SACRAMENTO STATE

MARIO PALACIOS
LAB COURSE: CpE 185 – SECTION 03
MONDAY (6:30PM – 9:10PM)
LAB 02: MULTICORE PROPELLER MICROCONTROLLER
INTRUCTOR: SEAN KENNEDY

Introduction:

In this lab we will be learning how to apply code to a propeller microprocessor through a program called SimpleIDE. In parts one and two, we use already made code, in order for us to understand how the propeller microprocessor functions. We are encouraged to change the code and see the various ways it can be written. In part three it allows us to see how the breadboard on the microprocessor can function with the code, and see how it can run different codes at the same time using its different cores. Lastly in part four we get to combine our newly gained knowledge and are able to create a design of our own.

Part 1. Software, Language, Digital I/O

Description: The goal of this part is to familiarize ourselves with the software and hardware. Along with the common programming and circuit tasks that can be completed on the propeller microprocessor. For this part of the lab I connected a pushbutton to the breadboard and displays for how long you hold onto the button.

Engineering Data:

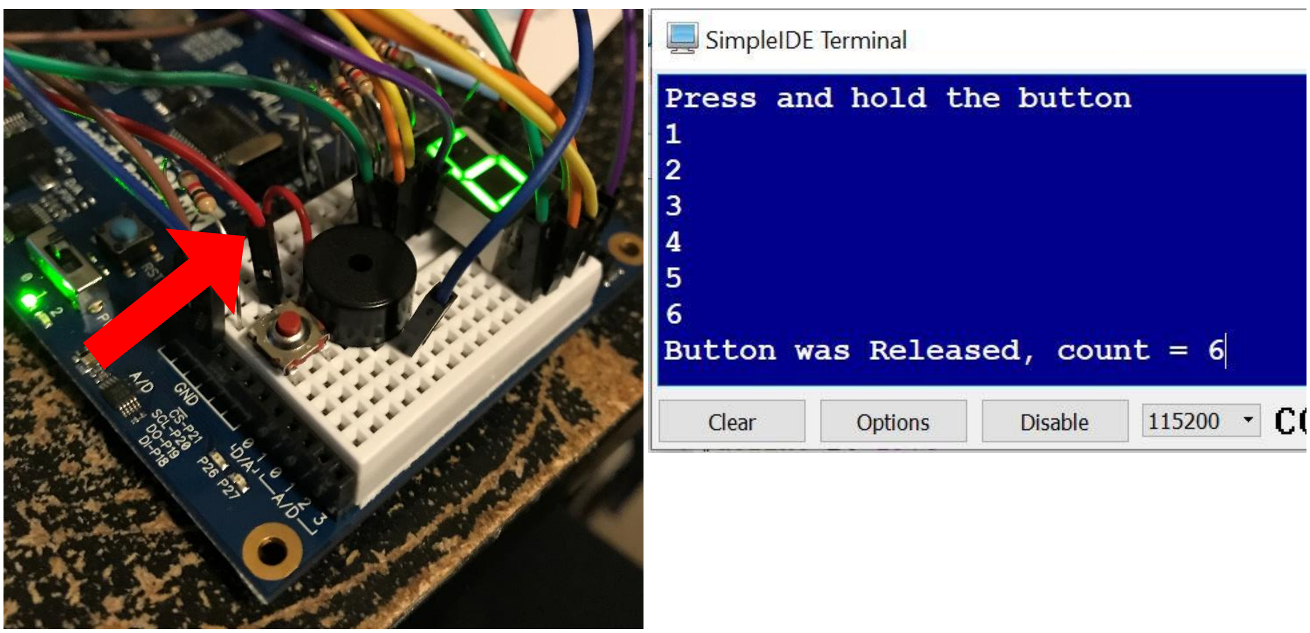


Figure 1. Propeller Microprocessor Connected to a Pushbutton (Left Image) and the Terminal of When it is Held (Right Image)

Conclusion: The program SimpleIDE is a user friendly program that has a vast library that can be used to fully understand how to properly use the Propeller microprocessor.

Part 2. Analog and Timed I/O

Description: In this part we will get introduced to topics such as; analog to digital and timed input/output techniques. We will used these techniques to send outputs to transducers and actuators.

Engineering Data:

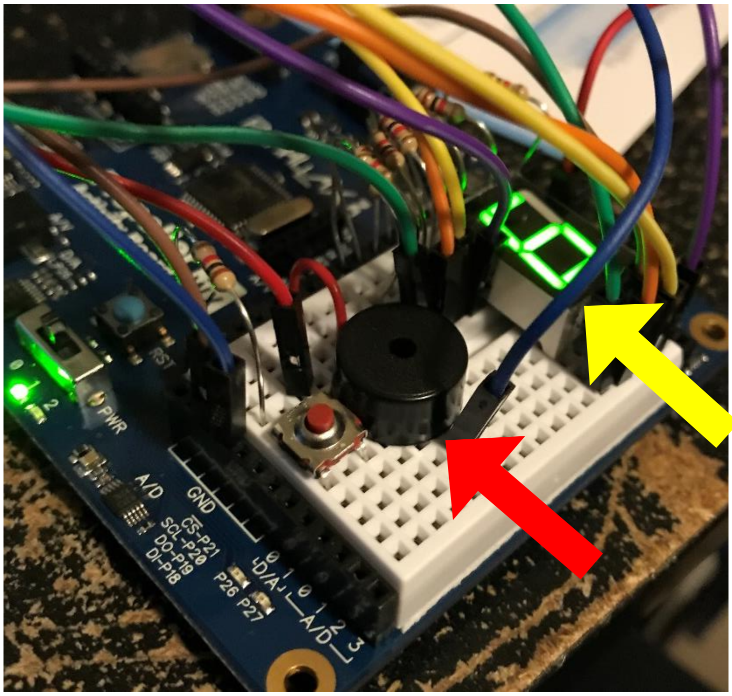


Figure 2. Piezospeaker Transducer Producing Tones (Red Arrow) and Seven Segment Display That Uses Parallel Outputs to Display Digits and Numbers (Yellow Arrow)

```
#include "simpletools.h"
#define C6 1047
#define D6 1175
#define E6 1319
#define F6 1397
#define G6 1568
#define A6 1760
#define B6 1976
#define C7 2093
int note[]
{C6,D6,E6,F6,G6,A6,B6,C7};

int main()
{
 for(int i=0; i<4; i++)
 {
  freqout(4, 150, note[0]);
  pause(45);
 }
 freqout(4, 500, note[2]);
 pause(100);
 freqout(4, 500, note[2]);
 pause(100);
 freqout(4, 500, note[3]);
 pause(100);
 freqout(4, 500, note[3]);
 pause(100);
 freqout(4, 500, note[4]);
 pause(100);
 for(int j=0; j<5; j++)
 {
  freqout(4, 250, note[6]);
  pause(25);
  freqout(4, 300, note[7]);
 }
}
```

Figure 3. Code for Piezospeaker Transducer that Produces Tones

Conclusion: Overall this part of the lab allowed us to work with an output you can see or hear. It was interesting to see how the software works effortlessly with the hardware.

Part3. Multicore Approaches, Libraries, and Counter Modules

Description: The goal of this part is to familiarize ourselves with the microcontroller cores execute function code. Using a core will allow execute code parallel with the program thread.

Engineering Data:

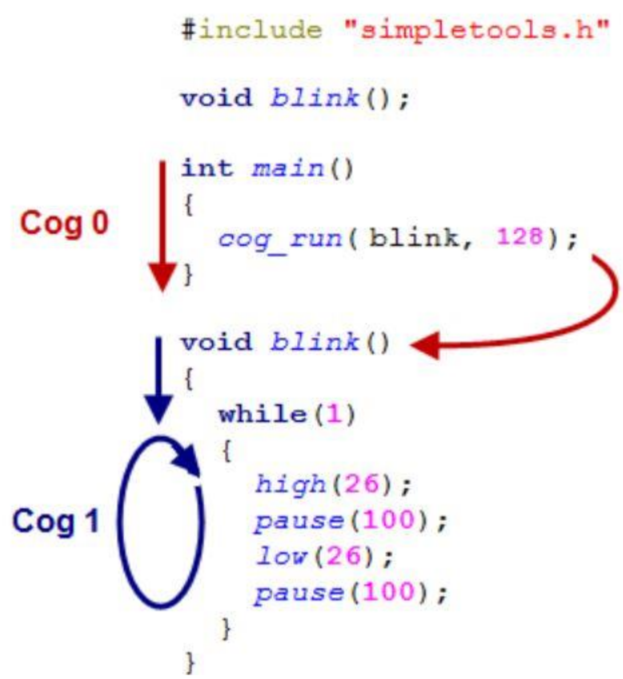


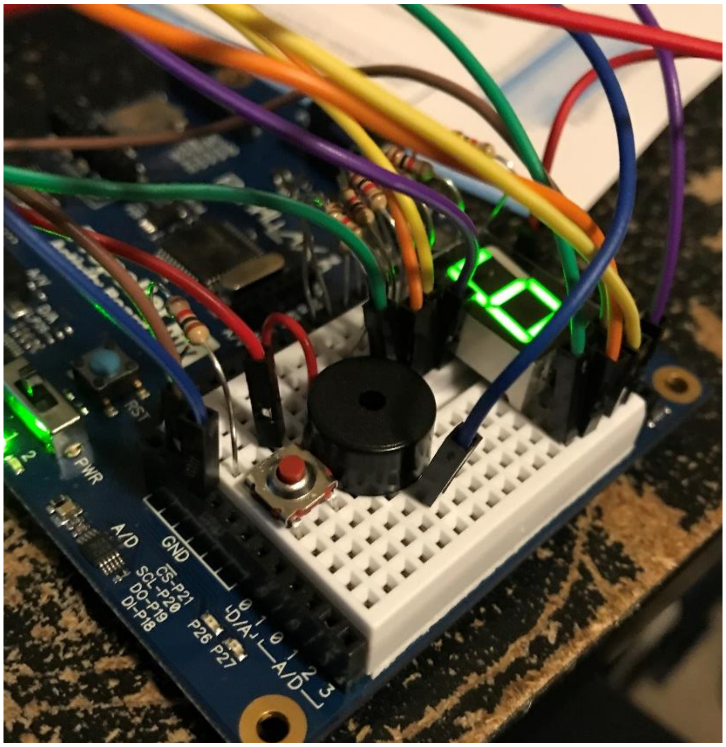
Figure 4. A Simple cog_run Function that Executes Cog0 Causing an LED to Blink in Cog1

Conclusion: Using multicores is a useful approach to programming microprocessors because it will allow the user to run multiple programs parallel with its main. This is important since some hardware will need to execute different parts of a whole program.

Part 4. Design Project

Description: The goal of this part is combine parts and techniques from the previous sections to make an application. It must incorporate an input, output, and have a multicore approach.

Engineering Data: For this part I designed a program that would play the melody in part two at the same time and indefinitely; when you are pressing and holding a button that will display its count on the seven segment display.




```
int main()
{
    cog_run(noise,128);
    int number = press();
    display(number);
}
```

Figure 5. Board Layout of the Propeller Microprocessor Along with the Main Thread of Code Used to Execute Part Four Design

Conclusion: Having everything work together was exciting because of how simple yet complex this microprocessor works. In the end I did learn more in detail what a Multicore Approach is and how to apply it in code.

Final Conclusion:

In conclusion this lab was straightforward, the one part I did have some trouble understanding was the multicore approach. I understood how it works but did not know how to type it into code form. However I did eventually learn how to apply it into code. Seeing how everything can work simultaneously together was an interesting topic, and overall can be useful in later projects.