CMPE1250
Winter 2025
v1.0

ICA 6
PLL Library

1 Introduction

Microcontrollers typically have peripherals that will allow the main clock to run at a speed other than the oscillator speed. Your microcontroller board has a 16 MHz crystal oscillator, but it doesn't normally run at that speed. In the MC9S12, this functionality is provided by the Phase-Locked Loop (PLL) module. The PLL allows us to adjust the *Bus Speed* of the microcontroller. In this assignment you will create a library to control the PLL and use an oscilloscope measurement how it effects the speed of code execution. If you need to review PLL operation, check Chapter 2 of 'Big Pink' and your notes on Brightspace.

2 Outcomes

This assignment will assess your ability to:

- Determine the correct PLL settings using the device datasheet
- Perform bitwise operations on microcontroller registers
- Create a reusable PLL library
- Incorporate a C library into a Code Warrior project
- Measure code execution speed with an oscilloscope
- Explain some factors in deciding which bus speed to use on a microcontroller

3 Assignment

In this assignment, you will be creating a library to control the bus speed of your microcontroller. The header file provided with your assignment contains most of the prototypes that you'll need for the assignment. If you choose not to implement the `Clock_GetBusSpeed()` function now, you will need to implement it when you create your SCI library.

To keep your library files tidy and well documented, it is recommended that you use the *Library Template.c* file from your GitHub repository when creating new C library files.

3.1 Blink an LED at 5 Hz

For this part of the assignment, your microcontroller will use the default bus speed of 8 MHz. Use the `while()` loop blocking delay that you developed in ICA04 to blink an LED at 5Hz. When you measure the LED with an oscilloscope, you should see a square wave with a period of 200ms. Remember that each full waveform represents two LED toggles. That means that the blocking delay is 100ms.
Fill out the Blocking Delay vs Bus Speed Table at the end of this assignment with your expected and measured values. Take a screenshot of your waveform and submit it through Brightspace.

3.2 Set the Bus Speed to 2 MHz

The *clock.h* header file contains a prototype for `Clock_Set2MHZ()`. Create a *clock.c* library file and implement `Clock_Set2MHZ()`. Add this function to the *One Time Initialization* section of your code. If it has been implemented correctly, you will see a noticeable change in the frequency of your LED. Measure the frequency of the LED and add your values to the Blocking Delay vs Bus Speed Table. Include a screenshot of your measurements in your Brightspace submission.

Your *clock.h* header file also contains a prototype for `Clock_EnableOutput()`. When properly implemented, this function will output the bus frequency on `Pin 39` of your microcontroller. Use this function to verify that your bus is running at 2 MHz.

3.3 Set the Bus Speed to 20 MHz

The *clock.h* header file also contains a prototype for `Clock_Set20MHZ()`. Implement this function in your *clock.c* file and modify your main code to have a bus speed of 20 MHz. As in the previous parts, measure the LED frequency with an oscilloscope and submit a screenshot through Brightspace. Use your measurements to fill out Table 1 at the end of this document.

Use `Clock_EnableOutput()` to output your bus speed divided by two. Measure this and submit a screenshot through Brightspace.

3.4 Set the Bus Speed to 40 MHz

You will now create a function to set the bus speed to 40 MHZ. However, your *clock.h* header file doesn't contain a prototype for this function. Before creating a function to set the bus speed to 40 MHz, you will need to add the prototype to *clock.h*. Once the prototype is added to the header file, implement it in *clock.c* and run your blocking delay with a 40 MHz bus speed. Measure the LED frequency with an oscilloscope and complete the Blocking Delay vs Bus Speed Table. Submit a screenshot of your waveform on Brightspace.

To verify that your bus speed has been set correctly, use `Clock_EnableOutput()` to output the bus speed divided by four.

3.5 Max Speed

What is the maximum bus frequency of the microcontroller? Look in appendix A of the 'Big Pink' Datasheet.
What are some reasons to not run a microcontroller at its maximum speed?
Enter your response in the text box below.

Enter your response here

3.6 Results Table

Bus Frequency	Expected Frequency	Measured Frequency	Blocking Delay Time
2 MHz			
8 MHz	5 Hz		
20 MHz			
40 MHz			

Table 1: Blocking Delay vs Bus Speed