1LSb Octal DAC Evaluation Board User's Guide

ISBN: 978-1-5224-9231-3

1LSb OCTAL DAC EVALUATION BOARD USER'S GUIDE

Table of Contents

Preface

NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our website (www.microchip.com) to obtain the latest documentation available.

Documents are identified with a "DS" number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is "DSXXXXXXXXA", where "XXXXXXXX" is the document number and "A" is the revision level of the document.

For the most up-to-date information on development tools, see the MPLAB® IDE online help. Select the Help menu, and then Topics to open a list of available online help files.

INTRODUCTION

This chapter contains general information that will be useful to know before using the 1LSb Octal DAC Evaluation Board. Items discussed in this chapter include:

- Document Layout
- Conventions Used in this Guide
- Recommended Reading
- The Microchip Website
- Customer Support
- Document Revision History

DOCUMENT LAYOUT

This document describes how to use the 1LSb Octal DAC Evaluation Board to demonstrate the performance of the MCP47CXBX8/MCP48CXBX8 DAC family. The manual layout is as follows:

- **Chapter 1. "Product Overview"** – Provides quick, step-by-step information on setting up the 1LSb Octal DAC Evaluation Board.
- **Chapter 2. "Installation and Operation"** – Important information about the 1LSb Octal DAC Evaluation Board.
- **Chapter 3. "Code"** – Includes instructions on how to get started with the 1LSb Octal DAC Evaluation Board.
- **Appendix A. "Schematics"** – Refer to the board's web page for the complete Schematics.
- **Appendix B. "Bill of Materials (BOM)"** – Refer to the board's web page for the complete Bill of Materials.

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

<table>
<tr><th>Description</th><th>Represents</th><th>Examples</th></tr>
<tr><td colspan="3">Arial font:</td></tr>
<tr><td rowspan="2">Italic characters</td><td>Referenced books</td><td>MPLAB® IDE User's Guide</td></tr>
<tr><td>Emphasized text</td><td>...is the only compiler...</td></tr>
<tr><td rowspan="3">Initial caps</td><td>A window</td><td>the Output window</td></tr>
<tr><td>A dialog</td><td>the Settings dialog</td></tr>
<tr><td>A menu selection</td><td>select Enable Programmer</td></tr>
<tr><td>Quotes</td><td>A field name in a window or dialog</td><td>"Save project before build"</td></tr>
<tr><td>Underlined, Italic text with right angle bracket</td><td>A menu path</td><td><u>File>Save</u></td></tr>
<tr><td rowspan="2">Bold characters</td><td>A dialog button</td><td>Click OK</td></tr>
<tr><td>A tab</td><td>Click the Power tab</td></tr>
<tr><td>N'Rnnnn</td><td>A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.</td><td>4'b0010, 2'hF1</td></tr>
<tr><td>Text in angle brackets < ></td><td>A key on the keyboard</td><td>Press <Enter>, <F1></td></tr>
<tr><td colspan="3">Courier New font:</td></tr>
<tr><td rowspan="7">Plain Courier New</td><td>Sample source code</td><td><code>#define START</code></td></tr>
<tr><td>Filenames</td><td><code>autoexec.bat</code></td></tr>
<tr><td>File paths</td><td><code>c:\mcc18\h</code></td></tr>
<tr><td>Keywords</td><td><code>_asm, _endasm, static</code></td></tr>
<tr><td>Command-line options</td><td><code>-Opa+, -Opa-</code></td></tr>
<tr><td>Bit values</td><td><code>0, 1</code></td></tr>
<tr><td>Constants</td><td><code>0xFF, 'A'</code></td></tr>
<tr><td>Italic Courier New</td><td>A variable argument</td><td><code>file.o</code>, where <code>file</code> can be any valid filename</td></tr>
<tr><td>Square brackets [ ]</td><td>Optional arguments</td><td><code>mcc18 [options] file [options]</code></td></tr>
<tr><td>Curly brackets and pipe character: { | }</td><td>Choice of mutually exclusive arguments; an OR selection</td><td><code>errorlevel {0|1}</code></td></tr>
<tr><td rowspan="2">Ellipses...</td><td>Replaces repeated text</td><td><code>var_name [, var_name...]</code></td></tr>
<tr><td>Represents code supplied by user</td><td><code>void main (void)
{ ...
}</code></td></tr>
</table>

RECOMMENDED READING

This user's guide describes how to use the 1LSb Octal DAC Evaluation Board. Another useful document is listed below. The following Microchip documents are available and recommended as a supplemental reference resource.

- **MCP47CXBX4/8 Data Sheet – *"8/10/12-Bit Digital-to-Analog Converters, 1 LSb INL, Quad/Octal Voltage Output with I2C Interface"* (DS20006537)**
- **MCP48CXBX4/8 Data Sheet – *"8/10/12-Bit Digital-to-Analog Converters, 1 LSb INL, Quad/Octal Voltage Output with SPI Interface"* (DS20006556)**

THE MICROCHIP WEBSITE

Microchip provides online support via our website at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded System Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the website at:
http://www.microchip.com/support.

DOCUMENT REVISION HISTORY

Revision A (November 2021)

- Initial release of this document.

Chapter 1. Product Overview

1.1 INTRODUCTION

This chapter provides an overview of the 1LSb Octal DAC Evaluation Board.

The MCP47CXBX8/MCP48CXBX8 is a 12-bit, 1 LSb DAC. The devices offer two memory options: MCP47CVBX8/MCP48CVBX8 devices have volatile memory, while the MCP47CMBX8/MCP47CMBX8 have 32-times programmable nonvolatile memory (MTP).The devices operate from a single supply voltage of 2.7V to 5.5V for full specified operation and 1.8V to 5.5V for digital operation.

The devices populated on the 1LSb Octal DAC Evaluation Board are the nonvolatile I^2C DAC (MCP47CMB28) and the nonvolatile SPI DAC (MCP48CMB28).

1.2 1LSb OCTAL DAC EVALUATION BOARD OVERVIEW

The Microchip 1LSb Octal DAC Evaluation Board is used to evaluate the MCP47CXBX8 and MCP48CXBX8 DAC families. Users can now easily evaluate features of the MCP47CXBX8/MCP48CXBX8 devices by connecting the evaluation board to any of the Microchip Curiosity microcontroller development boards. The 1LSb Octal DAC Evaluation Board supports the mikroBUS™ click™ board and can be mounted on any of the mikroBUS supported MCU boards. The 1LSb Octal DAC Evaluation Board supports both the I^2C family (MCP47CXBX8) of devices and the SPI DAC family (MCP48CXBX8).

The following figure shows the top view of the 1LSb Octal DAC Evaluation Board. Figure 1-2 shows the bottom view of the board.

FIGURE 1-1: 1LSb OCTAL DAC EVALUATION BOARD (TOP VIEW)

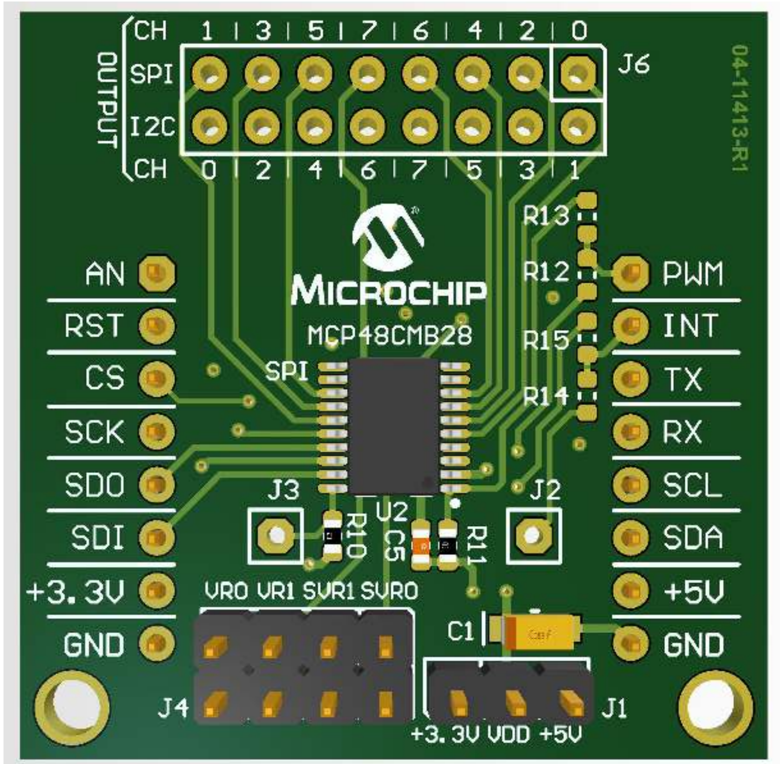
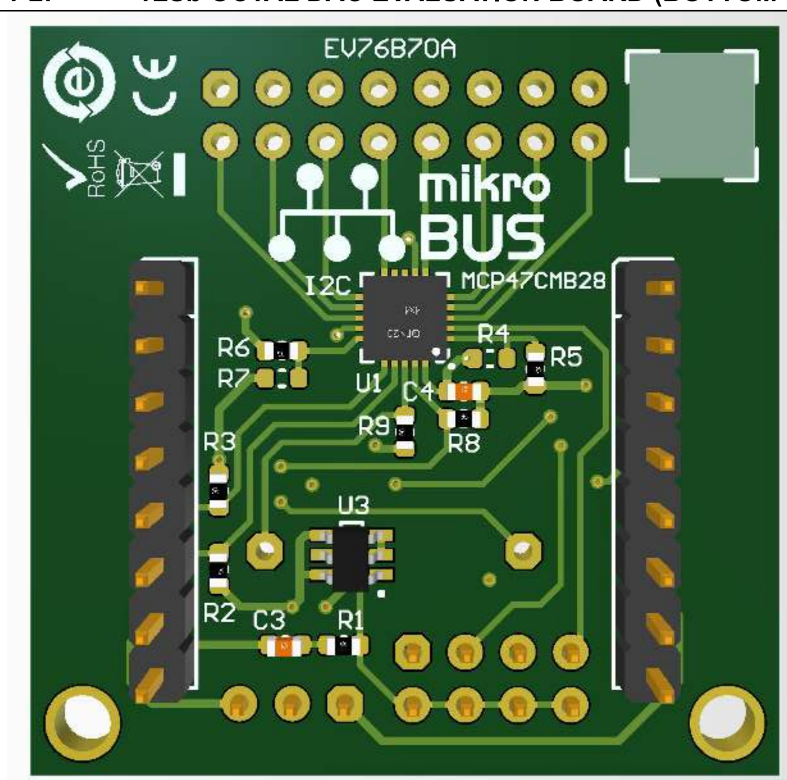
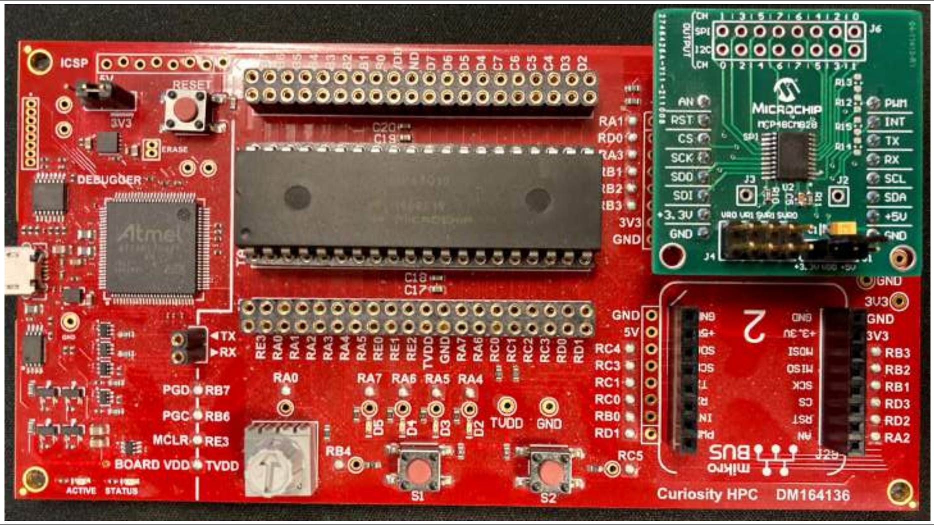


FIGURE 1-2: 1LSb OCTAL DAC EVALUATION BOARD (BOTTOM VIEW)



The following figure shows the 1LSb Octal DAC Evaluation Board mounted on the Curiosity microcontroller board, using the mikroBUS™ connector.

FIGURE 1-3: 1LSb OCTAL DAC EVALUATION BOARD MOUNTED ON THE MIKROBUS™ CLICK™ BOARD OF THE CURIOSITY HPC DEVELOPMENT BOARD (DM164136)



1.3 1LSb OCTAL DAC EVALUATION BOARD FEATURES

The 1LSb Octal DAC Evaluation Board is a fully assembled board. The board can be mounted on any of the Microchip MCU boards that feature the mikroBUS connector (MCU board not included with this board). The MCU board can be programmed to evaluate and demonstrate the operating performance of the MCP47CXBX8 and MCP48CXBX8 DAC family.

The user's guide includes the code example required to communicate with the MCP47CXBX8 and MCP48CXBX8 DAC family. The code example provided in the guide is intended for use with the Microchip Curiosity HPC Development Board (DM164136).

The 1LSb Octal DAC Evaluation Board features:

- Supports both SPI and I^2C devices
- Supports 5V and 3.3V options
- mikroBUS support makes it easy to evaluate with any of the Microchip MCU boards
- External voltage reference option for the DAC
- MCP150x voltage reference on board for external VREF option

1.4 1LSb OCTAL DAC EVALUATION BOARD KIT CONTENTS

The 1LSb Octal DAC Evaluation Board includes the following items:

- 1LSb Octal DAC Evaluation Board (1LSb OCTAL DAC EVALUATION BOARD)

NOTES:

Chapter 2. Installation and Operation

2.1 GETTING STARTED

The 1LSb Octal DAC Evaluation Board can be used by following the four steps listed below.

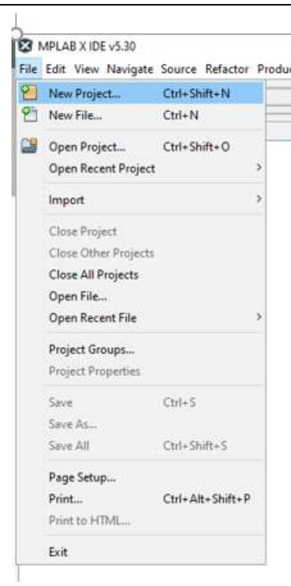
> **Note:** The demo code provided in **Chapter 3. "Code"** enables use of the 1LSb Octal DAC Evaluation Board with the Curiosity HPC Development Board (DM164136). Once the Curiosity HPC board has been programmed with this code, the DAC's output can be monitored on the V_{OUT} pin using an oscilloscope.

1. Connect the 1LSb Octal DAC Evaluation Board to the top-right mikroBUS™ header on the Curiosity HPC Board, as shown in Figure 1-3.
2. Compile the demo code provided in **Chapter 3. "Code"** and program the on-board PIC1847Q10 microcontroller. Steps to compile and program are explained below.

> **Note:** Download and install the MPLAB® IDE and XC18 compiler from www.microchip.com.

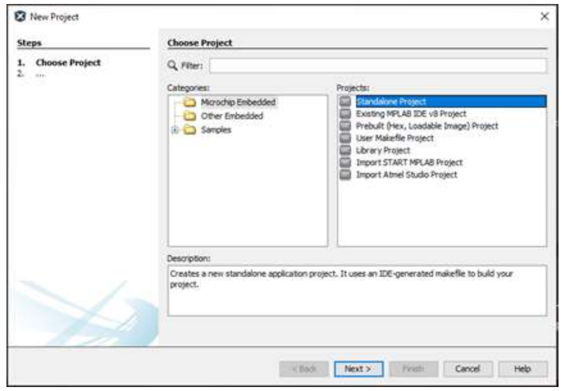
a) Open MPLAB IDE, go to the File menu and select "New Project...".

FIGURE 2-1: START A NEW PROJECT IN THE MPLAB IDE



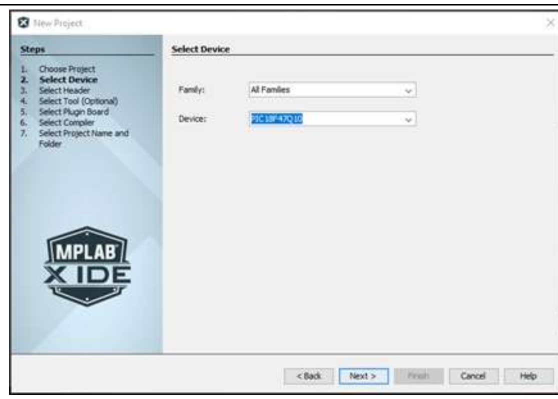
b) Select "Standalone Project" and click Next.

FIGURE 2-2: SELECT STANDALONE PROJECT



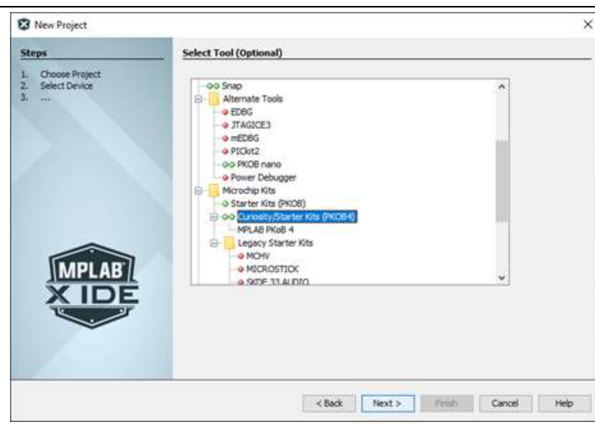
c) Select PIC18F47Q10 as the device and click Next.

FIGURE 2-3: SELECT THE DEVICE



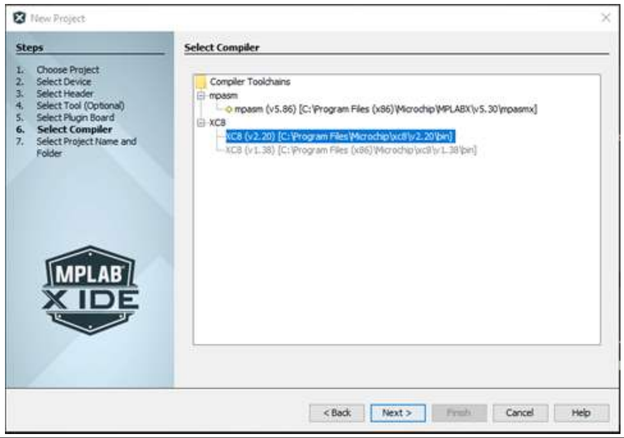
d) From the "Select Tool" menu, choose "Curiosity/Starter Kits (PKOB4)" and click Next.

FIGURE 2-4: SELECT THE TOOL



e) Select XC8 as the Compiler and click Next.

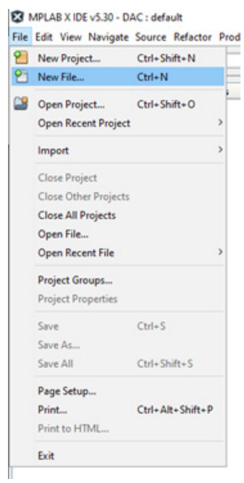
FIGURE 2-5: SELECT THE COMPILER



f) Name the project, provide the project location and click Finish.

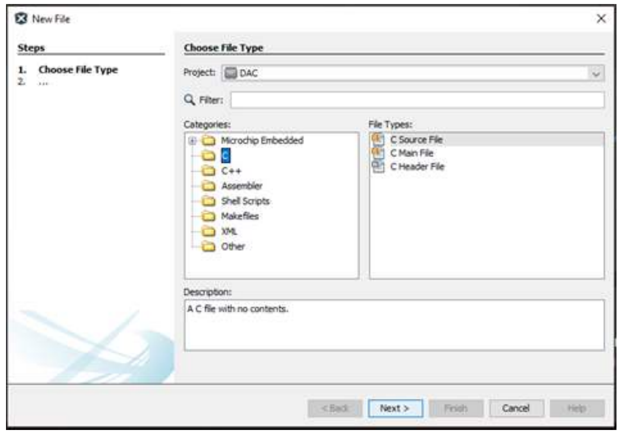
g) From the File menu, select “New File...”.

FIGURE 2-6: NEW FILE



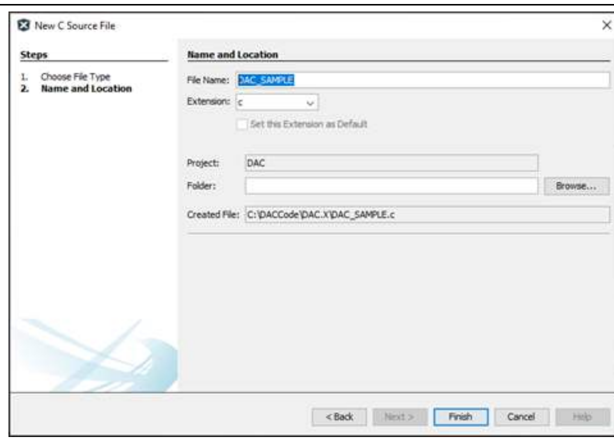
h) From Categories, select “C” and, from File Types, select “C Source File” and click Next.

FIGURE 2-7: NEW FILE



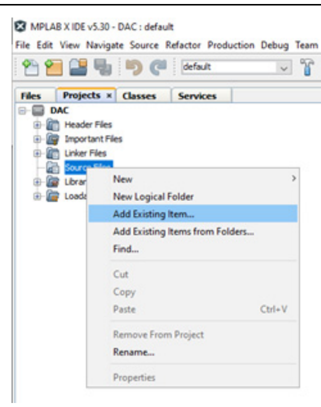
i) Name the file ("DAC_SAMPLE" in the example shown in the following figure) and click Finish.

FIGURE 2-8: NAME THE FILE



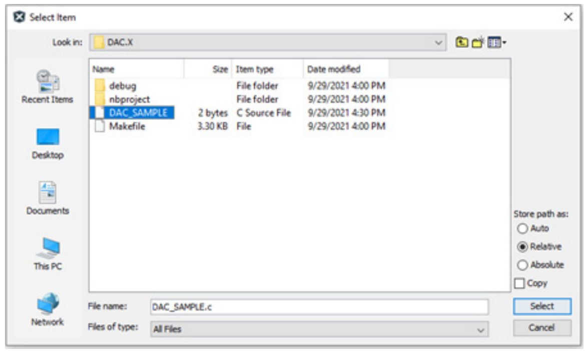
j) From the file, right click Source File and select "Add Existing Item...".

FIGURE 2-9: ADD A SOURCE FILE



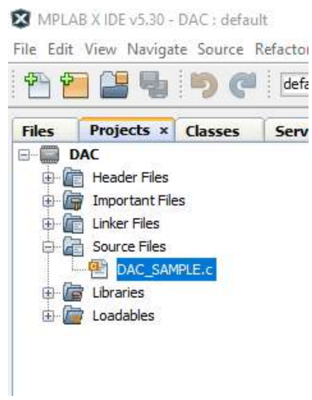
k) Navigate to the project folder, then select the `DAC_SAMPLE.c` file and click the Select button.

FIGURE 2-10: ADD A SOURCE FILE



l) This will add the code contained within the `DAC_SAMPLE.c` file to the source code, as shown in the following figure.

FIGURE 2-11: ADD A SOURCE FILE



m) Copy and paste the demo code to the `DAC_SAMPLE.c` file (make sure the code and comments are copied correctly). Connect the micro-USB cable to the micro-USB header on the left side of the Curiosity HPC Board using a micro-USB cable to provide power to the board. Press the icon shown in the following figure to compile and program the code.

FIGURE 2-12: COMPILE AND PROGRAM THE CODE

3. The LEDs on the Curiosity HPC Board will blink based on which code is running, and the user can monitor the DAC’s output using the V_{OUT} pin (see the following two figures).

FIGURE 2-13: THE LEDS, SWITCHES AND THE ANALOG POTENTIOMETER OF THE CURIOSITY HPC BOARD USED FOR THE 1LSb OCTAL DAC EVALUATION BOARD DEMO

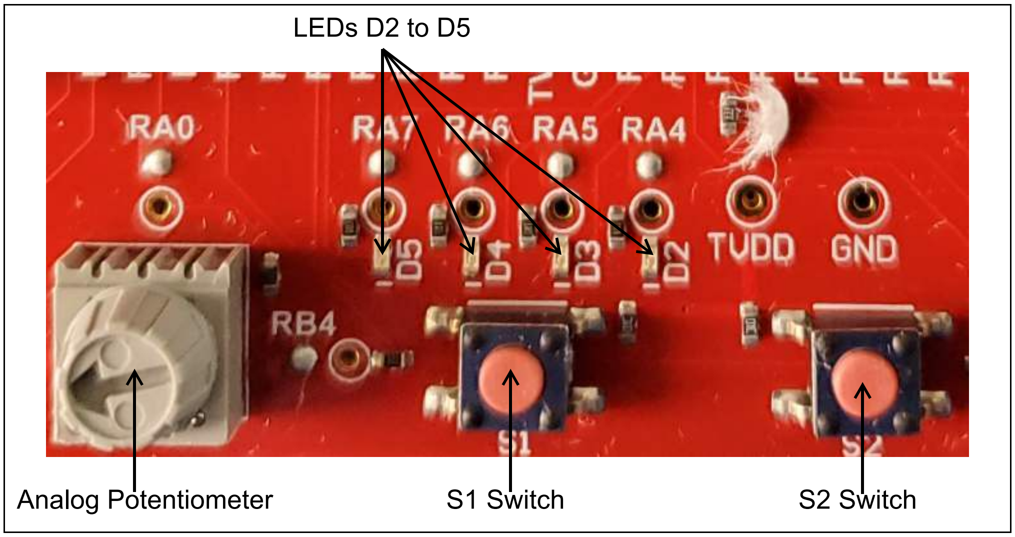
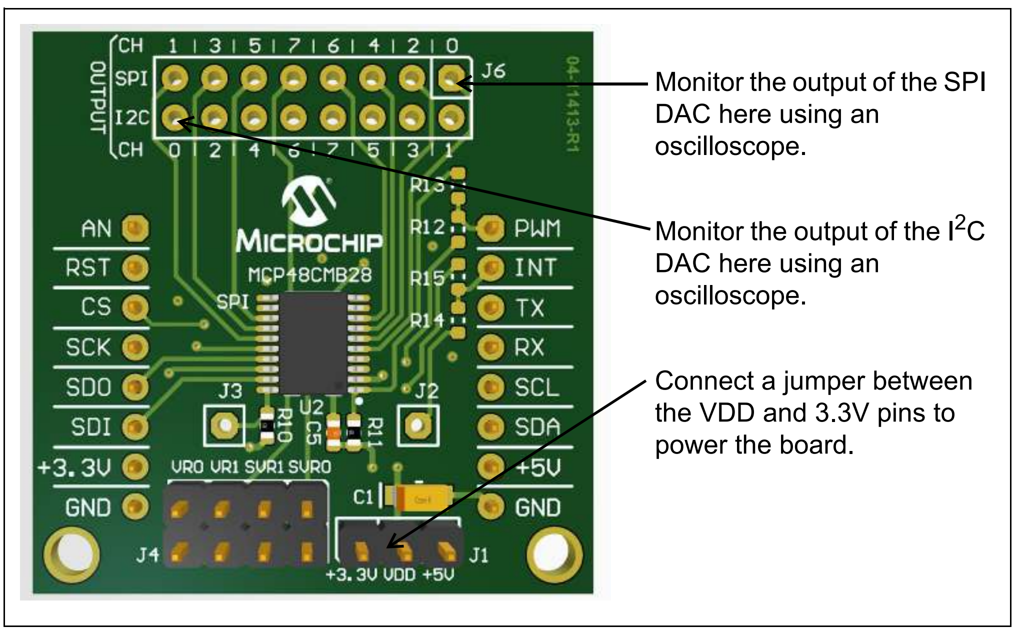


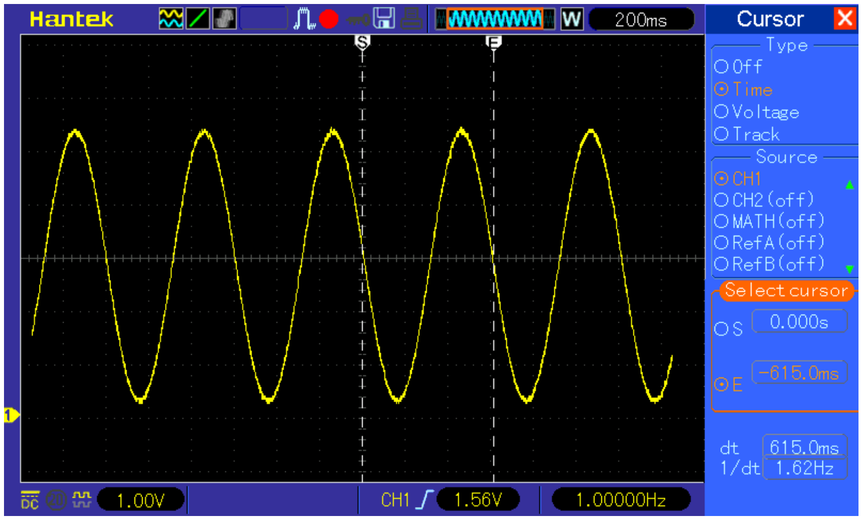
FIGURE 2-14: 1LSb OCTAL DAC EVALUATION BOARD SPI AND I²C OUTPUT WAVEFORM MONITORING



2.2 SPI DEMO

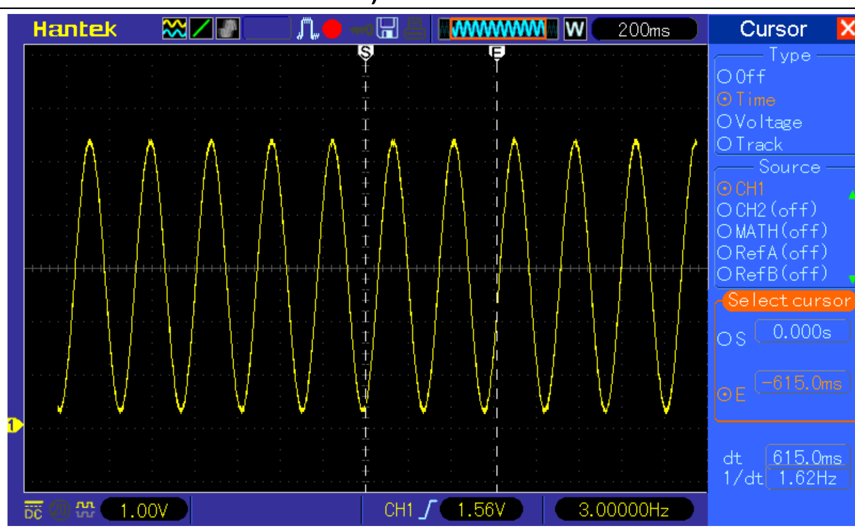
Once the Curiosity board is programmed and running, LED D4 will blink, while the other LEDs will remain off. This indicates that the SPI DAC is working and the output can be monitored on Channel 0 of the SPI output. When LED D4 is blinking, Channel 0 will output a sine wave as shown in the following figure.

FIGURE 2-15: SPI OUTPUT SINE WAVE (LED D4 BLINKING, LEDS D2, D3 AND D5 OFF)



The frequency of the sine wave can be modified by rotating the potentiometer on the Curiosity board, as shown in the following figure. Rotating the potentiometer will also change the blink rate of LED D4.

FIGURE 2-16: SPI OUTPUT SINE WAVE WITH VARYING FREQUENCY USING THE POTENTIOMETER (LED D4 BLINKING, LEDS D2, D3 AND D5 OFF)



When the S1 switch is pressed, the SPI output waveform will be a saw-tooth shape, as shown in the following figure. LED D5 will blink, while LEDs D2, D3 and D4 will be off. This indicates SPI DAC is working and the output can be monitored on Channel 0 of the SPI output. The frequency of the waveform can be modified using the potentiometer on the curiosity board (see Figure 2-18).

FIGURE 2-17: SPI OUTPUT SAW-TOOTH WAVEFORM (S1 SWITCH PRESSED, LED D5 BLINKING, LEDS D2, D3 AND D4 OFF)

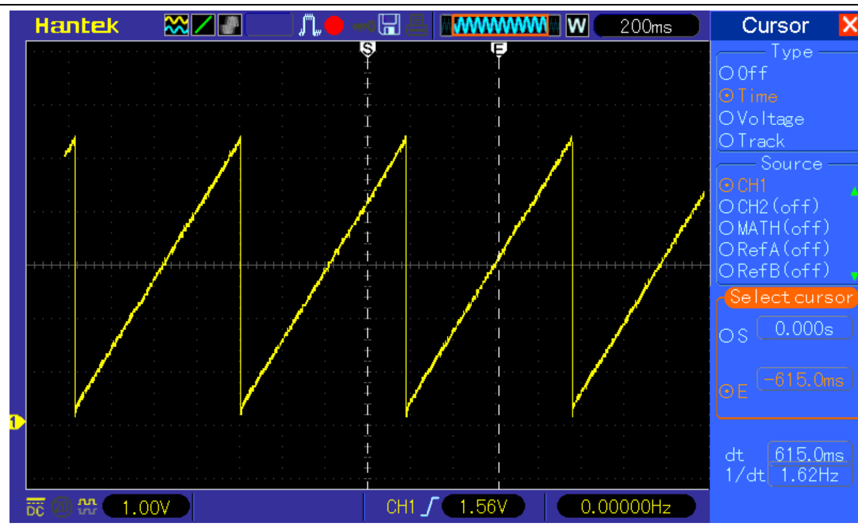
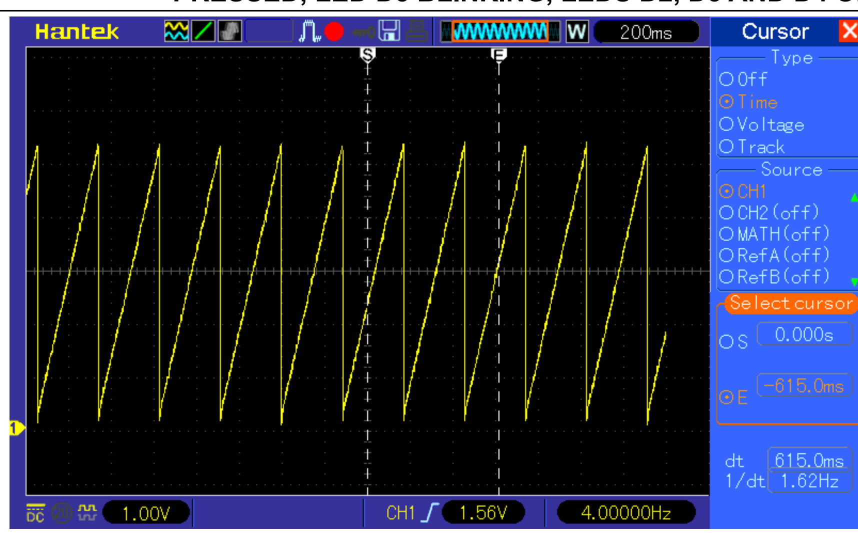


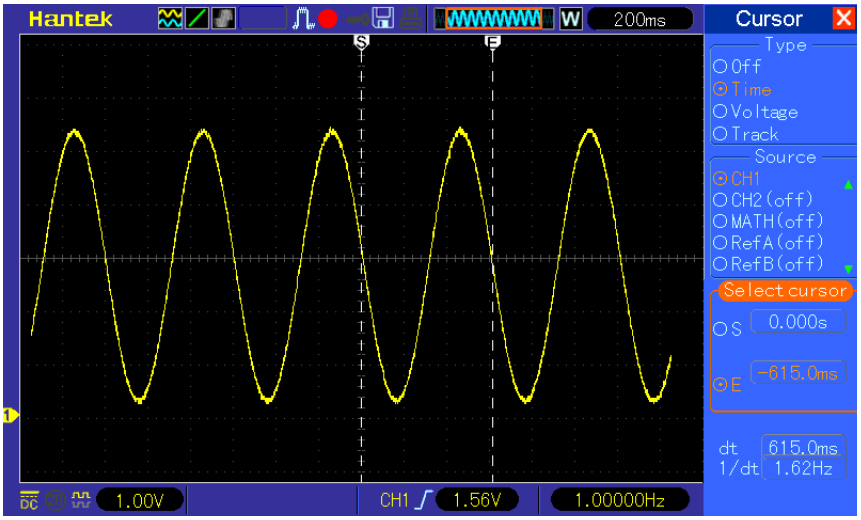
FIGURE 2-18: SPI OUTPUT SAW-TOOTH WAVE WITH VARYING FREQUENCY USING THE POTENTIOMETER (S1 SWITCH PRESSED, LED D5 BLINKING, LEDS D2, D3 AND D4 OFF)



2.3 I^{2}C DEMO

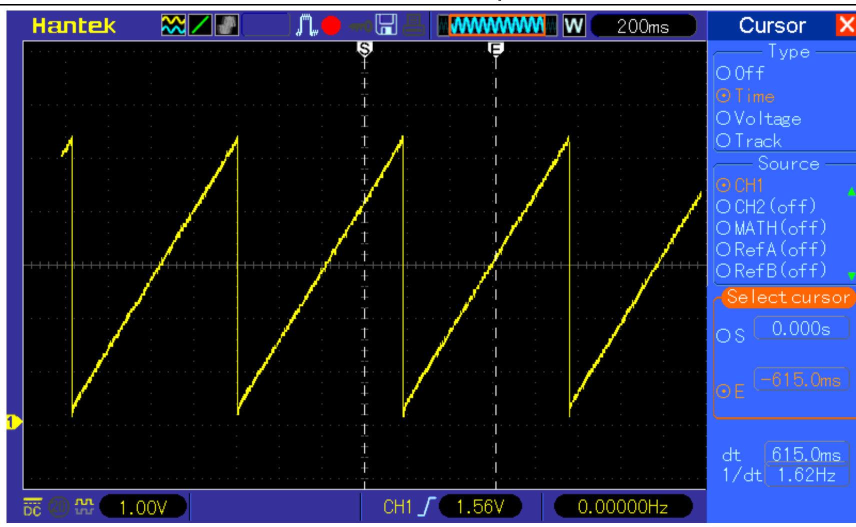
Once the Curiosity board is programmed and running, press the S2 switch. The D2 LED will blink while all the other LEDs will remain off. This indicates that the I^{2}C DAC is working and the output can be monitored on Channel 0 of the I^{2}C DAC output. When LED D2 is blinking, the DAC's Channel 0 will output a sine wave as shown in the following figure. The frequency of the sine wave can be varied using the analog potentiometer.

FIGURE 2-19: I^{2}C OUTPUT SINE WAVE (LED D2 BLINKING, LEDS D3, D4 AND D5 OFF)



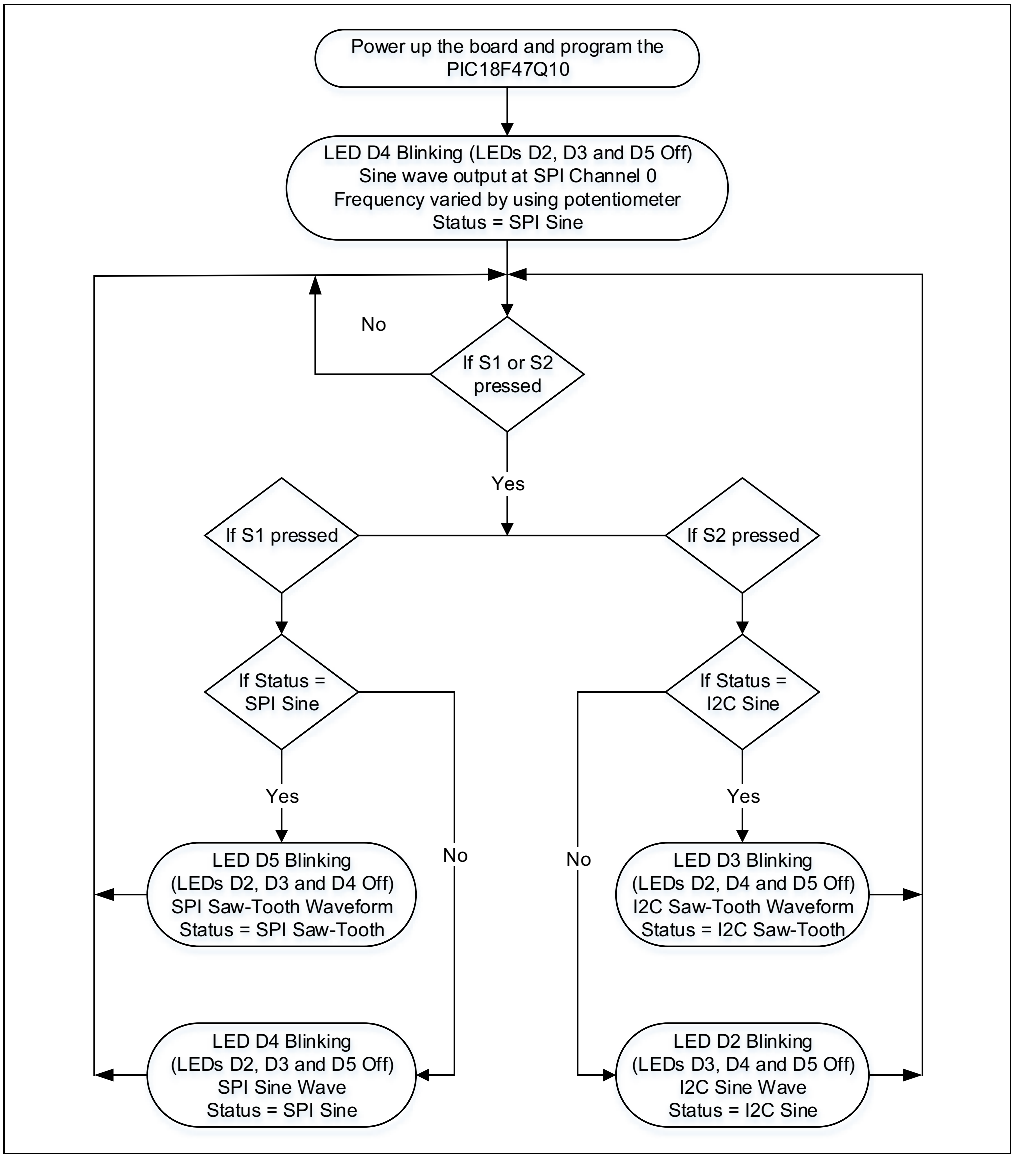
When the S2 switch is pressed, the I^{2}C output waveform will be saw-tooth shaped. The D3 LED will blink while all the other LEDs will remain off. This indicates that the I^{2}C DAC is working, and the output can be monitored on Channel 0 of the I^{2}C DAC output. Channel 0 will output a saw-tooth waveform as shown in the following figure. The frequency of the waveform can be modified using the potentiometer on the Curiosity board.

FIGURE 2-20: I^{2}C OUTPUT SAW-TOOTH WAVEFORM (LED D3 BLINKING, LEDS D2, D4 AND D5 OFF)



The following figure shows a flow-chart for the DAC demo.

FIGURE 2-21: FLOW-CHART FOR THE USER DEMO



Chapter 3. Code

3.1 DEMO CODE

Program the PIC18F47Q10 on the Curiosity board with the following code to enable use of the connected 1LSb Octal DAC Evaluation Board, which will allow for monitoring and testing the DAC. The sample code is also provided separately on the product page for convenience.

```
#include <xc.h>
#include <pic18f47q10.h>
// PIC18F47Q10 Configuration Bit Settings
//Sep29
// CONFIG1L
#pragma config FEXTOSC = OFF
#pragma config RSTOSC = HFINTOSC_1MHZ// Power-up default value for COSC bits (HFINTOSC with HFFRQ = 4 MHz and CDIV = 4:1)

// CONFIG1H
#pragma config CLKOUTEN = OFF   // Clock Out Enable bit (CLKOUT function is disabled)
#pragma config CSWEN = ON       // Clock Switch Enable bit (Writing to NOSC and NDIV is allowed)
#pragma config FCMEN = ON       // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor enabled)

// CONFIG2L
#pragma config MCLRE = EXTMCLR  // Master Clear Enable bit (MCLR pin (RE3) is MCLR)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (Power up timer disabled)
#pragma config LPBOREN = OFF    // Low-power BOR enable bit (Low power BOR is disabled)
#pragma config BOREN = SBORDIS  // Brown-out Reset Enable bits (Brown-out Reset enabled , SBOREN bit is ignored)

// CONFIG2H
#pragma config BORV = VBOR_190  // Brown Out Reset Voltage selection bits (Brown-out Reset Voltage (VBOR) set to 1.90V)
#pragma config ZCD = OFF        // ZCD Disable bit (ZCD disabled. ZCD can be enabled by setting the ZCDSEN bit of ZCDCON)
#pragma config PPS1WAY = ON     // PPSLOCK bit One-Way Set Enable bit (PPSLOCK bit can be cleared and set only once; PPS
registers remain locked after one clear/set cycle)
#pragma config STVREN = ON      // Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset)
#pragma config XINST = OFF      // Extended Instruction Set Enable bit (Extended Instruction Set and Indexed Addressing
Mode disabled)

// CONFIG3L
#pragma config WDTCPS = WDTCPS_31// WDT Period Select bits (Divider ratio 1:65536; software control of WDTPS)
#pragma config WDTE = OFF       // WDT operating mode (WDT Disabled)

// CONFIG3H
#pragma config WDTCWS = WDTCWS_7// WDT Window Select bits (window always open (100%); software control; keyed access not
required)
#pragma config WDTCCS = SC      // WDT input clock selector (Software Control)

// CONFIG4L
#pragma config WRT0 = OFF       // Write Protection Block 0 (Block 0 (000800-003FFFh) not write-protected)
#pragma config WRT1 = OFF       // Write Protection Block 1 (Block 1 (004000-007FFFh) not write-protected)
#pragma config WRT2 = OFF       // Write Protection Block 2 (Block 2 (008000-00BFFFh) not write-protected)
#pragma config WRT3 = OFF       // Write Protection Block 3 (Block 3 (00C000-00FFFFh) not write-protected)
```

```
#pragma config WRT4 = OFF       // Write Protection Block 4 (Block 4 (010000-013FFFh) not write-protected)
#pragma config WRT5 = OFF       // Write Protection Block 5 (Block 5 (014000-017FFFh) not write-protected)
#pragma config WRT6 = OFF       // Write Protection Block 6 (Block 6 (018000-01BFFFh) not write-protected)
#pragma config WRT7 = OFF       // Write Protection Block 7 (Block 7 (01C000-01FFFFh) not write-protected)

// CONFIG4H
#pragma config WRTC = OFF       // Configuration Register Write Protection bit (Configuration registers (300000-30000Bh)
not write-protected)
#pragma config WRTB = OFF       // Boot Block Write Protection bit (Boot Block (000000-0007FFh) not write-protected)
#pragma config WRTD = OFF       // Data EEPROM Write Protection bit (Data EEPROM not write-protected)
#pragma config SCANE = ON       // Scanner Enable bit (Scanner module is available for use, SCANMD bit can control the
module)
#pragma config LVP = ON         // Low Voltage Programming Enable bit (Low voltage programming enabled. MCLR/VPP pin
function is MCLR. MCLRE configuration bit is ignored)

// CONFIG5L
#pragma config CP = OFF         // UserNVM Program Memory Code Protection bit (UserNVM code protection disabled)
#pragma config CPD = OFF        // DataNVM Memory Code Protection bit (DataNVM code protection disabled)

// CONFIG5H

// CONFIG6L
#pragma config EBTR0 = OFF      // Table Read Protection Block 0 (Block 0 (000800-003FFFh) not protected from table reads
executed in other blocks)
#pragma config EBTR1 = OFF      // Table Read Protection Block 1 (Block 1 (004000-007FFFh) not protected from table reads
executed in other blocks)
#pragma config EBTR2 = OFF      // Table Read Protection Block 2 (Block 2 (008000-00BFFFh) not protected from table reads
executed in other blocks)
#pragma config EBTR3 = OFF      // Table Read Protection Block 3 (Block 3 (00C000-00FFFFh) not protected from table reads
executed in other blocks)
#pragma config EBTR4 = OFF      // Table Read Protection Block 4 (Block 4 (010000-013FFFh) not protected from table reads
executed in other blocks)
#pragma config EBTR5 = OFF      // Table Read Protection Block 5 (Block 5 (014000-017FFFh) not protected from table reads
executed in other blocks)
#pragma config EBTR6 = OFF      // Table Read Protection Block 6 (Block 6 (018000-01BFFFh) not protected from table reads
executed in other blocks)
#pragma config EBTR7 = OFF      // Table Read Protection Block 7 (Block 7 (01C000-01FFFFh) not protected from table reads
executed in other blocks)

// CONFIG6H
#pragma config EBTRB = OFF      // Boot Block Table Read Protection bit (Boot Block (000000-0007FFh) not protected from
table reads executed in other blocks)
```

```
// #pragma config statements should precede project file include

void ADC_Initialize(void);
void Delay( unsigned int );
void SPI1_Initialize(void);
void SwitchInit(void);
void SW1(void);
void SW2(void);
void SPI1_WriteWord(unsigned char addr,unsigned int data);
unsigned int SPI1_ReadWord (unsigned char addr );
void SystemInit(void);
void StartADCoversion(void);

void I2C1_Initialize(void);
void I2C1Write(unsigned char ControlByte ,unsigned char addr,unsigned int data);
unsigned int  I2C1ReadLastAddr(unsigned char ControlByte );
unsigned int  I2C1Read(unsigned char ControlByte ,unsigned char addr);
void I2C1_IntPoll(void);
void I2C1_ACKCheck(void);
void I2C1_ACK(void);
void I2C1_NACK(void);
#define CNTRLBYTE 0xC0

#define CS PORTAbits.RA3
#define TRIS_CS TRISAbits.TRISA3
#define DIGTAL_CS   ANSELAbits.ANSELA3
#define SWITCH1 PORTBbits.RB4
#define TRIS_SWITCH1 TRISBbits.TRISB4
#define DIGTAL_SWITCH1   ANSELBbits.ANSELB4
#define SWITCH2 PORTCbits.RC5
#define TRIS_SWITCH2 TRISCbits.TRISC5
#define DIGTAL_SWITCH2   ANSELCbits.ANSELC5

#define LED2 PORTAbits.RA4
#define LED3 PORTAbits.RA5
#define LED4 PORTAbits.RA6
#define LED5 PORTAbits.RA7

#define SPI_SINE        0x55
#define SPI_SAWTOOTH    0xAA
#define I2C_SINE        0x50
```

```
#define I2C_SAWTOOTH    0xA0
unsigned int Check[360];
unsigned int SINE[]={
2047,2083,2118,2154,2190,2225,2261,2296,2332,2367,2402,2438,2473,2507,2542,2577,
2611,2645,2680,2713,2747,2781,2814,2847,2880,2912,2944,2976,3008,3039,3071,3101,
3132,3162,3192,3221,3250,3279,3307,3335,3363,3390,3417,3443,3469,3494,3519,3544,
3568,3592,3615,3638,3660,3682,3703,3724,3744,3764,3783,3802,3820,3837,3854,3871,
3887,3902,3917,3931,3945,3958,3971,3982,3994,4005,4015,4024,4033,4042,4049,4056,
4063,4069,4074,4079,4083,4086,4089,4091,4093,4094,4094,4094,4093,4091,4089,4086,
4083,4079,4074,4069,4063,4056,4049,4042,4033,4024,4015,4005,3994,3982,3971,3958,
3945,3931,3917,3902,3887,3871,3854,3837,3820,3802,3783,3764,3744,3724,3703,3682,
3660,3638,3615,3592,3568,3544,3519,3494,3469,3443,3417,3390,3363,3335,3307,3279,
3250,3221,3192,3162,3132,3101,3071,3039,3008,2976,2944,2912,2880,2847,2814,2781,
2747,2713,2680,2645,2611,2577,2542,2507,2473,2438,2402,2367,2332,2296,2261,2225,
2190,2154,2118,2083,2047,2011,1976,1940,1904,1869,1833,1798,1762,1727,1692,1656,
1621,1587,1552,1517,1483,1449,1414,1381,1347,1313,1280,1247,1214,1182,1150,1118,
1086,1055,1024,
993,962,932,902,873,844,815,787,759,731,704,677,651,625,600,575,550,526,502,479,
456,434,412,391,370,350,330,311,292,274,257,240,223,207,192,177,163,149,136,123,
112,100,
89,79,70,61,52,45,38,31,25,20,15,11,8,5,3,1,0,0,0,1,3,5,8,11,15,20,25,31,38,45,
52,61,70,79,89,
100,112,123,136,149,163,177,192,207,223,240,257,274,292,311,330,350,370,391,412,
434,456,479,502,526,550,575,600,625,651,677,704,731,759,787,815,844,873,902,932,
962,993,
1024,1055,1086,1118,1150,1182,1214,1247,1280,1313,1347,1381,1414,1449,1483,1517,
1552,1587,1621,1656,1692,1727,1762,1798,1833,1869,1904,1940,1976,2011,2047,
};

unsigned int temp,del,del2,Status,SPIReadData,I2CReadData;
unsigned int *SinePtr;
#define DAC0 0
#define DAC1 1
#define DAC2 2
#define DAC3 3
#define DAC4 4
#define DAC5 5
#define DAC6 6
#define DAC7 7
int main()
{
    SystemInit();
```

```
SwitchInit();
ADC_Initialize();
SPI1_Initialize();
Status=SPI_SINE;
SinePtr=&SINE[0];
while (1)
{
    SPI1_Initialize();
    while(Status==SPI_SINE)
    {
        for(temp=0;temp<360;temp++)
        {
        StartADCoversion();
        for(del=0;del<(ADRES>>4);del++);    // delay based from ADC pot reading
        SPI1_WriteWord(DAC0,*(SinePtr+temp));
        SW1();
        SW2();
        }
        LED2=0;
        LED3=0;
        LED4^=1;
        LED5=0;
    }
    SPI1_Initialize();
    while(Status==SPI_SAWTOOTH)
    {
        for(temp=0;temp<0xFFF;temp++)
        {
        StartADCoversion();
        for(del=0;del<(ADRESL>>4);del++);
        SPI1_WriteWord(0x00,temp);
        SW1();
        SW2();
        }
        LED2=0;
        LED3=0;;
        LED4=0;
        LED5^=1;
    }
        I2C1_Initialize();
  while(Status==I2C_SINE)
    {
```

```
            for(temp=0;temp<360;temp++)
            {
            StartADCoversion();
            for(del=0;del<(ADRES>>2);del++);    // delay based from ADC pot reading
            I2C1Write(CNTRLBYTE,DAC0,*(SinePtr+temp));
            SW1();
            SW2();
            }
            LED2^=1;
            LED3=0;
            LED4=0;
            LED5=0;
        }
            I2C1_Initialize();
        while(Status==I2C_SAWTOOTH)
        {
          for(temp=0;temp<0xFFF;(temp=temp+20))
            {

            StartADCoversion();
            for(del=0;del<(ADRES>>2);del++);    // delay based from ADC pot reading
            I2C1Write(CNTRLBYTE,DAC0,temp);
            SW1();
            SW2();
            }

            LED2=0;
            LED3^=1;
            LED4=0;
            LED5=0;
        }
    }
    while(1);
    /*******Example for read routines************/
    I2CReadData=I2C1ReadLastAddr(CNTRLBYTE);    // Read the last address accessed
    I2CReadData=I2C1Read(CNTRLBYTE,DAC0);       // Read DAC0
    SPIReadData=SPI1_ReadWord(DAC0);
    while(1);
    /*******Example for read routines************/
}
/*****************************************************************
 * Function to Write to SPI1 Module
 *****************************************************************/
```

```
void SPI1_WriteWord (unsigned char addr,unsigned int data )
{

    CS=0;
    SSP1BUF = (addr<<3)&0xF8;   //Shift and & write command AD4:AD3:AD2:AD1:AD0:0:0:X
    while(!PIR3bits.SSP1IF);    //Wait for the interrupt
    PIR3bits.SSP1IF = 0;        //Clear the interrupt flag

    SSP1BUF = ((data>>8)&(0x00FF));// Shift the data to transmit the MS byte
    while(!PIR3bits.SSP1IF);
    PIR3bits.SSP1IF = 0;

    SSP1BUF = ((data)&(0x00FF));    //transmit the LS byte
    while(!PIR3bits.SSP1IF);
    PIR3bits.SSP1IF = 0;
   CS=1;
}
///*************************************************************
// * Function to Read SPI1
// *************************************************************/
unsigned int SPI1_ReadWord (unsigned char addr )
{
    unsigned int dataread;
    CS=0;
    SSP1BUF = (addr<<3)|0x06;    //Shift and & with 0000 011X for write command
    while(!PIR3bits.SSP1IF);
    PIR3bits.SSP1IF = 0;


    SSP1BUF = 0x00;
    while(!PIR3bits.SSP1IF);
    PIR3bits.SSP1IF = 0;
    dataread=SSP1BUF;
    dataread=dataread<<8;       //Read the MS byte
    SSP1BUF = 0x00;
    while(!PIR3bits.SSP1IF);
    PIR3bits.SSP1IF = 0;
    dataread=dataread|SSP1BUF;  //Read the LS byte
    CS=1;
    return dataread;
}
/*************************************************************
```

```
 * Function to Initialize  SPI1 Module
 ****************************************************************/
void SPI1_Initialize(void)
{
    SSP1CON1=0;
    SSP1STAT=0;
    SSP1ADD=0;
//Setup PPS Pins
    ANSELBbits.ANSELB1=0;   //Make the PORTS digital
    ANSELBbits.ANSELB2=0;   //Make the PORTS digital
    ANSELBbits.ANSELB3=0;   //Make the PORTS digital
    TRISBbits.TRISB1=0;     //Make port output
    TRISBbits.TRISB2=1;     //Make port output
    TRISBbits.TRISB3=0;
//PPS pin assignment
    SSP1CLKPPS = 9;
    SSP1DATPPS = 10;
    RB1PPS    = 15;
    RB3PPS    = 16;

    SSP1STAT = 0x40;
    SSP1CON1 = 0x00;
    SSP1ADD = 0x03;
    SSP1CON1bits.SSPEN = 1;
}

/****************************************************************
 * Initialize the Switch
 ****************************************************************/
void SwitchInit(void)
{
    DIGTAL_SWITCH1=0;       //Make the port digital
    TRIS_SWITCH1=1;         //Make the port input
    DIGTAL_SWITCH2=0;
    TRIS_SWITCH2=1;
}
/****************************************************************
 * Check for Switch1 pressed
 ****************************************************************/
void SW1(void)
{
    if(SWITCH1==0)
```

```
    {

        for(del2=0;del2<100;del2++)
        {
         if(SWITCH1)
         {
         break;
         }
        }
        while(SWITCH1==0);// wait here for the switch to release
        LED3=0;
        LED2=0;
        if(Status==SPI_SINE)
        {
           Status=SPI_SAWTOOTH;
        }
        else
        {
            Status=SPI_SINE;
        }

    }

}
/****************************************************************
 * Check for Switch2 pressed
 ****************************************************************/
void SW2(void)
{
   if(SWITCH2==0)
    {

        for(del2=0;del2<100;del2++)
        {
         if(SWITCH2)
         {
         break;
         }
        }
        while(SWITCH2==0);// wait here for the switch to release
        LED3=0;
        LED2=0;
```

```
        if(Status==I2C_SINE)
        {
           Status=I2C_SAWTOOTH;
        }
        else
        {
            Status=I2C_SINE;
        }
   }

}
/*****************************************************************
 * Initialize ADC module
 *****************************************************************/
void ADC_Initialize(void)
{
    ANSELAbits.ANSELA0=1;
    TRISAbits.TRISA0=1;
    //Setup ADC
    ADCON0bits.ADFM = 1;    //right justify     0=left
    ADCON0bits.ADCS = 1;    //FRC Clock
    ADPCH = 0x00;           //RA0 is Analog channel
    TRISAbits.TRISA0 = 1;   //Set RA0 to input
    ANSELAbits.ANSELA0 = 1; //Set RA0 to analog
    ADCON0bits.ADON = 1;    //Turn ADC On
}
/*****************************************************************
 * Start AD conversion
 *****************************************************************/
void StartADCoversion(void)
{
     if(ADCON0bits.GO ==0)// if conversion complete
     {
         ADCON0bits.GO = 1;
     }
}
/*****************************************************************
 * System initialization
 *****************************************************************/
void SystemInit(void)
{
    OSCCON1 = 0x62;
```

```
    OSCCON3 = 0x00;
    OSCEN = 0x00;
    OSCFRQ = 0x06;
    OSCTUNE = 0x00;
    TRISA=0; //make port output
    ANSELA=0; //make port digital

    LED2=1;
    LED3=0;
    LED4=0;
    LED5=0;
    CS=1;
    TRIS_CS=0;
    DIGTAL_CS=0;
}
/***************************************************************
 * Initialize the I2C1 module
 ***************************************************************/
void I2C1_Initialize(void)
{
    SSP1CON1=0;
    SSP1STAT=0;
    SSP1ADD=0;

    /* Set pins RB1 and RB2 as Digital */
    ANSELCbits.ANSELC3 = 0;
    ANSELCbits.ANSELC4 = 0;
    TRISCbits.TRISC3=0;
    TRISCbits.TRISC4=0;
    PORTCbits.RC3=0;
    PORTCbits.RC4=0;
    PORTCbits.RC3=1;
    PORTCbits.RC4=1;
    TRISCbits.TRISC3=1;
    TRISCbits.TRISC4=1;
    SSP1CON1=0;

    WPUCbits.WPUC3 = 1;  // enable pull up
    WPUCbits.WPUC4 = 1;
    SSP1CLKPPS = 0x13;
    RC3PPS = 0x0F;      //RC3 SCL1;
    RC4PPS = 0x10;      //RC4 SDA1;
```

```
    SSP1DATPPS = 0x14;

    SSP1CON1bits.SSPM3 = 1;
    SSP1ADD  = 0x9F;
    PIR3bits.SSP1IF = 0;    // clear the interrupt
    SSP1CON1bits.SSPEN = 1;
}
/****************************************************************
 *Function to write the I2C with Control Byte ,address and data
 ****************************************************************/
void I2C1Write(unsigned char ControlByte,unsigned char addr,unsigned int data)
{
    PIR3bits.SSP1IF = 0;
    SSP1CON2bits.SEN = 1;           // Start I2c
    I2C1_IntPoll();

    SSP1BUF  = (ControlByte&0xFE);  // write bit set
    I2C1_IntPoll();
    I2C1_ACKCheck();

    SSP1BUF  = ((addr<<3)&0xF8); // address and write command AD4:AD3:AD2:AD1:AD0:0:0:X
    I2C1_IntPoll();
    I2C1_ACKCheck();

    SSP1BUF = ((data>>8)&(0x00FF));
    I2C1_IntPoll();
    I2C1_ACKCheck();

    SSP1BUF = ((data)&(0x00FF));
    I2C1_IntPoll();
    I2C1_ACKCheck();

    SSP1CON2bits.PEN = 1;       //Stop
    I2C1_IntPoll();

}
/****************************************************************
 *Function to Read from an addressed location
 ****************************************************************/
unsigned int  I2C1Read(unsigned char ControlByte ,unsigned char addr)
{
    unsigned int dataread;
```

```
    PIR3bits.SSP1IF = 0;
    SSP1CON2bits.SEN = 1;           // Start
    I2C1_IntPoll();
    SSP1BUF  = (ControlByte&0xFE);//;write
    I2C1_IntPoll();
    I2C1_ACKCheck();

    //address where to read from
    SSP1BUF  = (((addr<<3)&0xF8)|0x06); //address,write command AD4:AD3:AD2:AD1:AD0:0:0:X
    I2C1_IntPoll();
    I2C1_ACKCheck();

    SSP1CON2bits.RSEN = 1;          //Repeated Start
    I2C1_IntPoll();

    SSP1BUF  =(ControlByte|0x01);   //; or with read bit 0
    I2C1_IntPoll();
    I2C1_ACKCheck();

    SSP1CON2bits.RCEN=1;            //enable receive

    I2C1_IntPoll();
    dataread=SSP1BUF;
    I2C1_ACK();                     //Send ACK for received data

    SSP1CON2bits.RCEN=1;            //enable receive
    I2C1_IntPoll();
    dataread=((dataread<<8)|SSP1BUF);;
    I2C1_NACK();                    //Send NACK for received data

    SSP1CON2bits.PEN = 1;           //Stop
    I2C1_IntPoll();
    return dataread;

}
/*****************************************************************
 *Function to read from the last addressed location
 *****************************************************************/
unsigned int  I2C1ReadLastAddr(unsigned char ControlByte )
{
    unsigned int dataread;
    PIR3bits.SSP1IF = 0;
```

```
    SSP1CON2bits.SEN = 1;           // Start
    I2C1_IntPoll();

    SSP1BUF  = (ControlByte|0x01);  //; or with read bit 0
    I2C1_IntPoll();
    I2C1_ACKCheck();

    SSP1CON2bits.RCEN=1;            //enable receive
    I2C1_IntPoll();
    dataread=SSP1BUF;
    I2C1_ACK();                     //Send ACK for received data

    SSP1CON2bits.RCEN=1;            //enable receive
    I2C1_IntPoll();
    dataread=((dataread<<8)|SSP1BUF);;
    I2C1_NACK();                    //Send NACK for received data


    SSP1CON2bits.PEN = 1;           //Stop
    I2C1_IntPoll();
    return dataread;
}
/****************************************************************
 *I2C Acknowledge
 ****************************************************************/
void I2C1_ACK(void)
{
    SSP1CON2bits.ACKDT=0;           //ACK
    SSP1CON2bits.ACKEN=1;
    I2C1_IntPoll();
}
/****************************************************************
 *I2C No Acknowledge
 ****************************************************************/
void I2C1_NACK(void)
{
    SSP1CON2bits.ACKDT=1;           //NACK
    SSP1CON2bits.ACKEN=1;
    I2C1_IntPoll();
}
/****************************************************************
 *I2C check ACK
```

```
 **************************************************************/
void I2C1_ACKCheck(void)            //Check ACK from Slave
{
    if ( SSP1CON2bits.ACKSTAT==1)
    {
        return;
    }
}
/*************************************************************
 *I2C Poll for interrupt flag
 **************************************************************/
void I2C1_IntPoll(void)             // Poll of the I2C interrupt
{
    while (!PIR3bits.SSP1IF);
    PIR3bits.SSP1IF = 0;

}
```

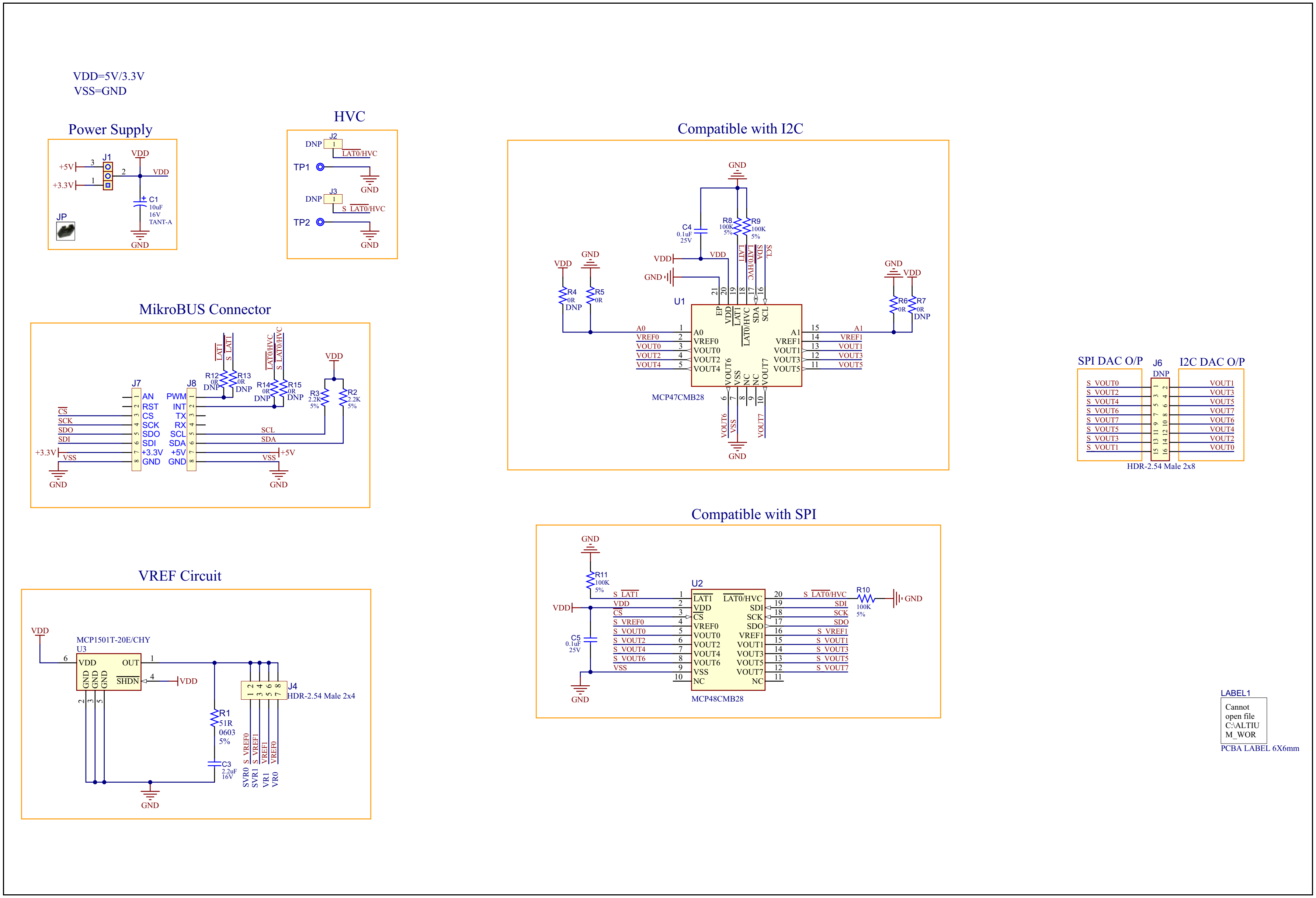
Appendix A. Schematics

A.1 INTRODUCTION

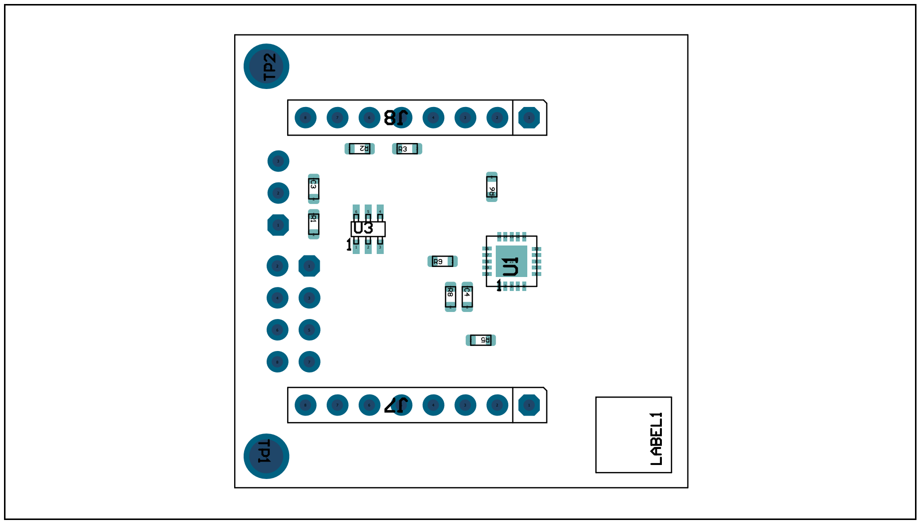
This appendix contains the following schematics and layouts for the 1LSb Octal DAC Evaluation Board - EV76B70A:

- Board – Schematics
- Board – Top Assembly Drawing
- Board – Bottom Assembly Drawing

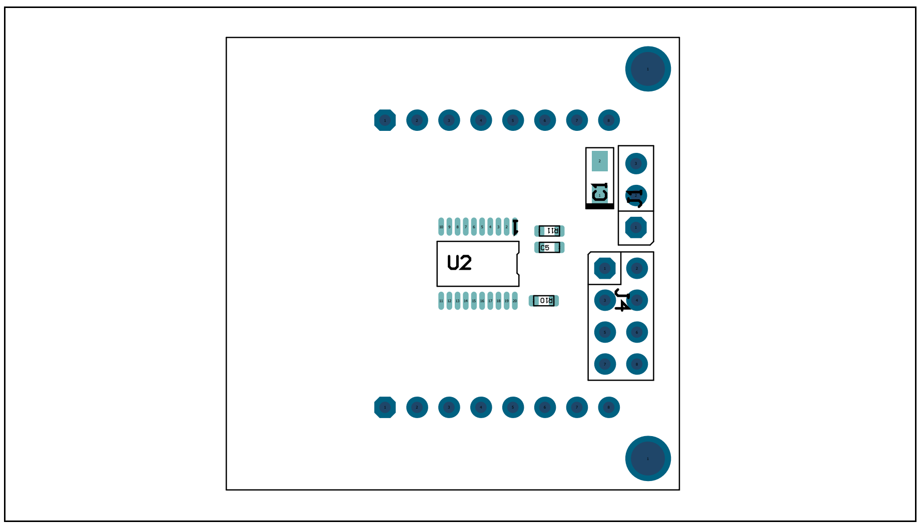
A.2 BOARD – SCHEMATICS



A.3 BOARD – TOP ASSEMBLY DRAWING



A.4 BOARD – BOTTOM ASSEMBLY DRAWING



NOTES:

Appendix B. Bill of Materials (BOM)

TABLE B-1: BILL OF MATERIALS (BOM)

Qty.	Reference	Description	Manufacturer	Manufacturer Part Number
1	C1	Capacitor, tantalum, 10 µF, 16V, 10%, 3Ω, SMD, A	KYOCERA AVX	TAJA106K016RNJ
1	C3	Capacitor, ceramic, 2.2 µF, 16V, 10%, X5R, SMD, 0603	TDK Corporation	C1608X5R1C225K080AB
2	C4, C5	Capacitor, ceramic, 0.1 µF, 25V, 10%, X7R, SMD, 0603	TDK Corporation	C1608X7R1E104K080AA
1	J1	Connector, hardware-2.54, male, 1x3, gold, 5.84MH, TH, vertical	Amphenol ICC (FCI)	68000-103HLF
1	J4	Connector, hardware-2.54, male, 2x4, tin, 5.84MH, TH, vertical	Amphenol ICC (FCI)	67996-408HLF
2	J7, J8	Connector, hardware-2.54, male, 1x8, gold, 5.84MH, TH	Amphenol ICC (FCI)	68001-108HLF
1	LABEL1	Label, PCBA, 6 x 6 mm, data matrix	ACT Logimark AS	505462
1	PCB	Printed Circuit Board	—	04-11413-R1
1	R1	Resistor, TKF, 51R, 5%, 1/10W, SMD, 0603	Panasonic®	ERJ-3GEYJ510V
2	R2, R3	Resistor, TKF, 2.2k, 5%, 1/10W, SMD, 0603	Panasonic®	ERJ-3GEYJ222V
2	R5, R6	Resistor, TKF, 0R, 1/10W, SMD, 0603	Yageo Corporation	RC0603JR-070RL
4	R8, R9, R10, R11	Resistor, TKF, 100k, 5%, 1/10W, SMD, 0603 (do not use, duplicate, use RSMT0026)	Panasonic®	ERJ-3GEYJ104V
1	U1	Microchip Analog, DAC, 8-Ch, 12-bit, MCP47CMB28-E/ML, QFN-20	Microchip Technology Inc.	**MCP47CMB28-E/ML**
1	U2	Microchip Analog, DAC, 8-Ch, 12-bit, MCP48CMB28-20E/ST, TSSOP-20	Microchip Technology Inc.	**MCP48CMB28-20E/ST**
1	U3	Microchip Analog, VREF, 2.048V, MCP1501T-20E/CHY	Microchip Technology Inc.	**MCP1501T-20E/CHY**
Mechanical Parts				
1	JP	Mechanical, hardware, jumper, 2.54 mm, 1x2 handle, gold	TE Connectivity AMP	881545-2
Do Not Populate Parts				
0	J2, J3	Connector, hardware-2.54, male, 1x1, gold, 5.84MH, TH, vertical	Samtec, Inc.	TSW-101-07-S-S

Note 1: The components listed in this Bill of Materials are representative of the PCB assembly. The released BOM used in manufacturing uses all RoHS-compliant components.

TABLE B-1: BILL OF MATERIALS (BOM) (CONTINUED)

Qty.	Reference	Description	Manufacturer	Manufacturer Part Number
0	J6	Connector, hardware-2.54, male, 2x8, gold, 5.84MH, TH, vertical	Amphenol ICC (FCI)	609-3364-ND
0	R4, R7, R12, R13, R14, R15	Resistor, TKF, 0R, 1/10W, SMD, 0603	Yageo Corporation	RC0603JR-070RL

Note 1: The components listed in this Bill of Materials are representative of the PCB assembly. The released BOM used in manufacturing uses all RoHS-compliant components.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820