CMPE1250 – System Clock and PLL

Your micro uses a crystal oscillator to generate an internal clock. The internal clocks drive and coordinate all activity on the micro. The **16MHz** crystal used on the board is divided by two to provide the Bus Clock at **8MHz**.

As we explore modules on the chip, we will find that the bus clock is critical to the timing and rates at which these modules operate. While an **8MHz** bus rate is not particularly slow, it can make some activities more difficult to perform, as the code can only run so fast.

Fortunately, the micro contains a PLL (phased-lock-loop) circuit that permits generation of frequencies as multiples of the oscillator clock. The technical details of how this is accomplished is beyond the scope of this course, but an analogy may help. Imagine that someone is clapping their hands a one second intervals. We can consider this the oscillator clock; a solid timing reference. If you were told to clap twice for every clap you hear, you would be able to do this after only (hearing) a few claps. You would be clapping twice as fast, using feedback from the reference clapping to sync you up such that one clap overlaps the reference, and one clap happens exactly in the middle of the reference claps.

It is possible to have the PLL circuit generate much faster clock rates than the bus rate of **8MHz**, but we will settle for a comfortable bus rate of **20MHz**. *Unlike your PC at home, you usually want to run an embedded system as slow as you can get away with. Your instructor will work this into the discussion.*

To achieve this, the clock synthesizer uses the following formula to generate the PLL clock from the oscillator clock:

$$\text{PLLCLK} = 2 \times \text{OSCCLK} \times \frac{(\text{SYNR} + 1)}{(\text{REFDV} + 1)}$$

Where, **SYNR** and **REFDV** are module registers with 6 active bits. The range of values is therefor 2^6 or 0 to 63. Note: to prevent divide by zero, whatever values are provided are added to 1.

Section 2.3.2.1 of Big Pink discusses this, and contains the following note:

NOTE

If PLL is selected (PLLSEL=1), Bus Clock = PLLCLK / 2
Bus Clock must not exceed the maximum operating system frequency.

If the bus clock is intended to be **20MHz**, and the bus clock is half the **PLLCLK**, what values can we plug into **SYNR** and **REFDV** to make the math work? Remember: **OSCCLK** is **16MHz**!

Your instructor will do the math with you, but you may have arrived at **5/4** or **SYNR = 4, REFDV = 3**. This will yield a bus rate of **20MHz** when the PLL clock is selected.

We will put this into a library and use the PLL from this point forward. We will build all other libraries and code to rely on a 20MHz bus, not the default 8MHz bus.

Every project from this point forward will call the PLL_To20MHz function in the one-time inits section. So that you don't forget this, *you should put it in your template file*!

Fortunately, the switch and LED library does not really care what rate the bus is at, so we need to make no changes there.

Create a new compilation unit, and include the following code, as discussed by your instructor.

We only need one function:

```
// bring the bus clock to 20MHz
void PLL_To20MHz (void)
{
  // PLLCLK = 2 x OSCCLK x ([SYNR + 1] / [REFDV + 1])
  // PLLCLK = 2 x 16Mhz x ([4 + 1] / [3 + 1])
  // 5/4 (1.25) * 16Mhz * 2 = 40MHz
  // bus is PLLCLOCK / 2, 40MHz / 2 = 20MHz    2.3.2.1 + 2.3.2.2
  SYNR = 4;
  REFDV = 3;
  // we could go faster, but we want to remain stable!

  CLKSEL_PSTP = 1;     // 2.3.2.6 (pseudo stop, oscillator runs in stop)

  PLLCTL = 0b11111111; // 2.3.2.7
  // monitor enable (clock is self-monitored)
  // PLL turned on
  // automatic acquisition/tracking mode
  // fast wakeup from full stop

  // can't switch to PLLCLK if (lock=0 and auto=1) - must wait for lock
  while (!CRGFLG_LOCK)
    ; // could, but shouldn't block for long

  // now that we are locked, use PLLCLK/2 for bus (20MHz)
  CLKSEL_PLLSEL = 1; // 2.3.2.6
}
```

When creating a library function like this, it is very important that you document your sources of information. In this case, sections from Big Pink are cited so that statements can be validated if something goes wrong, or the source can easily be consulted if changes are required in the future.