 0x08 ;Set 2 line

1435 **call** enable

1436 **call** delay

1437

1438 movlf 0x0F, cmdbyte ;Display on, cursor on, blinking on

1439 **call** lcdcmd

1440

1441 movlf 0x06, cmdbyte ;Entry mode set.

1442 **call** lcdcmd

1443

1444 **return**

1445

1446 lcdbusy

1447 banksel LCDDIR

1448 **bsf** LCDDIR,DB7

1449 clear LCDPORT

1450 **bsf** LCDPORT,RW

1451

1452 lcdbsy1 **bsf** LCDPORT,EN

1453 **nop**

1454 **movf** LCDPORT,w

1455 **bcf** LCDPORT,EN

1456 **nop**

1457

1458 **bsf** LCDPORT,EN

1459 **nop**

1460 **bcf** LCDPORT,EN

1461 **nop**

1462

1463 **btfsc** WREG,DB7

1464 **goto** lcdbsy1

1465

1466 clear LCDDIR

1467 **return**

1468

1469 lcdcmd

1470 **swapf** cmdbyte,w ;\

1471 **andlw** LCDMASK ; | Output the LCD command stored in cmdbyte

1472 **call** enable ; |

1473

1474 **movf** cmdbyte,w ; | Outputs high nibble then low nibble.

1475 **andlw** LCDMASK ; |

1476 **call** enable ; |

1477 ; |

1478 **call** lcdbusy ; |

1479 ; |

1480 **return** ;/

1481

1482 set\_lcd\_top\_line

1483 movlf 0x80,cmdbyte ; Move cursor to line 1

1484 **call** lcdcmd

1485 **return**

1486

1487 ; Move cursor to line 2

1488 set\_lcd\_bottom\_line

1489

1490 movlf 0xC0, cmdbyte

1491 **call** lcdcmd

1492 **return**

1493

1494 ; Clear screen

1495 lcd\_clear

1496 movlf 0x01, cmdbyte

1497 **call** lcdcmd

1498 **return**

1499

1500 ; Turn screen off

1501 lcdoff

1502 movlf 0x08, cmdbyte

1503 **call** lcdcmd

1504 **return**

1505

1506 ; Turn Screen on

1507 lcdon

1508 movlf 0x0C, cmdbyte

1509 **call** lcdcmd

1510 **return**

1511

1512 ; Print string to LCD

1513 lcd\_print

1514 **moviw** 0[FSR0]

1515 **andlw** 0xFF

1516 **btfsc** STATUS,Z ;

1517 **return** ;

1518 **swapf** WREG,w

1519 **andlw** LCDMASK ; Move first string byte from FSR, if zero

1520 **bsf** WREG,RS ; (null), return as end of string reached.

1521 **call** enable

1522

1523 **moviw** FSR0++ ; Send high then low nibble of character

1524 **andlw** LCDMASK ; incrementing FSR then call 'lcdbusy'.

1525 **bsf** WREG,RS

1526 **call** enable

1527

1528 **call** lcdbusy

1529

1530 **goto** lcd\_print ; Will loop until end of string reached.

1531

1532 enable

1533 banksel LCDPORT ;\

1534 **movwf** LCDPORT ; | Loads nibble onto LCD data lines, sets

1535 **bsf** LCDPORT,EN ; | the enable line then clears the LCD

1536 **nop** ; | port.

1537 **clrf** LCDPORT ; |

1538 **return** ;/

1539

1540 delay **movf** deltime,w ;\

1541 **goto** deljmp ; | Should be a 10us delay each loop, uses

1542 delloop **nop** ; | number stored in 'deltime' for iteration

1543 **nop** ; | count.

1544 **nop** ; |

1545 **nop** ; | First loop shorter due to instruction

1546 **nop** ; | times calling function.

1547 **nop** ; |

1548 deljmp **nop** ; | 10us only valid for 4MHz clock.

1549 **decfsz** WREG,w ; |

1550 **goto** delloop ; |

1551 **return** ;/

1552

1553 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1554 ; HEX16 TO ASCII

1555 ;

1556 ; Convert a 16-bit register X:2 into a 4 radix wide denary number, ascii:5

1557 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1558 hex16\_2\_ascii

1559 **call** hex16\_2\_bcd

1560 get\_ascii\_un BCD+1, ascii

1561 get\_ascii\_ln BCD+1, ascii+1

1562 get\_ascii\_un BCD+2, ascii+2

1563 get\_ascii\_ln BCD+2, ascii+3

1564 movlf "\0", ascii+4

1565

1566 **return**

1567

1568 hex16\_2\_bcd

1569 **bcf** STATUS, C

1570 movlf .16, count

1571 clear BCD

1572 clear BCD+1

1573 clear BCD+2

1574 clear BCD+3

1575 hex16\_2\_bcd\_loop

1576 banksel X

1577 **rlf** X,f

1578 **rlf** X+1,f

1579

1580 banksel BCD

1581 **rlf** BCD+2,f

1582 **rlf** BCD+1,f

1583 **rlf** BCD,f

1584

1585 banksel count

1586 **decfsz** count,f

1587 **goto** adjDEC

1588

1589 **return**

1590

1591 adjDEC

1592 movlf BCD+2,FSR1

1593 **call** adjBCD

1594

1595 movlf BCD+1,FSR1

1596 **call** adjBCD

1597

1598 movlf BCD,FSR1

1599 **call** adjBCD

1600

1601 **goto** hex16\_2\_bcd\_loop

1602

1603 adjBCD

1604 **movf** INDF1,w

1605 **addlw** 0x03

1606 **btfsc** WREG,3

1607 **movwi** 0[FSR1]

1608

1609 **moviw** 0[FSR1]

1610 **addlw** 0x30

1611 **btfsc** WREG,7

1612 **movwi** 0[FSR1]

1613

1614 **return**

1615

1616 ; get\_ascii

1617 ; Returns the ASCII equivalent of a single BCD number

1618 ; Uses the least significant nibble

1619 get\_ascii

1620 **brw**

1621 **retlw** "0"

1622 **retlw** "1"

1623 **retlw** "2"

1624 **retlw** "3"

1625 **retlw** "4"

1626 **retlw** "5"

1627 **retlw** "6"

1628 **retlw** "7"

1629 **retlw** "8"

1630 **retlw** "9"

1631

1632 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1633 ; SOFTWARE ERROR

1634 ; You don't want to end up here

1635 ; Sets all three LEDs ON

1636 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1637

1638 software\_error

1639 BANKSEL TRISB

1640 **bsf** TRISB,GREEN\_LED

1641 **bsf** TRISB,AMBER\_LED

1642 **bsf** TRISB,RED\_LED

1643 **goto** $

1644

1645 end

1646

# Appendix C: .map File

MPLINK 5.01, LINKER

Linker Map File - Created Wed Jan 28 23:12:04 2015

Section Info

Section Type Address Location Size(Bytes)

--------- --------- --------- --------- ---------

.org\_0 code 0x000000 program 0x000002

.cinit romdata 0x000001 program 0x000004

.org\_1 code 0x000004 program 0x000c7a

Program Memory Usage

Start End

--------- ---------

0x000000 0x000002

0x000004 0x000640

1600 out of 8455 program addresses used, program memory utilization is 18%

Symbols - Sorted by Name

Name Address Location Storage File

--------- --------- --------- --------- ---------

ISR 0x000004 program static D:\en0627\_assignment.X\en0627\_assignment.asm

TMR0\_reset 0x0000df program static D:\en0627\_assignment.X\en0627\_assignment.asm

\_.org\_1\_00B7 0x0000b7 program static D:\en0627\_assignment.X\en0627\_assignment.asm

\_.org\_1\_04CC 0x0004cc program static D:\en0627\_assignment.X\en0627\_assignment.asm

\_.org\_1\_0640 0x000640 program static D:\en0627\_assignment.X\en0627\_assignment.asm

above\_passband 0x000482 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add16x16 0x000556 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_1 0x000168 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_10 0x000155 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_100 0x000142 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_100\_string 0x000069 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_10\_string 0x000071 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_1\_string 0x000078 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_error\_check 0x0001d0 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_error\_check\_fail 0x0001dd program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_error\_check\_pass 0x0001de program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_y 0x00054c program static D:\en0627\_assignment.X\en0627\_assignment.asm

adjBCD 0x000628 program static D:\en0627\_assignment.X\en0627\_assignment.asm

adjDEC 0x00061b program static D:\en0627\_assignment.X\en0627\_assignment.asm

below\_passband 0x000475 program static D:\en0627\_assignment.X\en0627\_assignment.asm

cycle\_left 0x0000eb program static D:\en0627\_assignment.X\en0627\_assignment.asm

cycle\_right 0x0000ef program static D:\en0627\_assignment.X\en0627\_assignment.asm

delay 0x0005db program static D:\en0627\_assignment.X\en0627\_assignment.asm

deljmp 0x0005e3 program static D:\en0627\_assignment.X\en0627\_assignment.asm

delloop 0x0005dd program static D:\en0627\_assignment.X\en0627\_assignment.asm

div 0x000569 program static D:\en0627\_assignment.X\en0627\_assignment.asm

div40bin 0x000569 program static D:\en0627\_assignment.X\en0627\_assignment.asm

enable 0x0005d5 program static D:\en0627\_assignment.X\en0627\_assignment.asm

error\_check\_lower 0x0000fe program static D:\en0627\_assignment.X\en0627\_assignment.asm

error\_check\_upper 0x000105 program static D:\en0627\_assignment.X\en0627\_assignment.asm

fail\_string 0x00004f program static D:\en0627\_assignment.X\en0627\_assignment.asm

fin 0x0000fb program static D:\en0627\_assignment.X\en0627\_assignment.asm

fin\_y 0x000555 program static D:\en0627\_assignment.X\en0627\_assignment.asm

get\_ascii 0x000631 program static D:\en0627\_assignment.X\en0627\_assignment.asm

got\_result 0x0004b6 program static D:\en0627\_assignment.X\en0627\_assignment.asm

goto\_menu 0x0000e1 program static D:\en0627\_assignment.X\en0627\_assignment.asm

hex16\_2\_ascii 0x0005e7 program static D:\en0627\_assignment.X\en0627\_assignment.asm

hex16\_2\_bcd 0x000604 program static D:\en0627\_assignment.X\en0627\_assignment.asm

hex16\_2\_bcd\_loop 0x000610 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcd\_clear 0x0005b8 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcd\_print 0x0005c7 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcd\_setup 0x000572 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcdbsy1 0x000598 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcdbusy 0x000593 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcdcmd 0x0005a6 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcdoff 0x0005bd program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcdon 0x0005c2 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lower\_passband\_menu 0x000121 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lower\_passband\_string 0x000028 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lower\_tolerance\_menu 0x000137 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lower\_tolerance\_string 0x00003e program static D:\en0627\_assignment.X\en0627\_assignment.asm

main 0x0000d1 program static D:\en0627\_assignment.X\en0627\_assignment.asm

mm\_string 0x000065 program static D:\en0627\_assignment.X\en0627\_assignment.asm

mult16x16 0x0004d4 program static D:\en0627\_assignment.X\en0627\_assignment.asm

pass\_string 0x000048 program static D:\en0627\_assignment.X\en0627\_assignment.asm

perform\_scan 0x0001ee program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_add\_100\_string 0x000358 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_add\_10\_string 0x000379 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_add\_1\_string 0x00039a program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_fail\_string 0x0002fa program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_lower\_passband\_string 0x00033d program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_lower\_tolerance\_string 0x00034f program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_pass\_string 0x0002b4 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_scan\_string 0x0002ab program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_start\_string 0x0002a2 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_sub\_100\_string 0x0003bb program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_sub\_10\_string 0x0003dc program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_sub\_1\_string 0x0003fd program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_timeout\_string 0x00031d program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_tolerance\_string 0x0002d7 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_upper\_passband\_string 0x000336 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_upper\_tolerance\_string 0x000346 program static D:\en0627\_assignment.X\en0627\_assignment.asm

scan 0x00041e program static D:\en0627\_assignment.X\en0627\_assignment.asm

scan\_end 0x000495 program static D:\en0627\_assignment.X\en0627\_assignment.asm

scan\_fail 0x00048f program static D:\en0627\_assignment.X\en0627\_assignment.asm

scan\_pass 0x000424 program static D:\en0627\_assignment.X\en0627\_assignment.asm

scan\_string 0x00000f program static D:\en0627\_assignment.X\en0627\_assignment.asm

scan\_timeout 0x000491 program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_lcd\_bottom\_line 0x0005b3 program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_lcd\_top\_line 0x0005ae program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_lower\_passband 0x000254 program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_lower\_tolerance 0x000224 program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_max\_value 0x0001c6 program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_min\_value 0x0001cb program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_start 0x0001f7 program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_upper\_passband 0x00023b program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_upper\_tolerance 0x00020b program static D:\en0627\_assignment.X\en0627\_assignment.asm

setup 0x000093 program static D:\en0627\_assignment.X\en0627\_assignment.asm

software\_error 0x00063c program static D:\en0627\_assignment.X\en0627\_assignment.asm

start\_menu 0x00010d program static D:\en0627\_assignment.X\en0627\_assignment.asm

start\_string 0x000004 program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_1 0x0001ad program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_10 0x000194 program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_100 0x00017b program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_100\_string 0x00007e program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_10\_string 0x000086 program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_1\_string 0x00008d program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_error\_check 0x0001df program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_error\_check\_fail 0x0001ec program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_error\_check\_pass 0x0001ed program static D:\en0627\_assignment.X\en0627\_assignment.asm

subt16x16 0x00055f program static D:\en0627\_assignment.X\en0627\_assignment.asm

success 0x0004b3 program static D:\en0627\_assignment.X\en0627\_assignment.asm

testing\_string 0x000014 program static D:\en0627\_assignment.X\en0627\_assignment.asm

timeout 0x0004b0 program static D:\en0627\_assignment.X\en0627\_assignment.asm

timeout\_string 0x000056 program static D:\en0627\_assignment.X\en0627\_assignment.asm

ultrasound 0x000496 program static D:\en0627\_assignment.X\en0627\_assignment.asm

ults\_end 0x0004cb program static D:\en0627\_assignment.X\en0627\_assignment.asm

update\_lcd 0x0000f3 program static D:\en0627\_assignment.X\en0627\_assignment.asm

update\_lcd\_lower\_passband\_menu 0x000281 program static D:\en0627\_assignment.X\en0627\_assignment.asm

update\_lcd\_lower\_tolerance\_menu 0x000297 program static D:\en0627\_assignment.X\en0627\_assignment.asm

update\_lcd\_start\_menu 0x00026d program static D:\en0627\_assignment.X\en0627\_assignment.asm

update\_lcd\_upper\_passband\_menu 0x000276 program static D:\en0627\_assignment.X\en0627\_assignment.asm

update\_lcd\_upper\_tolerance\_menu 0x00028c program static D:\en0627\_assignment.X\en0627\_assignment.asm

upper\_passband\_menu 0x000116 program static D:\en0627\_assignment.X\en0627\_assignment.asm

upper\_passband\_string 0x00001c program static D:\en0627\_assignment.X\en0627\_assignment.asm

upper\_tolerance\_menu 0x00012c program static D:\en0627\_assignment.X\en0627\_assignment.asm

upper\_tolerance\_string 0x000034 program static D:\en0627\_assignment.X\en0627\_assignment.asm

wait\_loop 0x0004ab program static D:\en0627\_assignment.X\en0627\_assignment.asm

Symbols - Sorted by Address

Name Address Location Storage File

--------- --------- --------- --------- ---------

ISR 0x000004 program static D:\en0627\_assignment.X\en0627\_assignment.asm

start\_string 0x000004 program static D:\en0627\_assignment.X\en0627\_assignment.asm

scan\_string 0x00000f program static D:\en0627\_assignment.X\en0627\_assignment.asm

testing\_string 0x000014 program static D:\en0627\_assignment.X\en0627\_assignment.asm

upper\_passband\_string 0x00001c program static D:\en0627\_assignment.X\en0627\_assignment.asm

lower\_passband\_string 0x000028 program static D:\en0627\_assignment.X\en0627\_assignment.asm

upper\_tolerance\_string 0x000034 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lower\_tolerance\_string 0x00003e program static D:\en0627\_assignment.X\en0627\_assignment.asm

pass\_string 0x000048 program static D:\en0627\_assignment.X\en0627\_assignment.asm

fail\_string 0x00004f program static D:\en0627\_assignment.X\en0627\_assignment.asm

timeout\_string 0x000056 program static D:\en0627\_assignment.X\en0627\_assignment.asm

mm\_string 0x000065 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_100\_string 0x000069 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_10\_string 0x000071 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_1\_string 0x000078 program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_100\_string 0x00007e program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_10\_string 0x000086 program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_1\_string 0x00008d program static D:\en0627\_assignment.X\en0627\_assignment.asm

setup 0x000093 program static D:\en0627\_assignment.X\en0627\_assignment.asm

\_.org\_1\_00B7 0x0000b7 program static D:\en0627\_assignment.X\en0627\_assignment.asm

main 0x0000d1 program static D:\en0627\_assignment.X\en0627\_assignment.asm

TMR0\_reset 0x0000df program static D:\en0627\_assignment.X\en0627\_assignment.asm

goto\_menu 0x0000e1 program static D:\en0627\_assignment.X\en0627\_assignment.asm

cycle\_left 0x0000eb program static D:\en0627\_assignment.X\en0627\_assignment.asm

cycle\_right 0x0000ef program static D:\en0627\_assignment.X\en0627\_assignment.asm

update\_lcd 0x0000f3 program static D:\en0627\_assignment.X\en0627\_assignment.asm

fin 0x0000fb program static D:\en0627\_assignment.X\en0627\_assignment.asm

error\_check\_lower 0x0000fe program static D:\en0627\_assignment.X\en0627\_assignment.asm

error\_check\_upper 0x000105 program static D:\en0627\_assignment.X\en0627\_assignment.asm

start\_menu 0x00010d program static D:\en0627\_assignment.X\en0627\_assignment.asm

upper\_passband\_menu 0x000116 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lower\_passband\_menu 0x000121 program static D:\en0627\_assignment.X\en0627\_assignment.asm

upper\_tolerance\_menu 0x00012c program static D:\en0627\_assignment.X\en0627\_assignment.asm

lower\_tolerance\_menu 0x000137 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_100 0x000142 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_10 0x000155 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_1 0x000168 program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_100 0x00017b program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_10 0x000194 program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_1 0x0001ad program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_max\_value 0x0001c6 program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_min\_value 0x0001cb program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_error\_check 0x0001d0 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_error\_check\_fail 0x0001dd program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_error\_check\_pass 0x0001de program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_error\_check 0x0001df program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_error\_check\_fail 0x0001ec program static D:\en0627\_assignment.X\en0627\_assignment.asm

sub\_error\_check\_pass 0x0001ed program static D:\en0627\_assignment.X\en0627\_assignment.asm

perform\_scan 0x0001ee program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_start 0x0001f7 program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_upper\_tolerance 0x00020b program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_lower\_tolerance 0x000224 program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_upper\_passband 0x00023b program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_lower\_passband 0x000254 program static D:\en0627\_assignment.X\en0627\_assignment.asm

update\_lcd\_start\_menu 0x00026d program static D:\en0627\_assignment.X\en0627\_assignment.asm

update\_lcd\_upper\_passband\_menu 0x000276 program static D:\en0627\_assignment.X\en0627\_assignment.asm

update\_lcd\_lower\_passband\_menu 0x000281 program static D:\en0627\_assignment.X\en0627\_assignment.asm

update\_lcd\_upper\_tolerance\_menu 0x00028c program static D:\en0627\_assignment.X\en0627\_assignment.asm

update\_lcd\_lower\_tolerance\_menu 0x000297 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_start\_string 0x0002a2 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_scan\_string 0x0002ab program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_pass\_string 0x0002b4 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_tolerance\_string 0x0002d7 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_fail\_string 0x0002fa program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_timeout\_string 0x00031d program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_upper\_passband\_string 0x000336 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_lower\_passband\_string 0x00033d program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_upper\_tolerance\_string 0x000346 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_lower\_tolerance\_string 0x00034f program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_add\_100\_string 0x000358 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_add\_10\_string 0x000379 program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_add\_1\_string 0x00039a program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_sub\_100\_string 0x0003bb program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_sub\_10\_string 0x0003dc program static D:\en0627\_assignment.X\en0627\_assignment.asm

print\_sub\_1\_string 0x0003fd program static D:\en0627\_assignment.X\en0627\_assignment.asm

scan 0x00041e program static D:\en0627\_assignment.X\en0627\_assignment.asm

scan\_pass 0x000424 program static D:\en0627\_assignment.X\en0627\_assignment.asm

below\_passband 0x000475 program static D:\en0627\_assignment.X\en0627\_assignment.asm

above\_passband 0x000482 program static D:\en0627\_assignment.X\en0627\_assignment.asm

scan\_fail 0x00048f program static D:\en0627\_assignment.X\en0627\_assignment.asm

scan\_timeout 0x000491 program static D:\en0627\_assignment.X\en0627\_assignment.asm

scan\_end 0x000495 program static D:\en0627\_assignment.X\en0627\_assignment.asm

ultrasound 0x000496 program static D:\en0627\_assignment.X\en0627\_assignment.asm

wait\_loop 0x0004ab program static D:\en0627\_assignment.X\en0627\_assignment.asm

timeout 0x0004b0 program static D:\en0627\_assignment.X\en0627\_assignment.asm

success 0x0004b3 program static D:\en0627\_assignment.X\en0627\_assignment.asm

got\_result 0x0004b6 program static D:\en0627\_assignment.X\en0627\_assignment.asm

ults\_end 0x0004cb program static D:\en0627\_assignment.X\en0627\_assignment.asm

\_.org\_1\_04CC 0x0004cc program static D:\en0627\_assignment.X\en0627\_assignment.asm

mult16x16 0x0004d4 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add\_y 0x00054c program static D:\en0627\_assignment.X\en0627\_assignment.asm

fin\_y 0x000555 program static D:\en0627\_assignment.X\en0627\_assignment.asm

add16x16 0x000556 program static D:\en0627\_assignment.X\en0627\_assignment.asm

subt16x16 0x00055f program static D:\en0627\_assignment.X\en0627\_assignment.asm

div 0x000569 program static D:\en0627\_assignment.X\en0627\_assignment.asm

div40bin 0x000569 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcd\_setup 0x000572 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcdbusy 0x000593 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcdbsy1 0x000598 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcdcmd 0x0005a6 program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_lcd\_top\_line 0x0005ae program static D:\en0627\_assignment.X\en0627\_assignment.asm

set\_lcd\_bottom\_line 0x0005b3 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcd\_clear 0x0005b8 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcdoff 0x0005bd program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcdon 0x0005c2 program static D:\en0627\_assignment.X\en0627\_assignment.asm

lcd\_print 0x0005c7 program static D:\en0627\_assignment.X\en0627\_assignment.asm

enable 0x0005d5 program static D:\en0627\_assignment.X\en0627\_assignment.asm

delay 0x0005db program static D:\en0627\_assignment.X\en0627\_assignment.asm

delloop 0x0005dd program static D:\en0627\_assignment.X\en0627\_assignment.asm

deljmp 0x0005e3 program static D:\en0627\_assignment.X\en0627\_assignment.asm

hex16\_2\_ascii 0x0005e7 program static D:\en0627\_assignment.X\en0627\_assignment.asm

hex16\_2\_bcd 0x000604 program static D:\en0627\_assignment.X\en0627\_assignment.asm

hex16\_2\_bcd\_loop 0x000610 program static D:\en0627\_assignment.X\en0627\_assignment.asm

adjDEC 0x00061b program static D:\en0627\_assignment.X\en0627\_assignment.asm

adjBCD 0x000628 program static D:\en0627\_assignment.X\en0627\_assignment.asm

get\_ascii 0x000631 program static D:\en0627\_assignment.X\en0627\_assignment.asm

software\_error 0x00063c program static D:\en0627\_assignment.X\en0627\_assignment.asm

\_.org\_1\_0640 0x000640 program static D:\en0627\_assignment.X\en0627\_assignment.asm

Appendix D: Graph of Results

(mm)

(mm)

