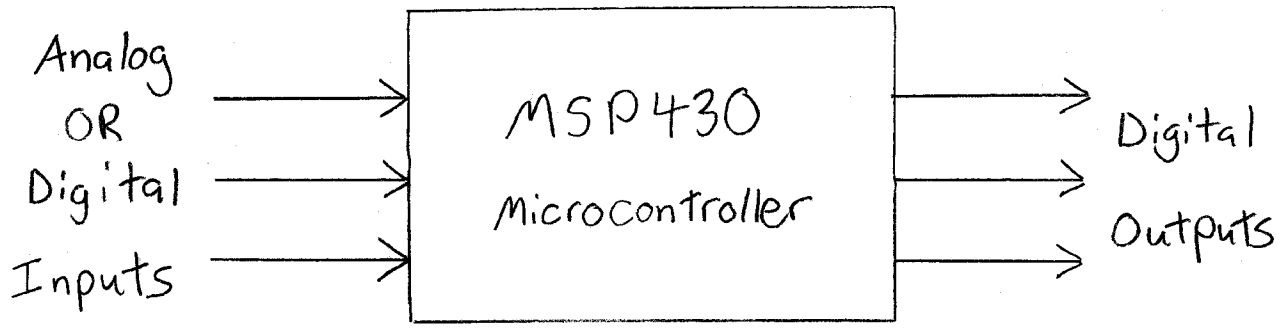
Lab III: Digital Color Organ - Code Composer Studio and the MSP430

A) MSP430



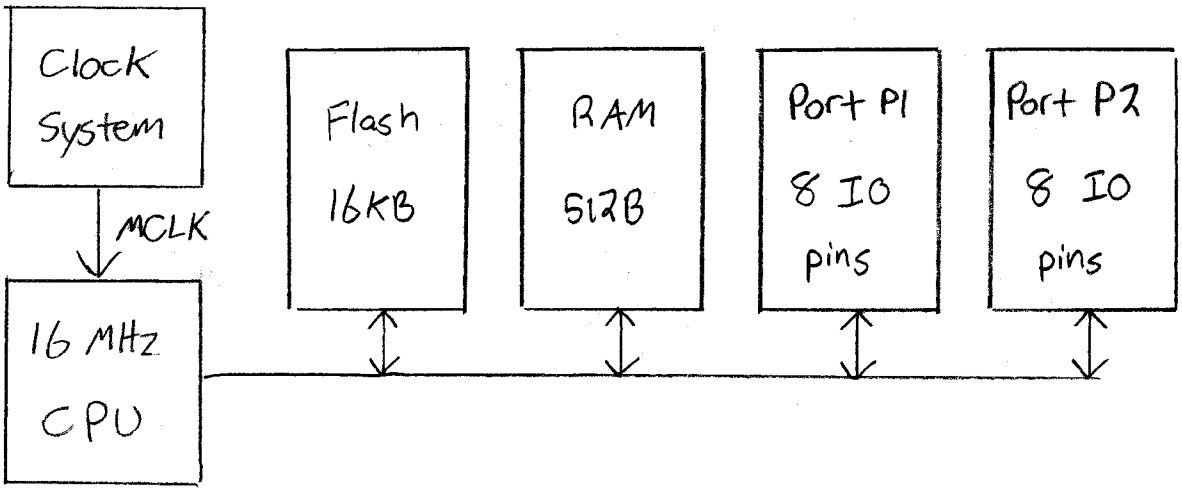
- A microcontroller contains a CPU, flash memory, RAM, and programmable input/output pins, all on the same chip.

- The Texas Instruments MSP430 adds analog to digital converters, comparators, and programmable timers at prices starting under $1.

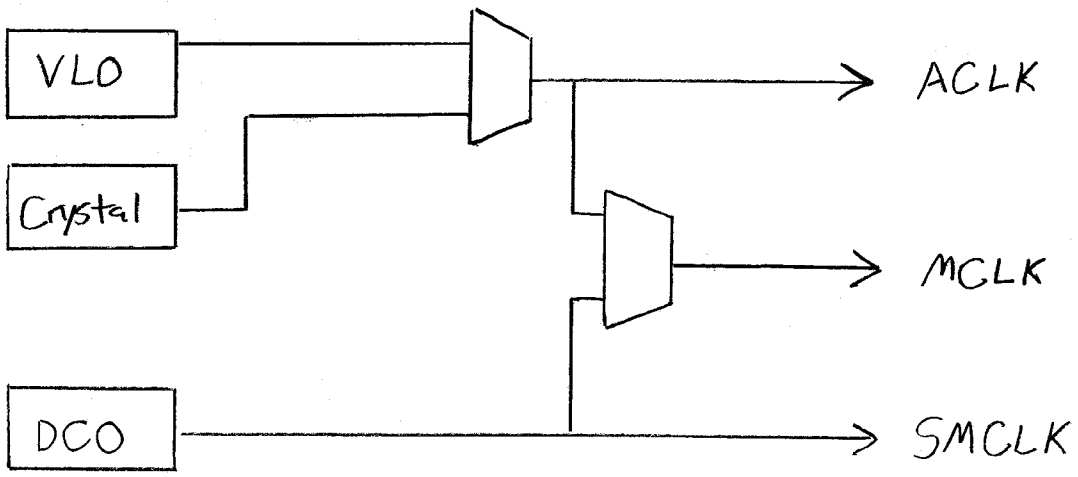
- The application is to take digital inputs (or analog inputs that are converted to digital), use them to make a decision in a C program, and create corresponding digital output.

B) MSP430G2553 Value Line Specifications

- 16-bit 16 MHz CPU
- 3 adjustable clocks (MCLK, SMCLK, ACLK)
- 16 KB Flash Memory
- 512 B RAM
- 16 General Purpose Input/Output (GPIO) pins
  - 8 Channel Comparator
  - 8 Channel ADC
- 2 16-bit programmable timers
- Universal Serial Communication Interface
  - UART, SPI, I^2C



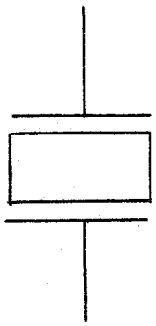
c) MSP430 Clock System



1) Very Low Power / Low Frequency Oscillator (VLO)

- 500 nA standby current
- 4 - 20 kHz (12 kHz Typical)
- Source for ACLK or MCLK

2) Crystal Oscillator



- Expensive and difficult to install
- 32.768 kHz (very accurate)
- Source for ACLK or MCLK

3) Digitally Controlled Oscillator (DCO)

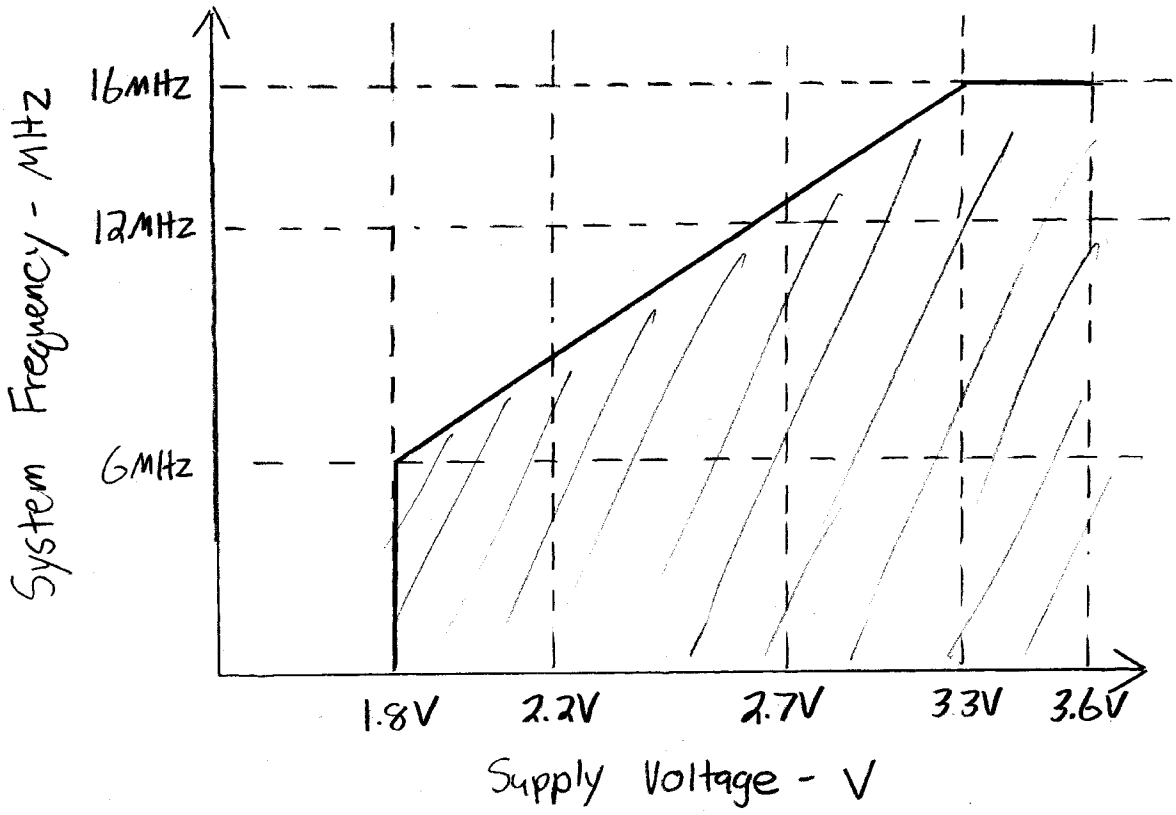
- Inexpensive and on chip
- 4 Calibrated frequencies available
  - 16 MHz, 12 MHz, 8 MHz, 1 MHz
- Source for MCLK or SMCLK

- The MCLK (which runs the CPU) is usually sourced from the DCO, although it can also be run off the VLO or the Crystal in low power or high accuracy applications, respectively.

- The ACLK and SMCLK are used to run peripherals.

- The selected clock speed has a direct effect on power usage.

D) MSP430 Low Power Operation

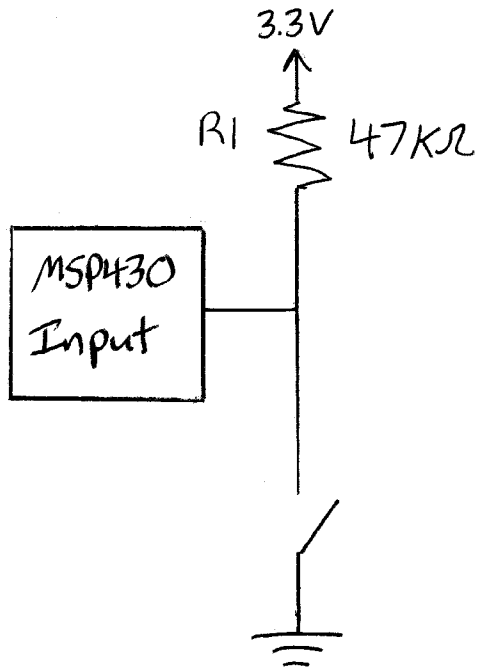


- Running at lower Clock speeds enables lower voltage operation, but limits the maximum Clock speed. Flash memory programming is not available below 2.2V

- Lower Clock Speeds Consume less Current. The general rule for the MSP430 is 250 μA / MIPS, or Million Instructions Per Second.

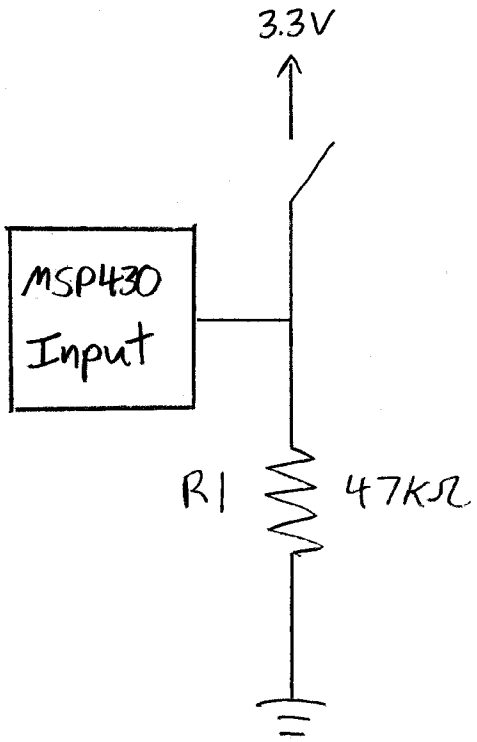
E) Pullup and Pulldown Resistors

1) Pullup Resistor



- When the switch is thrown, the input is grounded to 0V.
- When the switch is opened, the input is pulled up to 3.3V by resistor R1.

2) Pulldown Resistor



- When the switch is thrown, the input is connected to 3.3V.
- When the switch is opened, the input is pulled down to 0V by resistor R1.

F) Code Composer Studio

- The MSP430 can be programmed using industry standard tools such as IAR Integrated Workbench, however, Texas Instruments has developed their own tool for the MSP430.

- Code Composer Studio is an Integrated Development Environment, which is a tool that combines software for editing and compiling code, programming microcontrollers, and debugging code.

- All IDEs are slightly different, so familiarizing yourself with the Code Composer environment, along with learning the basic functions of the MSP430 will be the purpose of this lab.

ECE 480L: SENIOR DESIGN SCHEDULED LAB

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

MICHIGAN STATE UNIVERSITY

I. TITLE: Lab III : Digital Color Organ - Code Composer Studio and the MSP430

II. PURPOSE:

Texas Instrument's Code Composer Studio is an Integrated Development Environment (IDE), which is a software package that combines tools for writing code, programming microcontrollers, and debugging the resulting product. The MSP430 is a low cost, low power microcontroller meant for battery powered applications. The MSP430G2553 used in this lab has 16 General Purpose Input/Output (GPIO) pins, 3 programmable clocks, and can be run on as little as 500 μW of power with a 1 MHz CPU.

The concepts covered are:

1. Basic features available in an IDE.
2. Initializing registers in a microcontroller.
3. Calibrating and measuring clock signals.
4. C programming with a MSP430.
5. Adding a push button input to the MSP430.

The laboratory techniques covered are:

1. Measuring frequency and amplitude of clock signals using an oscilloscope.
2. Triggering, and setting up an oscilloscope without using autoscale.

III. BACKGROUND MATERIAL:

See Lab Lecture Notes.
MSP430G2553 Data Sheet
MSP430 User's Guide

IV. EQUIPMENT REQUIRED:

1 Your own personal Bound Lab Notebook
1 Agilent Infiniium DSO-9064A Digital Storage Oscilloscope
1 Agilent N2873A 10:1 Miniature Passive Probes
1 Soldering Iron Station

V. PARTS REQUIRED:

1 MSP-EXP430G2 LaunchPad kit
1 USB cable
1 32.768 kHz crystal
1 Female-Male 6" jumper wire

VI. LABORATORY PROCEDURE:

A) Code Composer Studio

1. Open Code Composer Studio, located at: Start > All Programs > Texas Instruments > Code Composer Studio 5.4.0 > Code Composer Studio 5.4.0.

2. You will be prompted to create a workspace; select a location that you will always be able to access when using a lab computer. Your M: drive works well for this. A workspace is simply a collection of projects (Lab III - Part A, Lab III - Part B, Lab IV for example). Each project can contain multiple .c files, but for this lab a single .c file will be all that is needed. ***Do NOT check*** the box labeled Use this as the default and do not ask again. We want to know which workspace we are using every time Code Compose Studio is started.

3. Go ahead and close the tab labeled TI Resource Explorer. We will not need this.

4. Select File > New > CCS Project. A New CCS Project Wizard should appear as shown in Fig. 1. In Project name: type Lab III - Part A. (When we move on to the next part we can create a new project, leaving the previous project easy to access if needed.)

   In the Device section, Family: select MSP430 if it is not already selected. The device we will be using is the MSP430G2553. (Double check to make sure that this is the device installed in your MSP430 LaunchPad. Again the MSP430 LaunchPad has a CMOS microcontroller on it and can be damaged by handling. Only touch the LauchPad by picking it up by the outside edges of the printed circuit board.) In Varient: click on the text box and type 2553. The text box to the right should show at least MSP430G2553. Click on this. Leave under Connection: the following TI MSP430 USB1 [Default].

   At this time, connect the LaunchPad to the USB port on your computer using the USB cable included in your ECE 480L Storage Box. The red and green LEDs on the LaunchPad should be alternating; this is the default program that is loaded onto the MSP430.

Finally, under Project templates and examples click on Empty Projects > Empty Project. The text box to the right should say Creates an empty project fully initialized for the selected device. Click Finish.

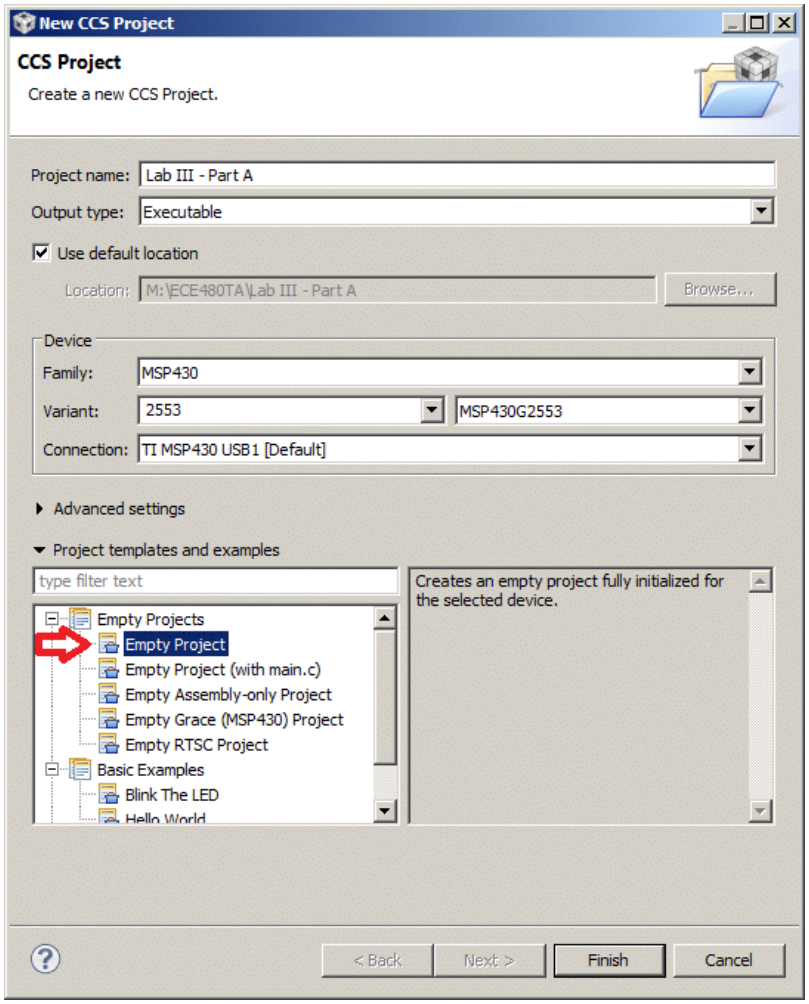


Figure 1. New CSS Project Wizard.

5. Rather than starting with a prewritten chunk of code, we will start with a blank canvas and build up a basic program line by line.

   Select File > New > Source File. In Source file: type lab3_parta.c and then click on Finish.

6. Take a few minutes to familiarize yourself with the Code Composer Studio IDE. The Project Explorer is located on the left side of the screen. (If it is minimized; simply click Restore to bring it back.) This will list all of the files contained within your project, as well as allow you to select

other projects. Notice that since we only have one project right now, it is selected as the active project by default. On the top right side of the screen you can toggle between the CCS Edit and CCS Debug perspectives. (If one of these is missing in the tab, see Fig. 2, you will have to add it by using the Open Perspective button on the tab, select Other... and pick the missing one.)

Click on CCS Debug. Switching to the CCS Debug perspective shows you the results of the debug session, and lets you step though the code line by line while watching variable, expression, or register values update dynamically.

Switch back to the edit mode by clicking CCS Edit. The green bug icon in the center of the Toolbar is the third function of an IDE that connects the Edit and Debug perspectives. This button (don't press) saves and complies your code, programs it onto the target MSP430 device, and initiates a debug session. An overview of this is shown in Fig. 2.

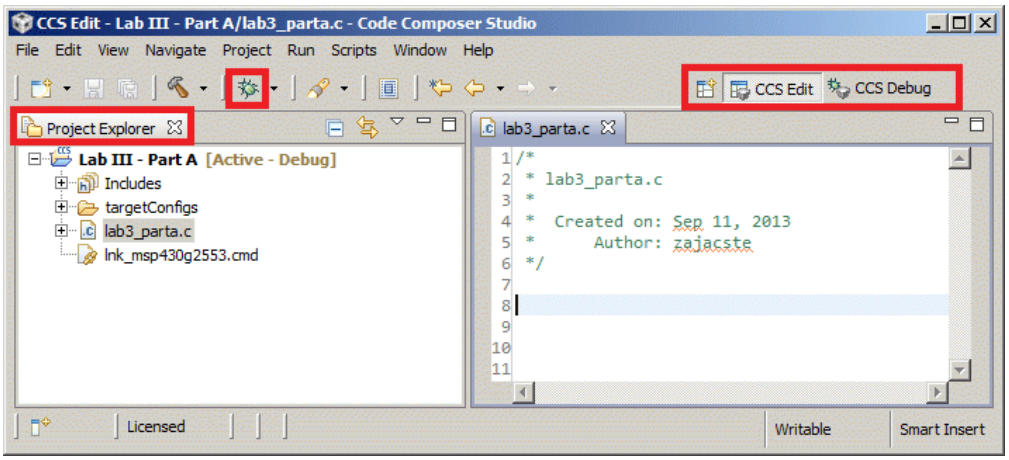


Figure 2. Code Composer Studio IDE After Step VI-A-6.

7. The first part of any MSP430 code is the header file. This file contains configuration data that is specific to the device variant being used. At line 8 in the lab3_parta.c file type the following line of code:

```
#include <msp430g2553.h>
```

8. Next, we will need to add a main function to our code. This is where the program starts every time power is applied to the MSP430. Header files and library files are usually placed outside and above this main function, and interrupts are placed outside of and below the main function. An interrupt is simply a block of code that is run only when it is requested in the main function. Copy and paste the following empty main function below the header include file:


```
void main(void)
{

}
```

9. We now have the most basic program that will run on the MSP430. Click on the Debug button (green bug icon) to compile this code and load it onto the target device. You will be prompted to save the lab3_parta.c file. Select Always save resources before launching to save time when making modifications to the code. You might get a prompt to launch the Ultra-Low-Power Advisor. Select Do not show this again and click Proceed.

10. You should automatically be switched to the CCS Debug perspective. Assuming there were no errors in your code, the Console tab at the bottom of the screen should say MSP430: Loading complete. Since there is no code in the main function, nothing should be happening on the LaunchPad. In the Toolbar of the Debug tab, click on the red square labeled Terminate to stop the debug session and return to the CCS edit perspective.

11. Next, we will start to add features to our code. The MSP430G2553 has two General Purpose Input/Output (GPIO) ports, each with 8 pins. Locate the datasheet for the MSP430G2553 on Texas Instruments web site: www.ti.com. Page 3 contains the pinout for the 20-pin PDIP package that is included with the LaunchPad. The 8 pins of Port 1 are labeled P1.0 - P1.7, and the 8 pins of Port 2 are labeled P2.0 - P2.7. Notice that almost all of the pins have several functions. The 16 pins of Port 1 and 2 default to GPIO pins, but can be set to other functions if necessary. Pin 1 (DVCC) is always power (3.3V), and Pin 20 (DVSS) is always ground (0V). On the LaunchPad, power is supplied through the USB connection. When moving a programmed MSP430 chip to a breadboard, the power connections must be made manually.

12. There are two LEDs included on the MSP430 LaunchPad. You can find these on the bottom left of the PCB; a red LED is connected to pin P1.0, and a green LED is connected to pin P1.6. In order to use these LEDs there are two things that need to be set: the direction of the pin (Input or Output), and the value of the pin (1 or 0). This can be accomplished by setting the correct registers.

13. Locate Table 15 in the msp430g2553 data sheet. Tables 14 and 15 list all of the registers that can be set in the MSP430. Scroll down to the Port P1 module. The two registers that we are interested in are the direction register (P1DIR), and the output register (P1OUT). Each register has 8 bits, which are set using hex notation. Each of the 8 bits represents 1 GPIO pin. For example, if we wanted to set pin P1.0 to a 1, we would

write 0x01 (0000 0001) to the register. Writing 0x80 (1000 0000) to the register would set pin P1.7 to a 1. Writing 0xFF (1111 1111) to the register would set all 8 pins a 1, and so on.

14. For the P1DIR register, a 1 corresponds to an output, and a 0 corresponds to an input. By default, all of the pins are set to inputs. We want to set both pins P1.0 and P1.6 to outputs, so copy the following line of code into the main function between the brackets:

```
P1DIR = 0x41;
```

15. Next, we will need to set the binary values of pins P1.0 and P1.6 to 1 to turn the LEDs on. Paste the following line of code into the main function after the first line:

```
P1OUT = 0x41;
```

16. Click the Debug button (green bug icon) to compile your new code and load it onto the target device. In the Toolbar of the Debug tab, you will have to click the green arrow button labeled Resume to run the code in debug mode. Both the red and green LEDs should light up on the LaunchPad at this time. If you get an error or the LEDs fail to light up, go back to the CCS Edit perspective and check your code. You will have to terminate the current debug session and start a new one before any changes to the code before effective. If everything looks good, click the red Terminate button in the Toolbar of the Debug tab to return to your code.

17. The next step will be to make the LEDs on the LaunchPad flash. We will start with just the green LED, so make the necessary changes to the P1DIR and P1OUT registers to turn the red LED off. Also change the = in the P1OUT line to a ^= (exclusive OR), which will toggle the output on and off.

18. We don't want the P1DIR register to be set every time the P1OUT register toggles, so place the P1OUT line inside of a while loop like the one given below:

```
while(1)
{

}
```

19. Press the debug button, but this time instead of running the code with the green Resume button, press the yellow arrow button in the Toolbar of the Debug tab labeled Step Into several times. You should be able to

watch the code running though the while loop on the Debug perspective, and watch the green LED on the LaunchPad blink every few clicks. This feature of an IDE helps identify problems in your code by slowly running through it.

20. Run the code at full speed using the Resume button. Notice that the green LED appears to be constantly on. This is because each cycle of the code runs in a fraction of a second. In order to see the LEDs flashing we need to slow things down.

21. Add the following delay loop after the P1OUT line:

```
for(delay=0; delay<10000; delay++);
```

22. Two more things need to be done before this code will run. First, we need to declare the variable delay. Add the following line of code after the header #include:

```
unsigned int delay;
```

23. We also need to add a line to stop something called the Watchdog Timer. The Watchdog Timer is an interrupt that runs by default every time the MSP430 is started. This interrupt interferes with the delay loop, so we need to turn it off. Add the following line of code at the start of the main function before the P1DIR line:

```
WDTCTL = WDTPW + WDTHOLD;
```

24. Debug your code, but do not run it yet. Instead use the yellow arrow button labeled Step Into to go through your code slowly. Notice that the delay variable listed in the top right of the screen increments each time you click step into. Run your code, and you should see the LED flashing with the delay loop included.

25. Modify the delay time to achieve a flashing rate of about 1 Hz. Assuming that each increment of the delay variable takes one clock cycle, and knowing that the program must run through the while loop twice to complete one on/off cycle of the LED, calculate the estimated clock speed of the MSP430. Record this in your lab notebook.

26. Comment your code to explain the purpose of each line. In Code Composer Studio, this can be accomplished by first typing // followed by your comment. The comments should turn green if this is done correctly. Locate your lab3_parta.c file in the folder that you selected to be your workspace. ***You will submit this as a part of your digital lab report.***

B) MSP430 Clocks

1. Before making any changes to your code, start a new CCS project titled Lab III - Part B and select Empty Project. You will have to select Lab III - Part B in the Project Explorer to make it the Active project. Select File > New > Source File. In Source file: type lab3_partb.c and then click on Finish. Copy your code from Part A into this new source file. This will allow you to go back to your old code if necessary.

2. There are 3 clocks available with the MSP430; the ACLK, the MCLK, and the SMCLK. The frequencies of these clocks can be modified to balance power usage and performance, or some of them can be turned off all together. The ACLK can run off either the Very Low power/Low frequency Oscillator (VLO) or the external crystal oscillator. The SMCLK can only run off of the Digitally Controller Oscillator (DCO). The MCLK can be run off of any of these three sources. See pages 3 - 4 of the lab lecture notes for an explanation of all the clocks.

3. By default, the MSP430 CPU runs of the MCLK sourced by the DCO. We can setup certain GPIO pins to output the different clock signals. This will allow us to measure the frequency and voltage of each clock using the oscilloscope.

4. Refer back to the pinout of the MSP430G2553 included in the datasheet, and find out which pins can be setup to output the ACLK and SMCLK. It is not possible to output the MCLK directly, but by default, the MCLK and the SMCLK are set to the same frequency. By measuring the SMCLK, we can determine what frequency the MCLK is running at, and thus the frequency that the CPU is running at.

5. In order to setup the two pins identified in the last step to output the clock signals, the P1SEL register must be modified. Find this register in Table 15 of the msp430g2553 datasheet. Setting the value of both pins in this register to a 1 will do the trick. Add this line of code after the P1DIR line.

6. We must also modify the P1DIR line to setup these two new pins as outputs. Remember that pin P1.6 still needs to be setup as an output to keep the green LED flashing. Converting all of this information into the Hex value that P1DIR needs to be set to is more difficult than it needs to be.

7. The header file included in this program contains a section of code to make this easier. When turning on both the red and green LEDs in the previous part, we used the line of code P1OUT = 0x41; We can also use the format P1OUT = BIT6 + BIT0; The header file will then covert this

into the Hex format required by the MSP430.

8. Feel free to change the values of the P1DIR, P1SEL and P1OUT registers to use this new technique. If you are comfortable working with Hex, you can leave the register values as is. Both methods work just as well.

9. We will need to make one more change to the code to select the VLO as the source for the ACLK. Locate the User's Guide for the MSP430. You should be able to find this in the same location that you found the msp430g2553 datasheet. The User's Guide is a 600+ page document that explains all of the features of the MSP430 microcontroller. Browse through Chapter 5 on the Basic Clock Module. Section 5.3 contains information on all of the registers that set up the clock system. We want to turn off the crystal LFXT1 to enable the VLO for the ACLK. Find the register that needs to be set, and add this line to your code. (Hint: only one bit in the Basic Clock System Control Register 3 needs to be set).

10. After making the necessary changes to P1DIR, P1SEL, and BCSCTL3, debug and run your new code. The green LED should be flashing at 1 Hz as before, and the red LED should be constantly on. If the red LED does not turn, or if it is very faint, you will have to spend more time reading through Chapter 5 to understand how to set the Basic Clock System registers. Remember that the ACLK output pin is the same pin that the red LED is connected to, so when the ACLK output pin is correctly set, there should be a 50% duty cycle square wave driving the LED.

11. Turn on the Infiniium Oscilloscope and log in. Start out by pressing the **Default Setup** button. This will restore all of the settings on the oscilloscope to factory default. Connect channel 1 of the scope to P1.0, which we setup to output the ACLK. Use a Female-Male 6" jumper wire connected to one of the two GND pins on J6 connector (lower right side) to connect the scope probe ground clip. **Do NOT try to make the GND connection without this jumper wire. Shorting the scope probe ground clip to the VCC connection on J6 connector will blow a fuse.**

    Adjust the horizontal scale of the scope to display one to two periods of the clock signal. The vertical scale for channel one should be OK at 1 V/Div. You will need to adjust the trigger level knob to achieve a stable waveform. Make sure that the trigger source is set to channel 1, and adjust the trigger level such that it is in the middle of the clock signal. Notice what happens when the trigger level is not within the clock signal. You can also gently press the trigger level knob in to automatically set the trigger level to the middle of the waveform. Once the clock signal is stable, you can turn on averaging by selecting **Setup > Acquisition > Averaging > Enabled**. If autoscale ever fails to find

a signal, this is the way to manually find the signal.

12. Using the toolbar measurements, measure the frequency and peak-to-peak voltage of this clock signal. Notice that there is some overshoot and ringing on the rising and falling edge of the clock signal if you zoom in with the Horizontal scale. This makes the peak-to-peak measurement inaccurate. You can either count divisions, or use the manual markers to get the true peak-to-peak voltage of the clock signal.

13. Make a hard copy of the screen (***select invert waveform colors)*** including all measurements. Since we are printing in black and white, you will need to mark which node is which on your printout.

    Mark this section letter and number on the plot. Give the plot an appropriate title. Attach as indicated in the Lab Report.

14. Repeat steps VI-B-10 thru VI-B-13 for the SMCLK (P1.4 of the MSP430).

15. One of the benefits of the MSP430 is the ability to adjust the frequency of the clocks to best suit your project. We will now change both the ACLK and the SMCLK.

16. The ACLK was originally sourced by the VLO, which is a very low power, but also low accuracy clock source. We can switch this to a more accurate clock source by using the included crystal. Locate the crystal oscillator included in your parts kit. This is a very small surface mount part which is difficult to solder.

    To help you here are some tips

a. First, locate the pad Q2 that you will be soldering the crystal to.

b. There are three connections that need to be made; XIN, XOUT, and the case ground. Open the crystal package and notice that the two pins are bent towards one side. The crystal and two pins should lay flat on the board without bending the pins. This is how you determine the polarity of the crystal.

c. Use a small piece of tape (located by the parts cabinet in the lab) to tape the case to the Q2 pad with the XIN and XOUT pins lined up. This will hold the crystal in place while you solder the two pins.

d. Apply the smallest amount of solder possible to connect each pin to the board, remove the tape, and finally solder the case of the crystal to the ground pad. Refer to Fig. 3 for a picture of a crystal attached to the MSP430 LaunchPad.

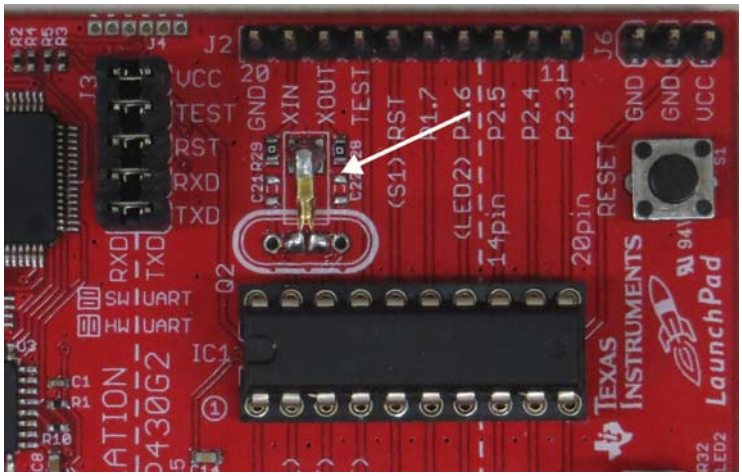


Figure 3. Crystal attached to the LaunchPad

e. If you have any difficulty with this process, see either the course TA, or one of the technical staff in the ECE Shop.

17. Remove the line of code that you added in step VI-B-9 to set the BCSCTL3 register. This will reset the ACLK source to the crystal instead of the VLO.

18. The frequency of the DCO, which controls the MCLK and SMLCK, can be switched between 4 different calibrated frequencies. These are listed on page 30 of the msp430g2553 data sheet. The two registers that need to be changed are the Digitally Controlled Oscillator Control Register (DCOCTL), and the Basic Clock System Control Register 1 (BCSCR1). Insert the following two lines of code directly after the line that stops the Watchdog Timer (WDTCLT), to load the calibration constants for 16 MHz into these two registers:

```
DCOCTL = CALDCO_16MHZ;
BCSCTL1 = CALBC1_16MHZ;
```

19. Debug and run this new code, and if everything is correct, the CPU should now be running 16 times faster than before. Instead of flashing once per second, the green LED should now be flashing 16 times per second. If not, go back and check your code for errors.

20. The MSP430G2553 comes with 4 calibrated frequencies to run the DCO: 1 MHz, 8 MHz, 12 MHz, and 16 MHz. These can be selected simply by changing the calibration constants that are loaded into the control registers mentioned in step VI-B-17. You can also select frequency dividers of 1, 2, 4, or 8 by setting the corresponding register, which you can find in section 5.3 of the User's guide. Finally, page 29 of the data

sheet also lists a range of uncalibrated DCO frequencies that can be used.

21. Now that the ACLK has been switched to the Crystal and the SMCLK has been switched from 1 MHz to 16 MHz, repeat steps VI-B-10 thru VI-B-13 for these two new clock signals. Be sure to differentiate between the ACLK sourced by the VLO and the ALCK sourced by the Crystal, and the two different frequencies of the SMLCK sourced by the DCO. In total you will print 4 scope plots for this section.

22. Comment your code for Part B as you did in Part A. Start collecting your .c files in a folder. ***You will submit a digital lab report with your code.***

C) Adding Inputs to the MSP430

1. In the previous two sections, the LED outputs were controlled solely by the code that was programmed into the MSP430. We can also use external inputs to the MSP430 in the code to control the outputs.

2. At the most basic level, an input can be a push button switch. The MSP430 LaunchPad includes a push button switch attached to pin P1.3 that we can use.

3. Start a new CCS project titled Lab III - Part C and select Empty Project. You will have to select Lab III - Part C in the Project Explorer to make it the Active project. Select File > New > Source File. In Source file: type lab3_partc.c and then click on Finish. Copy your code from Part B into this new source file.

   We can start out by deleting the P1SEL line since we do not need to measure the clock signals anymore. We can also remove BIT6 and BIT4 from the Port 1 Direction Register. We want pin P1.3 (BIT3) to be set as an input. Remember that all pins are defaulted to inputs at power up, so no further change to the P1DIR register is necessary.

4. Delete the contents of the while loop. We are going to start out by switching the red LED on and off by using the push button switch as an input.

5. First, we need to enable the internal pullup resistor for the pin P1.3. When the button is pressed, the pin is connected to ground (0 or an off state), and when the button is released the pin is left floating. We want it to be connected to VCC (1 or an on state). This is what the pullup resistor is for; it pulls the pin up to VCC.

6. Section 8.2.4 of the User's Guide tells you how to enable the pullup/pulldown resistor. Section 8.2.2 tells you how to switch between a pullup and a pulldown resistor. If the pushbutton switch were connected to VCC instead of ground, we would need a pulldown resistor to pull the pin down to ground when the button was released.

7. Set the registers P1REN and P1OUT to enable the pullup resistor for pin P1.3.

8. Add an if/else statement inside the while loop. We can use the push button as an input to switch between two lines of code. When the button is not pressed, pin P1.3 in the P1IN register will be a 1 due to the pullup resistor that we enabled. The following statement will be true when the button is not pressed, and can be used as the test criteria for the if statement:

```
P1IN & BIT3
```

9. To turn off the LED, we can use the following statement:

```
P1OUT = ~BIT0;
```

10. To turn the LED back on, we have to use a slightly different statement:

```
P1OUT |= BIT0;
```

11. Use the previous two lines of code in your if/else statement such that the red LED is off when the button is not pressed, and the red LED turns on when the button is pressed.

12. Modify your code to have the red LED flash at a rate of 1 Hz when the button is not pressed, and have the green LED flash at a rate of 1 Hz when the button is pressed. You will have to use what you learned about setting register values and delay loops in Part A. You might also have to adjust the clock source to achieve the flashing rate that you are looking for.

13. Take a color photo of your LaunchPad next to your solderless board and capture the red LED on. The resolution must be good enough to see your board number and the red LED. Email the photo to wierzba@msu.edu. State your name in the email, the title of the lab experiment and which section of ECE 480 your are enrolled in. This is due on or before the due date of this Lab Report.

13. Again, remember to comment your code to explain the purpose of each line. This will be the last .c file that is included in your digital lab report.

D) Clean Up

Do not remove the scope probes from the scope. Turn off all equipment. Put all of your things back into your Storage Box and take it home with you.

Assemble your lab report, staple it and hand it in as described in the ECE 480 Lab Syllabus. Please read and sign the Code of Ethics Declaration on the cover.

Assemble a digital lab report. This report is a collection of files. Rename your files with your last name, lab number, and title. Format the file exactly as follows: lastname_lab3_parta.c, where lastname is replaced with your own last name. You should have three .c files from sections VI-A, VI-B and VI-C. Place these files in a folder named lastname_lab3. Again, replace lastname with your own last name. ***Not following this format will result in a lowering of your lab report grade***.

Zip this folder by right clicking on it and selecting 7-Zip > Add to lastname_lab3.zip. Email the resulting file to zajacste@msu.edu. Make sure to include the exact terms ***ECE 480*** and ***Lab 3*** in the subject, otherwise the email might get lost. Add this declaration of ethics to the email message: All of the attached work was performed by me. I did not obtain any information or data from any other student. I will not post any of my work on the World Wide Web.

This is due as indicated in the ECE 480 Lab Syllabus.

VII. ASSIGNMENT FOR NEXT LAB PERIOD

1. Continue to work on your Digital Color Organ Final Report.

2. Listen to the next recorded lab lecture and read the Lab Procedure portion of that experiment.

Lab Report

Lab III : Digital Color Organ - Code Composer Studio and the MSP430

Name: . . . . . .

Date: . . . . . .

Code of Ethics Declaration

All of the attached work was performed by me. I did not obtain any information or data from any other student. I will not post any of my work on the World Wide Web.

Signature . . . . . .

VI-B-13

Mark VI-B-13 on the top right side of your plot and attach as the next page. Give the plot an appropriate title.

VI-B-14

Mark VI-B-14 on the top right side of your plot and attach after VI-B-13. Give the plot an appropriate title.

VI-B-21a

Mark VI-B-21a on the top right side of your plot and attach after VI-B-14. Give the plot an appropriate title.

VI-B-21b

Mark VI-B-21b on the top right side of your plot and attach after VI-B-21a. Give the plot an appropriate title.

