ASEE's 123rd Annual • Conference & Exposition • New Orleans, LA • June 26-29, 2016

New Orleans JAZZED about Engineering Education

Paper ID #14668

Programmable System-On-Chip (PSoC) Usage in an Engineering Technology Program

Mr. Stephen A. Strom, Pennsylvania State University - Erie

Stephen Strom is a lecturer in the Electrical and Computer Engineering Technology department of Penn State Behrend, and holds a B.S. in electrical engineering from Carnegie Mellon University. His career includes over thirty years experience in designing and programming embedded systems and has multiple patents for both hardware designs and software algorithms

Prof. David R. Loker, Pennsylvania State University - Erie

David R. Loker received the M.S.E.E. degree from Syracuse University in 1986. In 1984, he joined General Electric (GE) Company, AESD, as a design engineer. In 1988, he joined the faculty at Penn State Erie, The Behrend College. In 2007, he became the Chair of the Electrical and Computer Engineering Technology Program. His research interests include wireless sensor networks, data acquisition systems, and communications systems.

Programmable System-On-Chip (PSoC)
Usage in an Engineering Technology Program

Abstract

Part of the requirements for courses in an Electrical and Computer Engineering Technology (ECET) program includes the ability to design, analyze and debug electrical as well as computer circuits. Typical courses over multiple semesters include analog electronics, digital electronics, C/C++ programming, instrumentation and measurement systems, communications systems, and microcontrollers. Each of these courses also includes a lab component that provides students with hands-on projects which are intended to reinforce the lecture material.

The goal of this paper is to demonstrate that a Programmable System-On-Chip embedded board (PSoC 5LP) can be used as a basis for multiple lab projects in these courses. The intent is to also include examples of the types of lab projects that can be used within these courses. Possible lab projects include: function generator, audio collection, digital processing, serial communication, on-off keying modulation, wireless sensor network, and robot position control. For each lab, project documentation (e.g., engineering requirements, flowchart, schematic, results, and student assessment) is provided. Student assessment consists of results from a survey containing questions regarding approximate time, level of difficulty, areas of difficulty, and prerequisites needed to complete project, and questions regarding level of interest in PSoC devices as a results of this project, suggestions for improvement, and any other comments. Each lab project will be mapped to one or more applicable courses in an ECET program. Additionally, each lab project will be mapped to one or more ABET ETAC student outcomes (including program specific criteria) for direct assessment of these outcomes.

Introduction

In an Electrical/Computer Engineering Technology degree, there are many classes that use microprocessors/microcontrollers as part of their curriculum. The format for these classes are similar (in curriculum) in that their end goal is to teach 'C' or 'C++' programming, as well as embedded hardware and applications. Most computer courses employ a large amount of hands-on lab material and selecting a proper embedded processor can greatly improve the outcome/success of the course. In general, the preference is to use a processor that has:

- Wide industry acceptance (usage after graduation). This allows the students to leverage their knowledge into better/more advanced job positions.
- Development tools that are quickly installed and are easy to use. There are always questions about compiling, downloading and debugging, and it is important to have local (and online) help tools that can provide solutions to common problems.
- Cost of development tools, boards, kits. The goal is to have the students purchase their own components. Students that own their own board take better care of them, and are more likely to use them in other projects.

In addition to these requirements, a microcontroller also needs to fulfill the outcomes required by the curriculum of the course. An intermediate programming class would need to teach both 'C' programming as well as microprocessor hardware/peripherals and also algorithms. To teach a wide range of subject matter, it is necessary to examine:

- The performance of the microprocessor
- The amount of flash and ram memory (within the processor)
- Number of I/O pins (digital and analog)
- Internal circuits (digital, a-to-d, pwm, uart, spi, etc.)
- Timers and Interrupts

The goal of this paper is to show how the PSoC 5LP can be used successfully in an intermediate embedded systems course[1].

Course Overview

The Computer Engineering Technology curriculum is set up so that programming and embedded circuit design is taught via a series of courses: (a) Introduction to 'C' programming, (b) Digital Design and Embedded Systems, (c) C++ and object orientated programming and (d) an Intermediate Embedded Systems course. After completing this series, the students can continue on with an Embedded Linux course, as well as a Software Engineering (process and agile development) course.

This arrangement of courses is a bit unusual in that Technology departments do not always offer embedded systems designs to this depth; however, it is one that really prepares the student to succeed as an embedded programmer/designer. But with this said, success in a curriculum does not always relate to the number of courses available, and often it is the content and delivery that meters the student's progress.

The beginning (or entry level) computer course uses an "Arduino" style computer board to teach 'C' programming and embedded hardware. As this is an introductory course, all of the hardware interfaces are accomplished via API/library calls. Students learn 'C' programming and with the available hardware libraries, and can (for the most part) ignore the low-level register bit manipulations that are actually occurring. The final lab project is to incorporate the computer board plus motor and sensing circuits into an autonomous robot.

The intermediate embedded course is more in depth, where the students build their own computer (PCB) board, learn to surface mount components and design software that interfaces directly to the hardware. Students need to understand manufacturer's datasheets and learn how to use 'C' to directly address the microcontroller's internal status and control registers. Topics such as processor interrupts, (digital) input debouncing and A-to-D filtering are also covered. Each lab builds on the knowledge gained from the previous applications (and homeworks), and the final lab project is of their own choosing.

The last course is in embedded Linux, where the students learn to use operating system calls, plus threads, fork, shared memory, semaphore, sockets, and pipes. In addition to the operating system aspects of applications, students get more depth to their ‘C’ programming studies in learning data structures and pointers. Class projects include algorithms such as queuing, sorting and binary search techniques.

This was the first year that the PSoC board was used, and it was added to the embedded Linux course as a test case. Students had to learn and use the development tools, understand the board’s capabilities and design a final lab all within three weeks. Normally, (if this was the intermediate course) additional lecture time would be used to cover all of the components of the controller in a more sequential and detailed manner, but this was an experiment to see how fast a group of students could pick up on a new set of development tools.

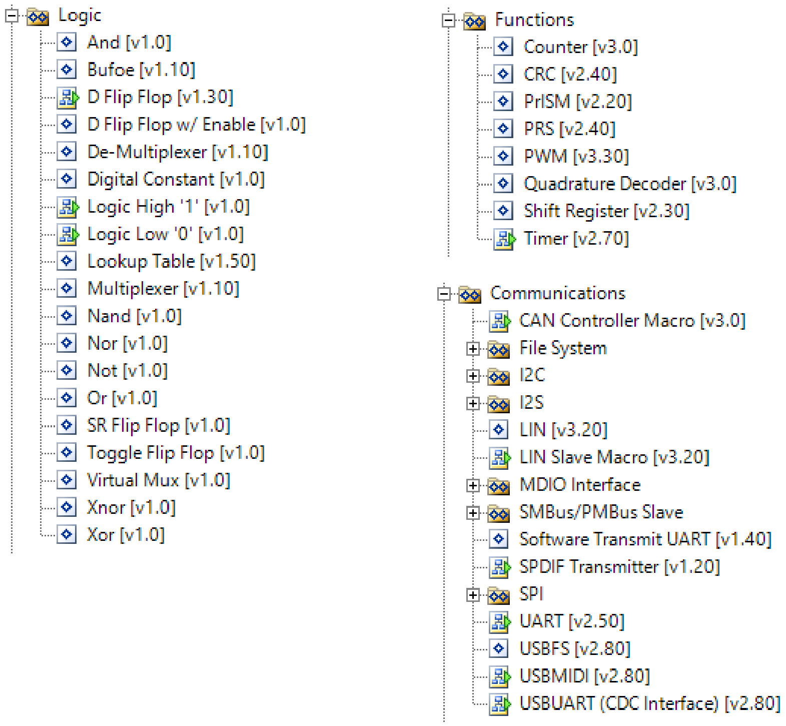
PSoC 5LP Board / Development Environment

The PSoC 5LP board includes a 32-bit ARM Cortex-M3 core. In addition, there is a configurable gate array that includes both digital as well as analog peripheral circuits. Users can select simple logic gates such as: AND, OR and XOR, more complex logic functions like flip-flops, shift registers and counters, as well as communication components like UART’s and SPI circuits.

To create a design that includes logic/analog operations all integrated with ‘C’ code, PSoC Creator (from Cypress Semiconductor) is used. This is a windows based development tool that can perform the concurrent hardware/schematic entry as well as the software compilation, download and debugging.

There are three main tool windows within a PSoC Creator project – (1) the hardware schematic, (2) the pin connection mapping and (3) the software development/debugging.

Beginning with the hardware schematic, all of the components can be selected from simple user menus/pick-lists. An example of the types of logic functions that are available can be seen in figure 1.



- Logic
  - And [v1.0]
  - Bufoe [v1.10]
  - D Flip Flop [v1.30]
  - D Flip Flop w/ Enable [v1.0]
  - De-Multiplexer [v1.10]
  - Digital Constant [v1.0]
  - Logic High '1' [v1.0]
  - Logic Low '0' [v1.0]
  - Lookup Table [v1.50]
  - Multiplexer [v1.10]
  - Nand [v1.0]
  - Nor [v1.0]
  - Not [v1.0]
  - Or [v1.0]
  - SR Flip Flop [v1.0]
  - Toggle Flip Flop [v1.0]
  - Virtual Mux [v1.0]
  - Xnor [v1.0]
  - Xor [v1.0]

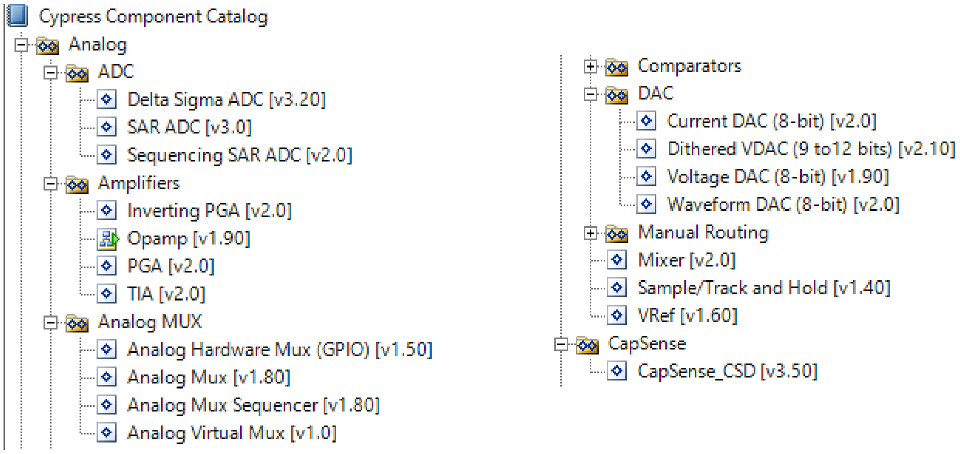
- Functions
  - Counter [v3.0]
  - CRC [v2.40]
  - PrISM [v2.20]
  - PRS [v2.40]
  - PWM [v3.30]
  - Quadrature Decoder [v3.0]
  - Shift Register [v2.30]
  - Timer [v2.70]

- Communications
  - CAN Controller Macro [v3.0]
  - File System
  - I2C
  - I2S
  - LIN [v3.20]
  - LIN Slave Macro [v3.20]
  - MDIO Interface
  - SMBus/PMBus Slave
  - Software Transmit UART [v1.40]
  - SPDIF Transmitter [v1.20]
  - SPI
  - UART [v2.50]
  - USBFS [v2.80]
  - USBMIDI [v2.80]
  - USBUART (CDC Interface) [v2.80]

Figure 1 – PSoC Creator Digital Logic Functions

This is a fairly complete list of digital components, which allows for the design of complex logic circuits without the need for external gates. If the rest of the PSoC architecture is ignored, the programmable logic array could be easily used for the lab portion of a digital logic course.

In addition to the logic circuits, there are analog peripheral components, such as: OpAmps, D-to-A circuits, A-to-D and an analog input mux. An example of the types of analog circuits that are available can be seed in figure 2.



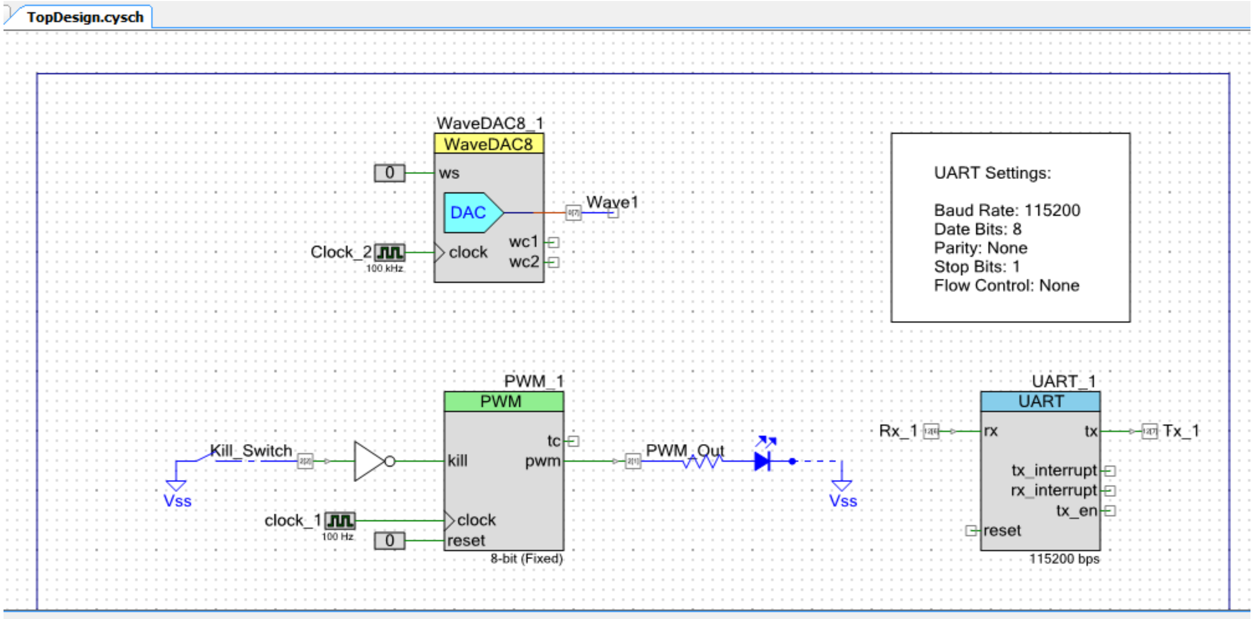
- Cypress Component Catalog
  - Analog
    - ADC
      - Delta Sigma ADC [v3.20]
      - SAR ADC [v3.0]
      - Sequencing SAR ADC [v2.0]
    - Amplifiers
      - Inverting PGA [v2.0]
      - Opamp [v1.90]
      - PGA [v2.0]
      - TIA [v2.0]
    - Analog MUX
      - Analog Hardware Mux (GPIO) [v1.50]
      - Analog Mux [v1.80]
      - Analog Mux Sequencer [v1.80]
      - Analog Virtual Mux [v1.0]
    - Comparators
    - DAC
      - Current DAC (8-bit) [v2.0]
      - Dithered VDAC (9 to12 bits) [v2.10]
      - Voltage DAC (8-bit) [v1.90]
      - Waveform DAC (8-bit) [v2.0]
    - Manual Routing
    - Mixer [v2.0]
    - Sample/Track and Hold [v1.40]
    - VRef [v1.60]
  - CapSense
    - CapSense_CSD [v3.50]

Figure 2 – PSoC Creator Analog Circuits

While it is a huge advantage of having analog circuits embedded in the programmable array, the only drawback is that if resistors or capacitors are needed (for an amplifier/filter circuit), then the components will need to be connected externally to the board.

Schematic Entry:

The technique of building a design is to start with the schematic and one part at a time, select a component and bring that "element" onto the schematic page. As components are added to the project, they get wired together (and comments added). An example of this is seen in figure 3.



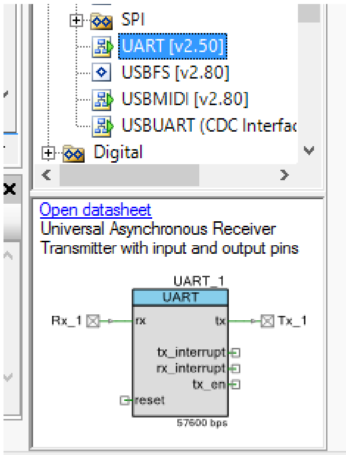
UART Settings:

Baud Rate: 115200
Date Bits: 8
Parity: None
Stop Bits: 1
Flow Control: None

Figure 3 – PSoC Creator Schematic Entry Page

Once wired, the part may need to be initialized and/or configured. For example, some parts like "clocks" can be configured to auto-start at a specific frequency, or a PWM starting up with a known duty cycle, or a UART with a specific baud rate. This is accomplished by double-clicking on the schematic part and via series of menus, select the starting configuration.

Once configured, the parameters (duty cycle, pulse timing, baud rate, etc.) can be changed via software. If the part is configurable (for example a Uart is configurable, a NAND gate is not), then PSoC Creator will insert a set of 'C' source files that contain the API calls for that component. Once included in the project, the application can use these API calls to initialize and/or adjust the parameters of the part. To learn what 'C' function calls are available and how the part works, Cypress has included help information in PSoC Creator. For every component, there is a corresponding datasheet that details the operation and available API calls. An example of how to bring up the UART datasheet is shown in figure 4.



Universal Asynchronous Receiver Transmitter with input and output pins

Figure 4 – Component with Datasheet Selector

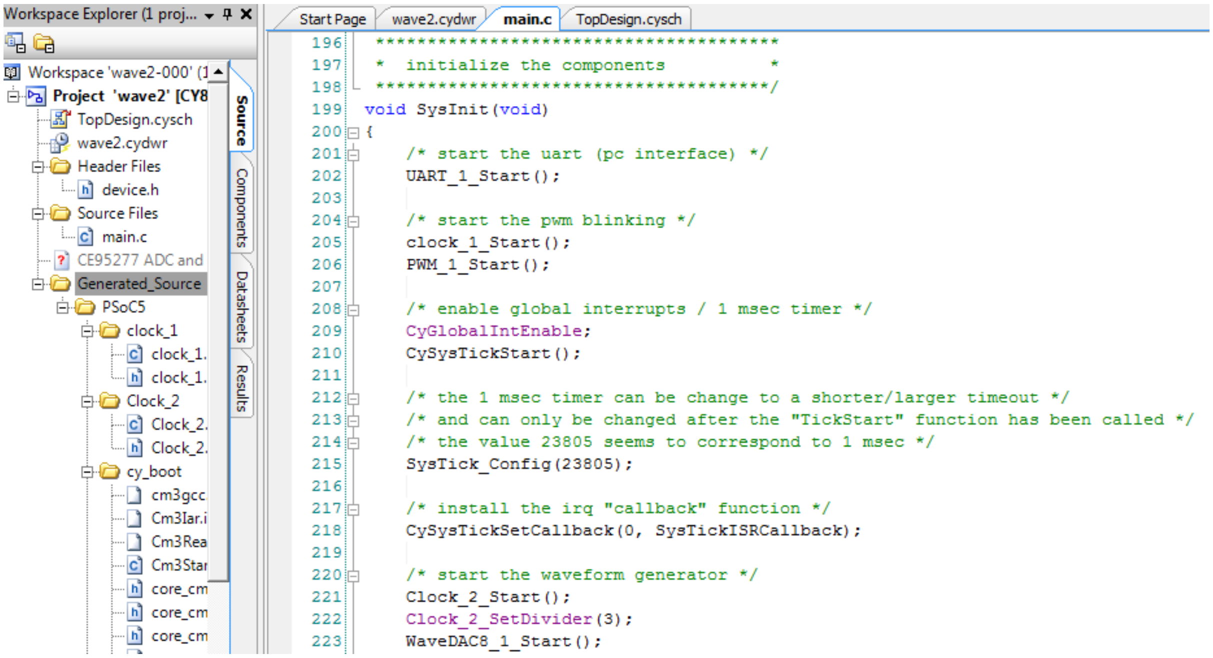
‘C’ Source Entry:

The entry point into the program is the same as in any ‘C’ program (main). The code is created in a similar manner to any embedded project -- Start with an initialization routine and followed by repetitively calling a main loop (see figure 5):

```
void main( )
{
    Initialization( );

    for ( ; ; )
    {
        MainLoop( );
    }
}
```

Figure 5 – C Code (main) Example

Once the hardware has been brought onto the schematic page, the corresponding ‘C’ libraries for that part are also attached to the project and will get included and compiled into the application. The ‘C’ source page will display a series of included files (see figure 6).



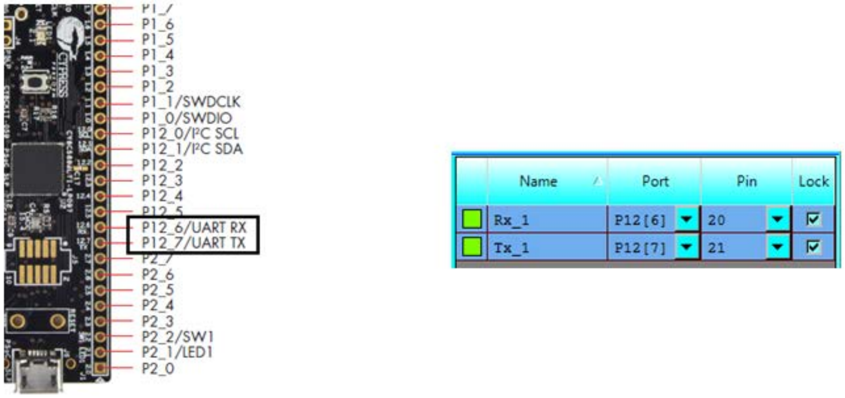
```
 ****************************************
 *  initialize the components           *
 ****************************************/
void SysInit(void)
{
    /* start the uart (pc interface) */
    UART_1_Start();

    /* start the pwm blinking */
    clock_1_Start();
    PWM_1_Start();

    /* enable global interrupts / 1 msec timer */
    CyGlobalIntEnable;
    CySysTickStart();

    /* the 1 msec timer can be change to a shorter/larger timeout */
    /* and can only be changed after the "TickStart" function has been called */
    /* the value 23805 seems to correspond to 1 msec */
    SysTick_Config(23805);

    /* install the irq "callback" function */
    CySysTickSetCallback(0, SysTickISRCallback);

    /* start the waveform generator */
    Clock_2_Start();
    Clock_2_SetDivider(3);
    WaveDAC8_1_Start();
```

Figure 6 – PSoC Creator 'C' Source Page

Note: The source files are part of the project, but the API functions within them are not automatically called; it is up to the user to add initialization/configuration calls as appropriate. For example, pulling in an A-to-D converter (into the schematic) still requires the user to initialize the part (ADC_DelSig_1_Start), and then start the A-to-D conversion (ADC_DelSig_1_StartConversion). Specifics of what needs to be done in documented in the component's datasheet.

External Pin Mapping:

Once the hardware components have been brought into the project, the next step is to wire the external I/O pins (on the PSoC 5LP board) to the "wires" in the schematic. When the I/O pins are brought onto the schematic page, each wire gets a unique name. In the wiring page, this wire name is then "mapped" to the physical pin on the edge of the board. An example of this is shown in figure 7.



	Name	Port	Pin	Lock
🟩	Rx_1	P12[6]	20	☑
🟩	Tx_1	P12[7]	21	☑

Figure 7 – Internal to External Wire Mapping

Once the pins are mapped, the project is ready to compile and download on to the PSoC board.

While there are a lot of steps to creating a project, Cypress provides many tutorials as well as example projects to speed up the “startup” process and shorten the learning curve in using PSoC Creator.

The way that the class was introduced to PSoC Creator was to use a portion of the Lab time and have each of the students walk through a basic digital lab and prove that they can monitor an input pin and set the state of an output pin. Once they had successfully created a project, compiled and downloaded on to a board, the students were ready to tackle a more complex goal.

Students were given a list of possible lab projects and each student was allowed to pick their own project. There were students that designed digital sampling circuits, amplifiers, PWM controls as well as digital logic circuits.

Lab Projects:

The end goal for this paper was to examine the possibility of incorporating the PSoC 5LP board into the ECET curriculum, and to come up with a series of labs that could be used in a variety of classes. The Electrical/Computer Engineering Technology curriculum included several classes where the board could be easily integrated into both the class and lab work:

Embedded Systems Design – This is the basic course for computer programming and microcontroller circuit designs.

Logic Systems – An introductory course into digital design.

Digital Signal Processing – A course focused on the mathematics of digital filters and FFT analysis.

Wireless Communications – This spans everything from Infra-Red to RF communications.

Below is a series of lab’s that could be used for each of these courses, an example of what the hardware circuitry could appear as and a list of suggested ABET ETAC outcomes[2].

Embedded Systems Design – PWM LED Control

In this lab, the student’s goal is to create software that controls an RGB (tri-color) LED. Each LED will be enabled via its own PWM circuit. In addition, the students need to wire in a Joystick that can be used as an input to the program (adjusting the joystick will alter the brightness of each LED).

One possible circuit for this lab is shown in figure 8.

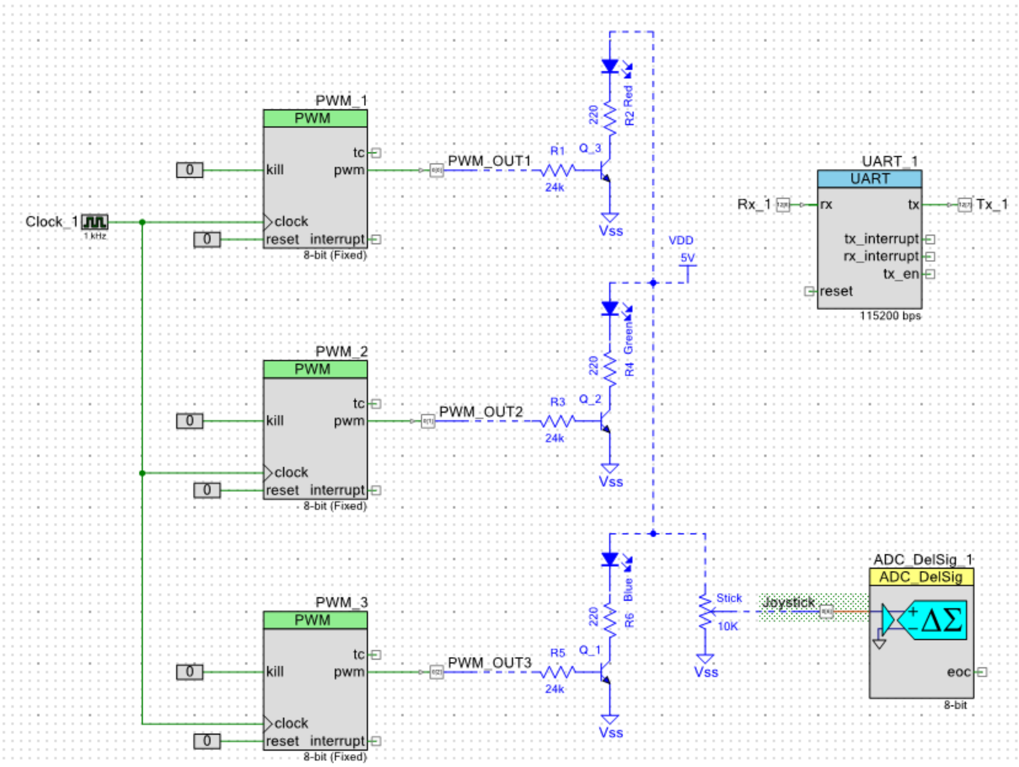


Figure 8 – PWM Control Circuit

In this circuit, the external LED's (and transistors) are shown in blue. The joystick is connected into an A-to-D converter and is read by the processor. The PSoC 5LP on-board PWM circuits control the brightness of the LED's by adjusting their duty cycle.

Once completed, this lab could be used for a variety of ABET ETAC outcomes, two that are suggested include:

Student Outcome-A
Apply the knowledge, techniques, skills, and modern tools of the discipline to narrowly and broadly defined engineering technology activities.

Student Outcome-B
Apply knowledge of mathematics, science, engineering, and technology to engineering technology problems that require the application of principles and applied procedures or methodologies.

Logic Systems – Pinewood Derby Lab

In this lab, the student's goal is to create software that monitors the start/finish of a two-lane pinewood derby track. This is the basic two car (cub-scout) race in which the cars traverse a 30 foot track and the first car across the finish line wins. The circuit needs to be designed so that the finish line sensors clock a flip-flop and latch the results. Without a latch, it is possible that the processor might miss seeing a car cross the finish line.

One possible circuit for this lab is shown in figure 9.

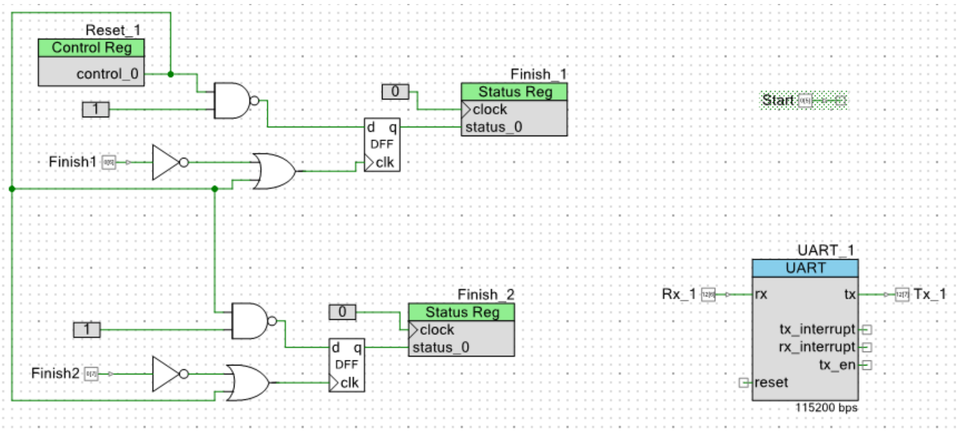


Figure 9 – Two Lane Pinewood Derby Monitor

In this circuit, the external sensors are connected to the input pins: Finish_1 and Finish_2. A start signal is brought into the input pin: Start. The way that the board communicates the results is by sending the data out the RS-232 port (through a USB cable) over to the PC.

Once completed, this lab could be used for a variety of ABET ETAC outcomes, two that are suggested include:

Student Outcome-D
Design systems, components, or processes for broadly-defined engineering technology problems appropriate to program educational objectives.

Program Specific (CMPET and EET) Student Outcome-A
Application of circuit analysis and design, computer programming, associated software applications, analog and digital electronics, microcomputers, operating systems, local area networks, and engineering standards to the building, testing, operation, and maintenance of electrical, computer and software systems.

Digital Signal Processing – Audio Filtering

The goal is to filter (in real time) audio data. The user needs to design a microphone circuit that in turn passes in the analog data to the on-board's A-to-D circuits. The processor then samples this input (at 20,000 samples per second), processes the results and outputs a new (filtered) signal.

It was suggested that the students implement a second-order digital filter, using the approach that is detailed in figure 10.

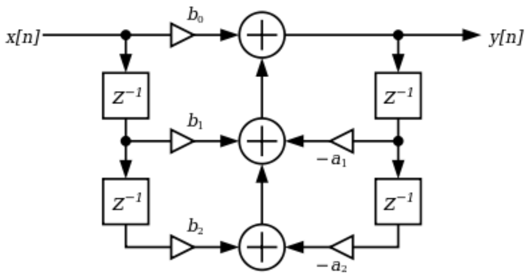


Figure 10 – Second Order Digital Filter

The students were asked to create a filter that would pass frequencies in the range of human speech, and reject everything else. One approach is to create a bandpass filter using two separate (digital) filters – a high pass and a low pass. The coefficients for the feedback paths would be based on the frequencies selected for the lab (and implemented in software).

One possible circuit for this lab is shown in figure 11.

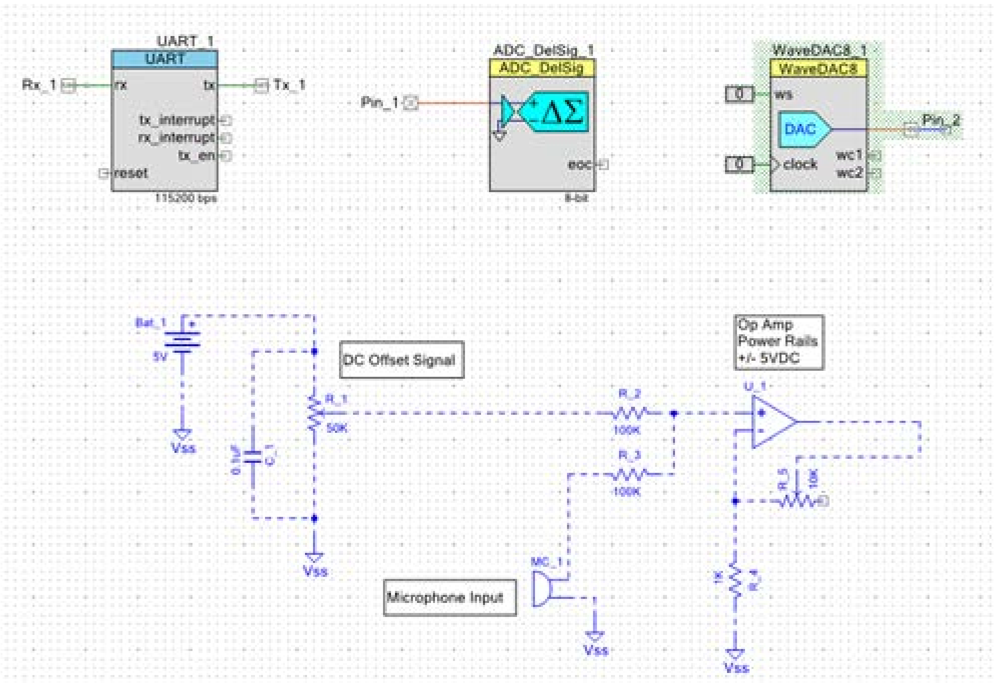
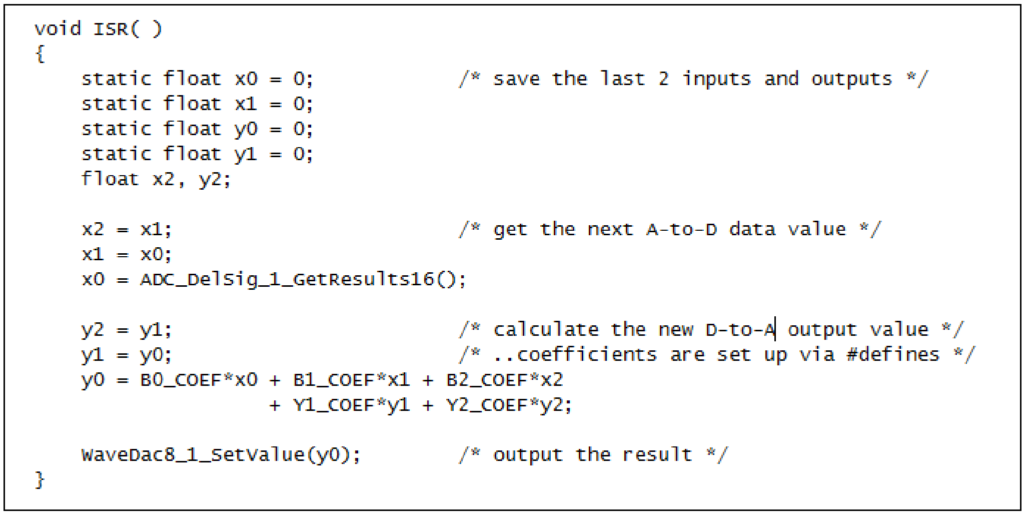


Figure 11 – Digital Filter Lab

In this circuit, the external microphone circuit is shown in blue. The analog signal is brought on to the board via Pin_1 and the output waveform is sent to Pin_2. A Uart was added to the design so that debug/analysis information can be sent back to the PC.

The source code for this can be implemented entirely within the interrupt service routine (assuming that the hardware is initialized in the main loop). An example is shown in figure 12.



```
void ISR( )
{
    static float x0 = 0;            /* save the last 2 inputs and outputs */
    static float x1 = 0;
    static float y0 = 0;
    static float y1 = 0;
    float x2, y2;

    x2 = x1;                        /* get the next A-to-D data value */
    x1 = x0;
    x0 = ADC_DelSig_1_GetResults16();

    y2 = y1;                        /* calculate the new D-to-A output value */
    y1 = y0;                        /* ..coefficients are set up via #defines */
    y0 = B0_COEF*x0 + B1_COEF*x1 + B2_COEF*x2
                + Y1_COEF*y1 + Y2_COEF*y2;

    WaveDac8_1_SetValue(y0);        /* output the result */
}
```

Figure 12 – Digital Filter Source Code Example

Once completed, this lab could be used for a variety of ABET ETAC outcomes, three that are suggested include:

Student Outcome-B
Apply knowledge of mathematics, science, engineering, and technology to engineering technology problems that require the application of principles and applied procedures or methodologies.

Student Outcome-C
Conduct standard tests and measurements; to conduct, analyze, and interpret experiments; and to apply experimental results to improve processes.

Program Specific (CMPET and EET) Student Outcome-E
Utilize statistics/probability, transform methods, discrete mathematics, or applied differential equations in support of electrical, computer and networked systems.

Wireless Communications – Infra Red Transmitter

The student's goal was to create software to modulate an RS-232 signal and have this data sent out over an Infra-Red LED (similar to a TV remote control).

One possible circuit for this lab is shown in figure 13.

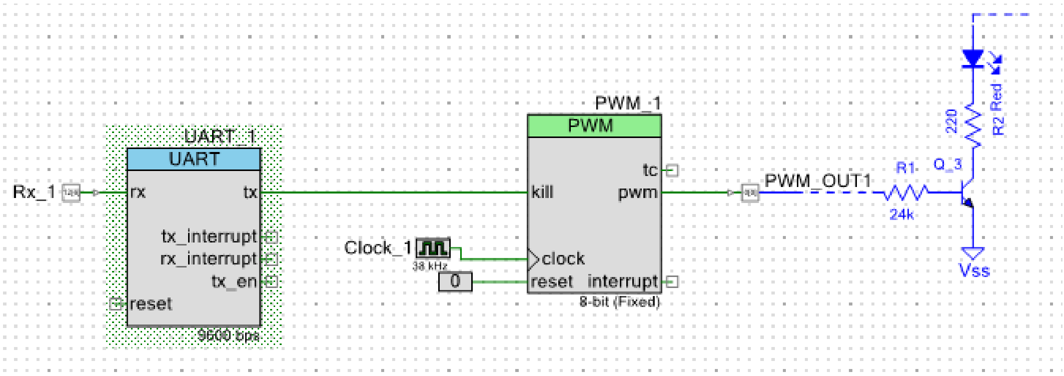


Figure 13 – Infra-Red Modulator Circuit

In this circuit, the internal UART is connected to the PWM circuit which is then wired to an external LED (transistor) circuit. The PWM outputs a 38 KHz signal which is enabled/disabled by the UART's output.

Once completed, this lab could be used for a variety of ABET ETAC outcomes. Two that are suggested include:

Student Outcome-F
Identify, analyze, and solve narrowly and broadly defined engineering technology problems.

Program Specific (CMPET and EET) Student Outcome-C
Analyze, design, and implement control systems, instrumentation systems, communications systems, or power systems, and hardware and software computer systems.

Summary

The PSoC 5LP is a good, general purpose microcontroller board and can be used as a basis for a variety of embedded projects. The 256 K Flash, 64 K Ram and programmable 32-bit ARM processor has enough performance to allow a variety of complex algorithms to be implemented. In addition, there is a programmable digital and analog logic array, thus minimizing the amount of external logic needed to complete a project. The interface is relatively easy to learn and the development kit includes a large number of examples and help information.

Typically, using a product that has all of these capabilities, it is expected that there would be a penalty in pricing or licensing, however the cost of a PSoC 5LP is $10 and the development tools (PSoC Creator) is a free download. (PSoC Creator comes with all of the help information, datasheets, examples and Windows drivers), making the kit very affordable to students.

Using only a basic tutorial, the students were able to quickly learn the development environment, create their schematics and debug the code. While this was an "experiment", it ended up proving very useful to the students. With just a short time developing on the PSoC 5LP board, two student groups chose to incorporate the processor into their senior projects, as they needed a low cost processor capable of doing analysis on audio input signals.

Assessment

As was earlier mentioned, this is the first time the students had ever used the PSoC 5LP, or any Systems-On-Chip device. They had never seen PSoC Creator before this point, however, they were familiar with Visual Studio as well as embedded programming and peripheral components.

Students were asked to first compare PSoC Creator to Microsoft's Visual Studio:

1a. How easy was it to pick up and understand PSoC Creator?
1b. Compare this to learning and using Visual Studio (easier/harder)?

In general all of the students found that PSoC creator was pretty easy to pick up and use. It had all of the debugging capabilities needed, and the error messages were straightforward to understand.

Students were asked to compare developing an application using the PSoC 5LP board/processor, compared to using a PIC microcontroller:

2a. How easy was it to write "useful" code/applications using the PSoC 5LP?
2b. Compare this to writing code for the PIC processor?

They found that writing code for the PSoC 5LP was fairly straight forward. The development tools included a variety of examples, plus detailed datasheets. All of the components had 'C' API calls that can be used to initialize and operate the circuits and in a short time were able to master their use.

3a. How easy was it to create/wire the schematic using PSoC Creator?
3b. Compare this to entering schematic data for pSpice (or EagleCad)?

The wiring/creation of the schematic was a great feature. It was very simple to just double click on the part and edit the settings by double clicking.

Thus the recommendation is to continue using the PSoC 5LP board in future class projects. It was relatively easy to learn, the information was useful to their senior capstone projects and the added value can be used for ABET ETAC Student Outcome-H (an understanding of the need for and an ability to engage in self-directed continuing professional development).

Bibliography

1. Web Site http://www.cypress.com/documentation/development-kitsboards/cy8ckit-059-psoc-5lp-prototyping-kit-onboard-programmer-and
2. Web Site http://www.abet.org/accreditation/accreditation-criteria/