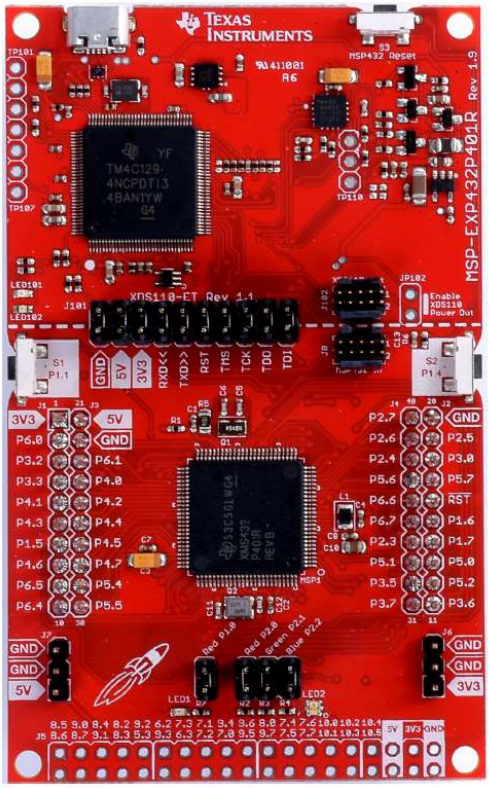
Lab 4 – Using the MSP432P401R for DSP

For the digital signal processing (DSP) labs in this course, you will be using the MSP432P401R launchpad from ELEX3305. which includes an ARM Cortex-M4F processor. At present, ARM (Advanced RISC Machines) dominates the mobile market, and ARM processor cores can be found in many smartphones.



Code development to implement the DSP algorithms will be done by writing C-code in TI's Code Composer. If you wish, you can download Code Composer from TI's website and install it onto your own PC.

The prep for this lab involves reading the lab manual and some source code prior to the lab session. You will also be asked to connect the Lab 2 filter to the MSP432P401R launchpad, and run the sample code with a sine wave input.

1. Overview

The MSP432P401R launchpad features an ARM Cortex-M4F processor running at 48 MHz.

A basic DSP system, suitable for processing audio frequency signals comprises a digital signal processor and analog interfaces as shown in figure 1. An analog to digital converter (ADC) convert the input signal to a stream of numeric samples. The DSP then processes these samples according to some defined algorithm. The result of this computation is then transferred to a digital to analog converter (DAC), where it is converted back into an analog signal.

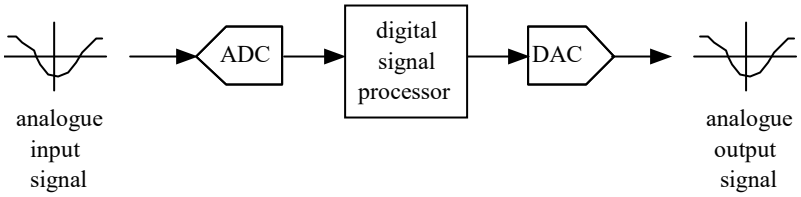


Figure 1 – Basic Digital Signal Processing System

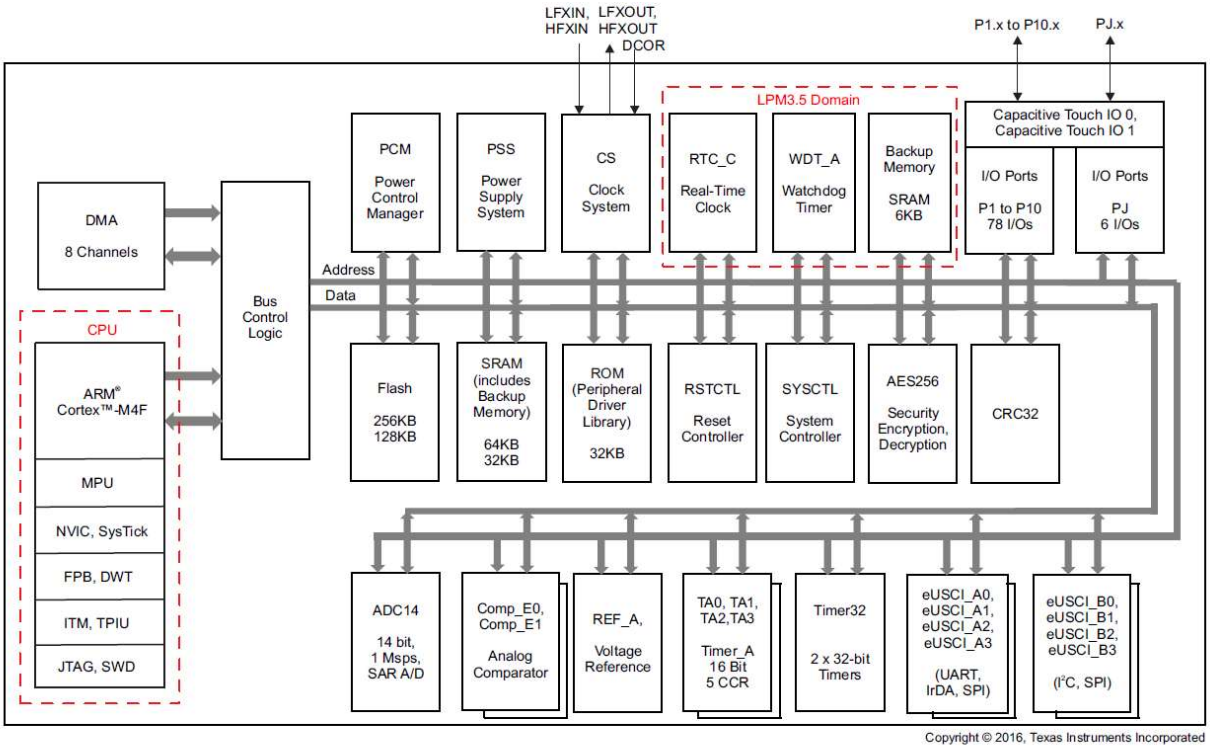
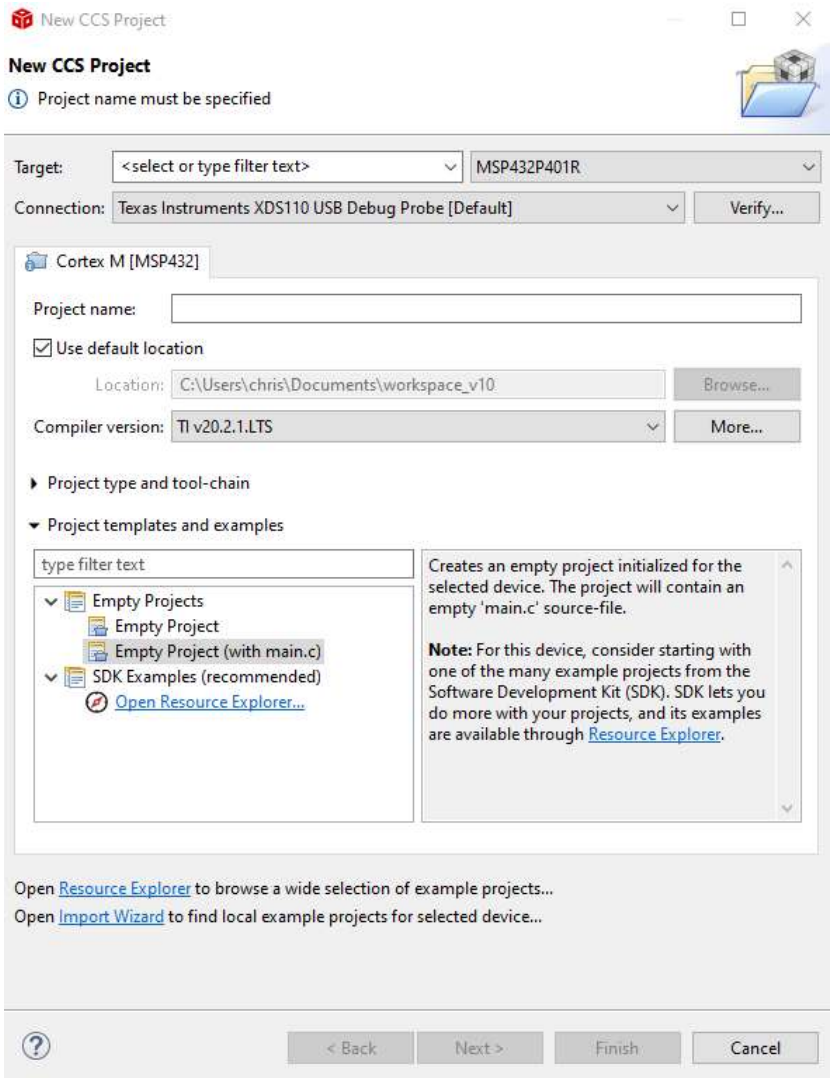


Figure 1-1. MSP432P401R, MSP432P401M Functional Block Diagram

2. Setting Up a Code Composer Project

Open Code Composer, and go to **Project → New CCS Project**:



- Select the **Target** to be *MSP432P401R*
- Enter a **Project name** (eg. X7620_Lab4)
- Click **Finish** to create project

Next, go to the project directory and replace **main.c** with the template provided to you on the Learning Hub. Then open **main.c** in Code Composer and check that the correct file is in place.

In this important step, you will change the microcontroller clock speed. From the Code Composer project, open **system_msp432p401r.c**, search for the line with the label SYSTEM_CLOCK, and modify the CPU Frequency to *48 MHz* as shown below.

```
64 /*--------------------- CPU Frequency Configuration -------------------------*/
65 //  CPU Frequency
66 //     <1500000> 1.5 MHz
67 //     <3000000> 3 MHz
68 //     <12000000> 12 MHz
69 //     <24000000> 24 MHz
70 //     <48000000> 48 MHz
71 #define  __SYSTEM_CLOCK    48000000
```

When done, go to **Project → Build All**. The compile and build should be clean with no errors. Flash the code onto your microcontroller.

2.1. Basic Verification

Using the AD2 or lab signal generator, create a *1kHz 1V_{pp} sine wave*. Since the MSP432P401R microcontroller is running on a +3.3V supply, a DC offset must be applied to the input signal. Why is this?

Connect the signal generator to the ADC input. Next, connect the MSP432P401R PWM output to the 3rd order DAC filter from lab 2. With the code loaded and everything working properly, you will see a sine wave at the filter output also. Using the oscilloscope, show the input signal on Channel 1, and the output signal on Channel 2. *Paste this screenshot into your report*.

2.2. Reading the Code

The code is commented with important parameters, so please find the following values:

- What is the value of the ADC sampling frequency?
- What is the value of the PWM frequency?

2.3. Measurements

- Using the AD2 spectrum analyzer, measure the frequency spectrum of the filter output, for a sine wave input as stated in the previous part. Show the spectrum from **100 Hz to 200 kHz**, and find the Total Harmonic Distortion (THD); see the instructions in Appendix B. *Paste a screenshot of this into your report, and quote the measured THD value*
- Using the AD2 oscilloscope, measure time domain waveform of the PWM output waveform; *paste the screenshot into your report*
- Next, measure frequency spectrum of the PWM output from **100 Hz to 200 kHz**. Note the frequency component at the PWM frequency. How much is it above the desired signal at 1kHz in **dB**? Find the THD of the PWM output, and compare it to the filtered output's THD. *Paste a screenshot of this into your report*
- Why is a lowpass filter used on the PWM output?

What to Hand In ?

- I suppose there is no particular reason for using Matlab in this case, so please write your report in Microsoft Word, name your file **Lastname_FirstInitial_Lab4.docx**, and submit it to the Learning Hub before the start of Lab 5.

Appendix A: MSP432P401R Code Template for DSP Labs

```
#include "msp.h"

//assumes system clock is at 48 MHz
//input dividers on Timer A = div 1, input to Timer A = 48 MHz
//Divide by 512 -> PWM frequency = 93.75 kHz

#define SMCLK   BIT0        //port 7.0
#define A8_IN   BIT5        //port 4.5 ADC input channel 8

#define PWM_PERIOD  512     //9-bit DAC (2^9 = 512)
#define ADCSCALE    5       //14-bit ADC; reduce by 5 bits to match DAC

#define SR_12kHz    4096
#define FSAMP       SR_12kHz //sampling frequency


void main(void)
{
    volatile unsigned int ADC_In=0;

        WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;             // stop watchdog timer

        //P7->DIR |= SMCLK;
        //P7->SEL0 |= SMCLK;

        P6->DIR |= BIT0;        //sampling flag P6.0
        P6->OUT = 0;            //for measuring sampling frequency and ADC conversion time

        P5->DIR |= BIT6;        //PWM output P5.6
        P5->SEL0 |= BIT6;
        P5->SEL1 &= ~BIT6;

        P4->SEL0 |= A8_IN;      //use analog input #8 = P4.5 for ADC input
        P4->SEL1 |= A8_IN;
```

```
        //Disable conversion while setting up ADC registers
        ADC14->CTL0 &= ~ADC14_CTL0_ENC;

        ADC14->CTL0 = ADC14_CTL0_SHP | ADC14_CTL0_SSEL__SMCLK | ADC14_CTL0_DIV__4
                | ADC14_CTL0_SHT0__16 | ADC14_CTL0_ON;

        //Memory Control Register 0 to receive ADC samples on input #8
        ADC14->MCTL[0] = 8u;

        //Sets up timer to creating ADC sampling clock
        TIMER_A0->CTL = TIMER_A_CTL_SSEL__SMCLK | TIMER_A_CTL_ID__1 | TIMER_A_CTL_MC__CONTINUOUS;
        TIMER_A0->CCR[0] = FSAMP-1;
        TIMER_A0->CCTL[0] = TIMER_A_CCTLN_CCIE;

        //Sets up timer for PWM output
        TIMER_A2->CTL = TIMER_A_CTL_SSEL__SMCLK | TIMER_A_CTL_ID__1 | TIMER_A_CTL_MC__UP;
        TIMER_A2->CCR[0] = PWM_PERIOD-1;
        TIMER_A2->CCR[1] = PWM_PERIOD/2;
        TIMER_A2->CCTL[1] = TIMER_A_CCTLN_OUTMOD_7;
        //TIMER_A2->CTL |= TIMER_A_CTL_IE;

        NVIC_EnableIRQ(TA0_0_IRQn);
        //NVIC_EnableIRQ(TA2_N_IRQn);
        __enable_interrupts();


        while (1)  {

            if ((ADC14->IFGR0 & ADC14_IFGR0_IFG0) != 0)  {
                P6->OUT = 0x00;
                ADC_In = ((ADC14->MEM[0]) >> ADCSCALE);

                TIMER_A2->CCR[0] = 0;               //disable timer
            TIMER_A2->CCR[1] = ADC_In;          //load new duty cycle value
                TIMER_A2->CCR[0] = PWM_PERIOD-1;    //enable timer
            }

        }

}


void TA0_0_IRQHandler(void)  {
    TIMER_A0->CCTL[0] &= ~TIMER_A_CCTLN_CCIFG;
    TIMER_A0->CCR[0] += FSAMP;
    P6->OUT = 0x01;
    ADC14->CTL0 |= ADC14_CTL0_ENC | ADC14_CTL0_SC;
}
```

Appendix B: Measuring Total Harmonic Distortion (THD) on the AD2

THD quantifies how much a given signal deviates from a sine wave, by comparing the magnitude of the fundamental frequency against the sum of all harmonics. Shown below is a screenshot from the internet on how to use the AD2 to measure THD.

2.4 Spectrum - *Total Harmonic Distortion (THD)*

- Connect Analog Discovery 2 to your computer and run Waveforms
- Connect W1 and 1+ to Vi (input), 2+ to Vo (output), 1-, 2-, ⏚ to ground
- Click on "Wavegen", enter the type, frequency and amplitude of the input waveform, and click on "Run"
- Click on "Welcome" tab, then on "Spectrum", then "Run"
- Output spectrum should appear on the Spectrum window as shown in Fig. 14
  - Uncheck "Trace 1" (Yellow Box)
  - Set "Stop" to $(N+1)f_i$, where f_i is the input frequency and N is the number of harmonics
  - Set "Type" to "Linear dB Average"
  - Set "Count" to 10
- Click on "View" → "Measurements", on the Measurements Panel:
  - Click on "Add", then "Trace 2"
  - Expand the "Dynamic" menu (click on the arrow)
  - Click on "THD", then "Add", then "Close"
- Total Harmonic Distortion (THD) in % can be found using the following formula:

$$THD\ in\ \% = 10^{(THD\ in\ dBc)/20} \times 100\%$$