Here's some info so you can get started now. Please read the entire email as this lab contains a lot of information. Lab 2 has a very steep learning curve, but once you get the hang of how to navigate the user guide and datasheet it becomes a lot more enjoyable.

BRIEF HELP TO GET YOU STARTED ON LAB 2

CODE EXAMPLES BY TEXAS INSTRUMENTS

First - have you taken a look at the code examples provided by Texas Instruments? Taking a brief look at a few examples will help you get started.

1) Link: http://www.ti.com/product/MSP430FR5739/toolssoftware

2) Scroll down to the file **"**MSP430FR573x, MSP430FR572x C Code Examples (IAR and CCS) (Rev. I) (ZIP 318 KB)" under the heading **Software (13)**

3) Download the zip folder

4) Inside you will see folders with code examples in different languages. Navigate to the **C** folder. CCS will compile your **.c** code (written in C) for the MSP430, so you can write your code in C.

5) Use the **!Readme.txt** file as the table of contents to look at the different examples

You can search the table of contents for examples of how to set up the clocks.

While example code is a good way to know how to start to write, you need to understand why you are writing each statement.

- You should have two pdf documents: *"*MSP430FR57xx Family User's Guide (Rev. D)
- *"* and *"*MSP430FR573x Mixed-Signal Microcontrollers datasheet (Rev. L)*".* You will need to constantly refer to these two documents.

USER GUIDE

The user guide is a behemoth of a document at 576 pages. While this may seem like a lot when you are first getting started, you will soon come to wish this was two times as long. I'll try to show you how to find and learn the information you need, rather than me tell you everything (because I wouldn't be able to, I also need to refer to this document all the time).

Since you are trying to do *Lab exercise 1.1 - Set up MCLK and ACLK to run on XT1*, you can then navigate to the chapter **3 Clock Systems (CS).** A good idea is to read the introduction (page 71), and take a brief look at the diagram in the following page. Don't worry if you don't completely understand the diagrams, but try to briefly follow the paths in the diagram.

The introduction (p.71) describes the clocks available in the MSP430F5739. There are 4 clock (CLK) sources - The XT1, XT2, DCO, and VLO.

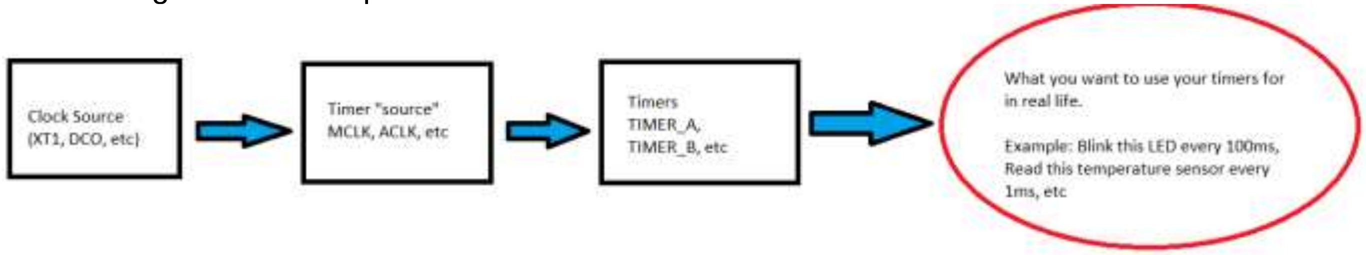
A microcontroller needs some way of keeping precise time. It can either do that using an external source, or an internal source.

Read the description of the XT1CLK. What it is saying is that, if you want to use an e**xt**ernal source for your clock between 400kHz and 16MHz (such as a crystal oscillator, which is what most non-mechanical wrist-watches run on nowadays), then you should choose XT1CLK. The name XT1CLK means "e**XT**ernal 1 Clock".

For the EXP Board you are provided for this class, there is an optional crystal oscillator on the printed circuit board (PCB) that is connected to the MSP. To use this, you will need to change some registers later described in the same chapter 3.

An alternative to using an external clock and XT1 is using the internal clock on the MSP4305739, the DCO (Digitally controlled oscillator). The DCO is located within the MSP430 chip itself. One reason you would use this instead of an external oscillator is convenience or having less parts on your PCB. A downside of the DCO is that the frequency/timing will change slightly with temperature and maybe other factors (refer to the datasheet if you want to know more).

The MCLK (master clock) and ACLK (auxiliary clock) is what you will actually use as your measure of frequency for your timers and other things later in the lab. See the rushed diagram I drew in paint below.


Clock Source (XT1, DCO, etc)
Timer "source" MCLK, ACLK, etc
Timers TIMER_A, TIMER_B, etc
What you want to use your timers for in real life.
Example: Blink this LED every 100ms, Read this temperature sensor every 1ms, etc

Generally, your clock source is at a high frequency (MHz) range, then you can divide your frequency into slower parts to use as the MCLK, and then further manipulate the time to precise intervals. To do this you will have to look at the user guide and at the diagrams inside the user guide.

DATASHEET

The datasheet, *"msp430f5739"* has a lot of information such as max clock speeds, input voltage range, etc which is important to circuit design - **but** one important part that a lot of people miss out on in order to get started is the Detailed Description

If you open the Detailed Description section, it will give you a detailed diagram for each pin, and information on which registers need to be set to what values for different situations. For example, the picture below shows pin P1.0 to P1.2. If I wanted to make P1.1 an **O**utput pin, then P1DIR **|**= BIT1. The **x** in the table refers to the pin that you are trying to access. Try to figure out what you would need to make P1.1 an input pin. Hopefully this helps you in how to read the table and diagram.

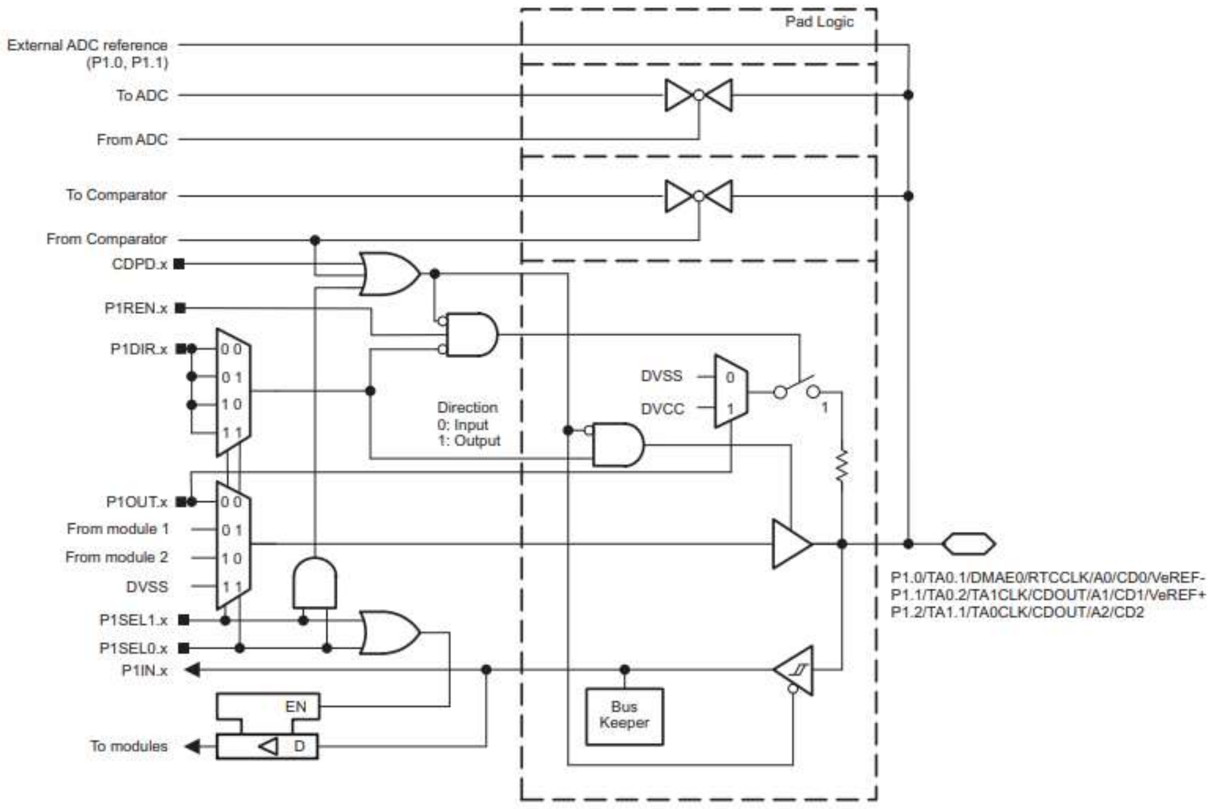


Figure 6-9. Port P1 (P1.0 to P1.2) Diagram

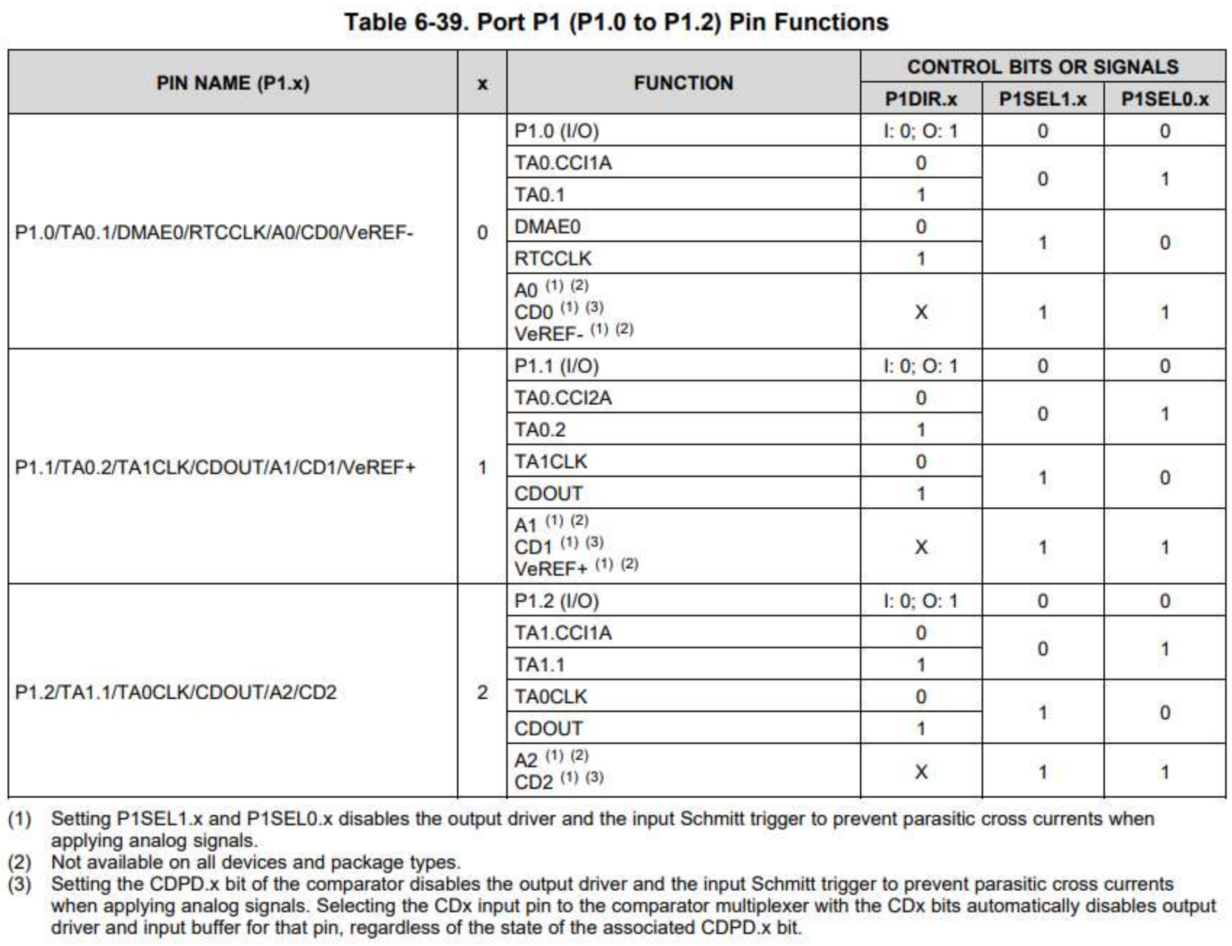


Table 6-39. Port P1 (P1.0 to P1.2) Pin Functions

PIN NAME (P1.x)	x	FUNCTION	CONTROL BITS OR SIGNALS		
			P1DIR.x	P1SEL1.x	P1SEL0.x
P1.0/TA0.1/DMAE0/RTCCLK/A0/CD0/VeREF-	0	P1.0 (I/O)	I: 0; O: 1	0	0
		TA0.CCI1A	0	0	1
		TA0.1	1		
		DMAE0	0	1	0
		RTCCLK	1		
		A0 (1) (2) CD0 (1) (3) VeREF- (1) (2)	X	1	1
P1.1/TA0.2/TA1CLK/CDOUT/A1/CD1/VeREF+	1	P1.1 (I/O)	I: 0; O: 1	0	0
		TA0.CCI2A	0	0	1
		TA0.2	1		
		TA1CLK	0	1	0
		CDOUT	1		
		A1 (1) (2) CD1 (1) (3) VeREF+ (1) (2)	X	1	1
P1.2/TA1.1/TA0CLK/CDOUT/A2/CD2	2	P1.2 (I/O)	I: 0; O: 1	0	0
		TA1.CCI1A	0	0	1
		TA1.1	1		
		TA0CLK	0	1	0
		CDOUT	1		
		A2 (1) (2) CD2 (1) (3)	X	1	1

(1) Setting P1SEL1.x and P1SEL0.x disables the output driver and the input Schmitt trigger to prevent parasitic cross currents when applying analog signals.
(2) Not available on all devices and package types.
(3) Setting the CDPD.x bit of the comparator disables the output driver and the input Schmitt trigger to prevent parasitic cross currents when applying analog signals. Selecting the CDx input pin to the comparator multiplexer with the CDx bits automatically disables output driver and input buffer for that pin, regardless of the state of the associated CDPD.x bit.

I hope this helps in getting you started in lab 2. Everything you need to program the MSP430 is listed in these two documents. Read the entire page including footnotes on the pages you are referring to - there will often be information there as well.

<u>One very important thing to keep in mind when writing firmware:</u>

"Haste makes waste" - It will be faster to write your code slowly and carefully than trying to speed your way through the writing of the code.

It is VERY frustrating to debug your C code for the microcontroller. There is no nice debugger or intelligence like in visual studio or higher-level languages. One wrong character can give you a lot of frustration and you will spend a lot of time pulling your hair out.

Every year, students come to me and say: "my code is exactly the same as my friends, but it doesn't work!". This means that your code is not the same as your friend's, and you made a mistake somewhere. Scan your code line by line and think about why that line is there. A **very** common mistake is using a equals = instead of an or-equals |=; which will ruin the intention of your code completely if it happens when you are setting up the registers in the beginning.

I hope this helps,

Samuel & Jeff