Lab 0: Introduction to PSoC and PSoC Creator

Objective

Introduce the concepts and general practices for programming the Cypress PSoC using the PSoC Creator software.

Background

B1: PSoC

The Cypress PSoC (Programmable System-on-Chip) is a device that includes programmable analog and digital components that can interface with an on-board CPU microcontroller. In other words, it is capable of using many components commonly used in computer hardware design.

Many computer hardware applications have a fixed set of components and capabilities, based on their application. FPGAs (Field-Programmable Gate Arrays) are examples of digital computer hardware that can be reconfigured to the specifics of each application, but configuration requires learning a hardware descriptor language (HDL). HDLs use different methodologies than functional or object oriented languages, so learning how to use them is not trivial. Additionally, many FPGAs do not include a CPU for general purpose processing.

In comparison, the PSoC contains programmable digital and analog hardware, allowing it to be configured for many types of projects with differing requirements. Unlike an FPGA, configuring the hardware in PSoC Creator, the IDE for the PSoC, involves configuring and connecting together component blocks in a visual programming environment, thus eliminating the need to learn an HDL. Finally, the PSoC includes a CPU that can interface with the hardware components. PSoC Creator uses C to program the CPU and generates functions that form the interface between the software and hardware, so there is no need to learn how the CPU interfaces with the hardware.

B2: Overview of PSoC Creator

Programming the PSoC involves the following five steps:

1. Create a project in PSoC Creator
2. Design the hardware by choosing, configuring, and connecting components in the project schematic
3. Configure design-wide resources (such as connections between physical ports and ports in the schematic)
4. Program firmware on the CPU as necessary to interact with the hardware
5. Compile the project and program the PSoC

All of these steps are necessary for every PSoC project. This lab will describe how to perform these steps in PSoC Creator. Future labs will not go into as much detail on how to use PSoC Creator, so refer back to this lab in the future, if necessary.

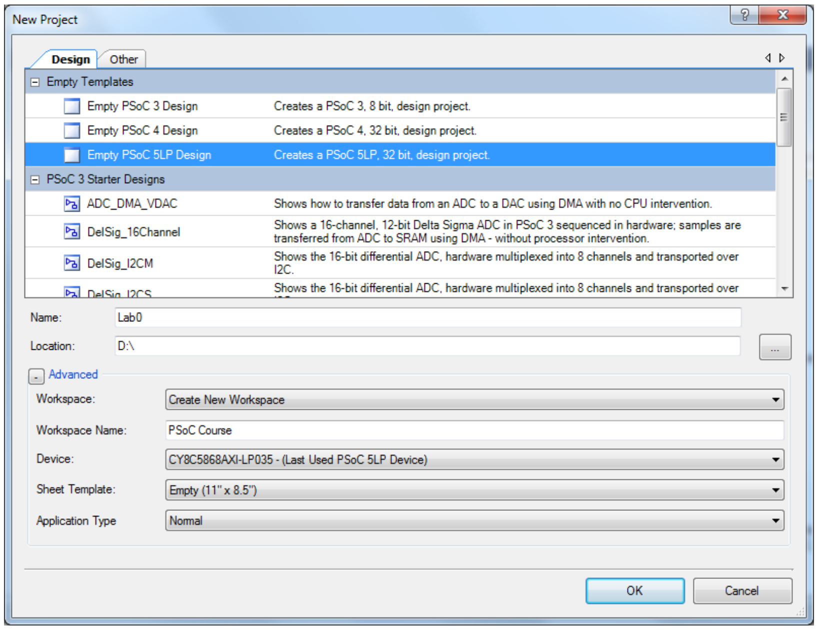
This lab and all following labs use a complete installation of PSoC Creator 3.0. Older versions may not have the same hardware component capabilities. Newer versions may look different, but should be compatible. Download PSoC Creator for free here: http://www.cypress.com/psoccreator/

Procedure

Part 1: Creating a Project

1. Start PSoC Creator.
2. Click on Create Project or File>New>Project. Refer to Figure 1 for the following steps.
3. The development kits use the PSoC 5LP chip, so choose “Empty PSoC 5LP Design”
4. Name the project “Lab0” and set a location to save it to (ask instructor for the best location).
5. Expand the Advanced section. Name the new workspace something like “PSoC Course”, ensure that the device selected is CY8C5868AXI-LP035 (should be the default), which is the PSoC model used in the development kit, and ensure that “Application Type” is set to “normal”. Click OK.

This workspace can contain all the projects created for the course. A new workspace could be created for every project, but the IDE can only open projects in the currently open workspace, so that would make it hard to refer back to previous labs. To add future labs to the same workspace, open the workspace, open the new project dialog, and choose “Add to Current Workspace” next to Workspace in the Advanced area.



Empty Templates	
Empty PSoC 3 Design	Creates a PSoC 3, 8 bit, design project.
Empty PSoC 4 Design	Creates a PSoC 4, 32 bit, design project.
Empty PSoC 5LP Design	Creates a PSoC 5LP, 32 bit, design project.
PSoC 3 Starter Designs	
ADC_DMA_VDAC	Shows how to transfer data from an ADC to a DAC using DMA with no CPU intervention.
DelSig_16Channel	Shows a 16-channel, 12-bit Delta Sigma ADC in PSoC 3 sequenced in hardware; samples are transferred from ADC to SRAM using DMA - without processor intervention.
DelSig_I2CM	Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over I2C.
DelSig_I2CS	Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over

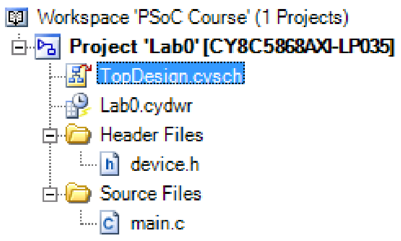
Name: Lab0
Location: D:\
Advanced
Workspace: Create New Workspace
Workspace Name: PSoC Course
Device: CY8C5868AXI-LP035 - (Last Used PSoC 5LP Device)
Sheet Template: Empty (11" x 8.5")
Application Type: Normal

Figure 1: Creating a Project

Part 2: Designing the Schematic

The first part of every project is to design the hardware portion by drawing a schematic of all the needed components, configuring them, and connecting them. The hardware for this lab will turn on an LED while a pushbutton is depressed. Additionally, the LCD panel will be used by the software to display how many times the button was pressed.

1. If the tab isn't open already, open TopDesign.cysch by double clicking it in the Workspace Explorer on the left side of the main window.


Workspace 'PSoC Course' (1 Projects)
Project 'Lab0' [CY8C5868AXI-LP035]
TopDesign.cysch
Lab0.cydwr
Header Files
device.h
Source Files
main.c

2. Take note of the main tools in the schematic view tab in Figure 2.

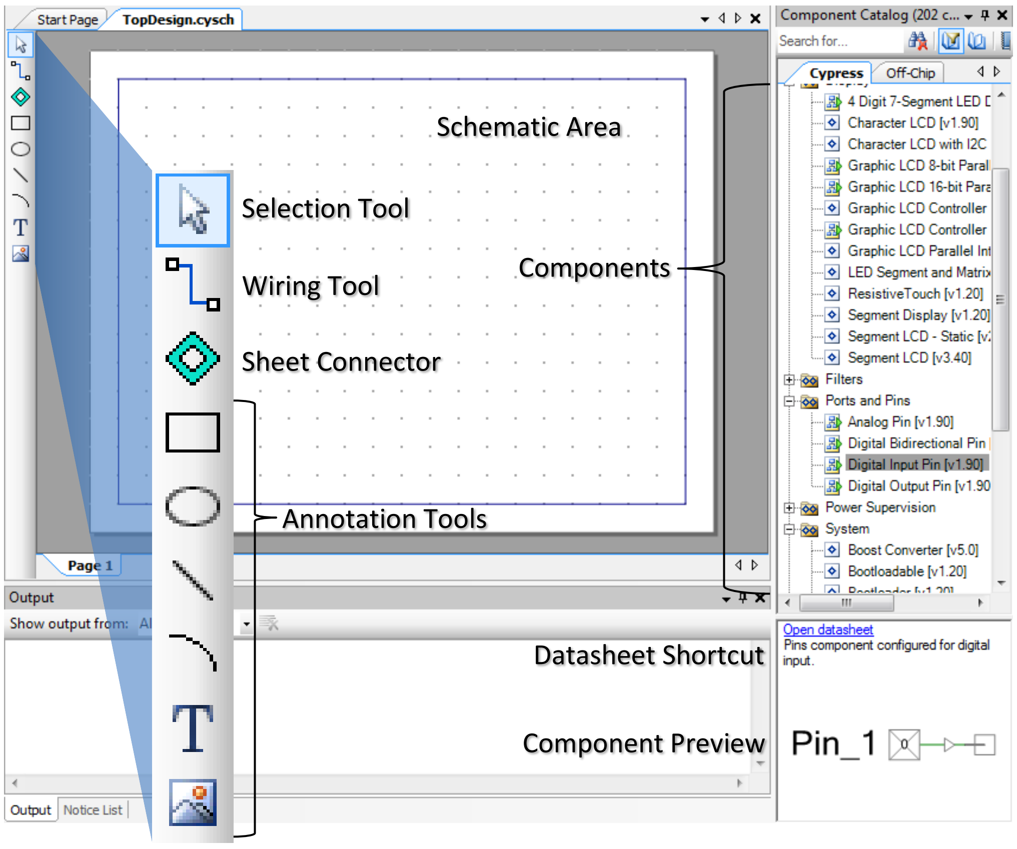


Figure 2: Schematic Editor

Schematic Area – the place to draw the hardware schematic

Components – A list of hardware components that can be placed on the schematic. There are two types of components:

- Cypress – components that the PSoC hardware will be configured to use. Future labs will generally revolve around introducing these.
- Off-Chip – external components, such as LEDs, that the PSoC interfaces with. These essentially annotate the schematic to show how the PSoC is interfacing with other hardware. These will be explained more in the next lab.

Component Preview – shows what the selected component looks like

Datasheet Shortcut – brief description of the currently selected component and a link to the datasheet PDF. The datasheet contains all the documentation needed for configuring and using the component.

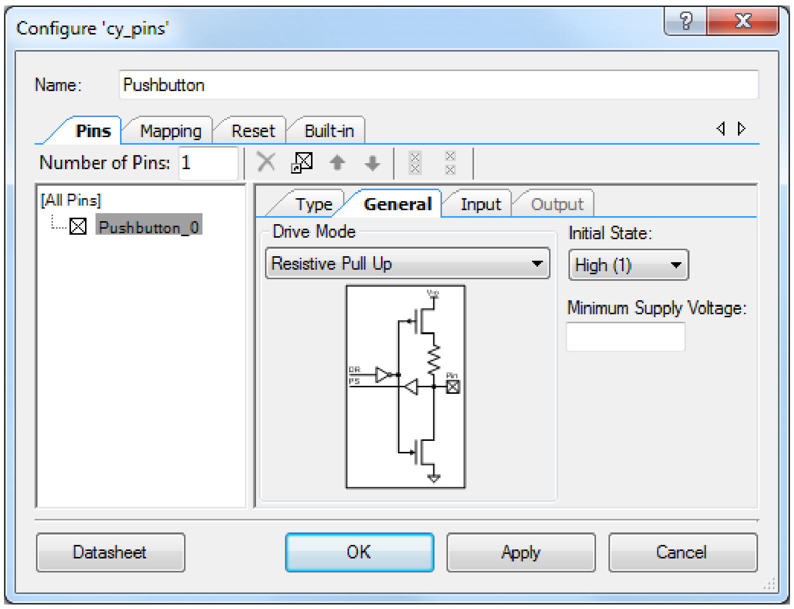
Selection Tool – standard manipulation tool

Wiring Tool – for wiring hardware components together (shortcut: press W)

Sheet Connector – if the schematic is too big, a new page can be created in the same project. This tool allows for connections between two schematic pages.

Annotation Tools – allows for drawing shapes, adding label text, and placing images to improve the clarity of a schematic.

3. From the Cypress components, click and drag a Ports and Pins>Digital Input Pin onto the schematic area.
4. Double click on the just added input pin to bring up the component configuration window.
5. Rename the pin to “Pushbutton”.
6. In the Pins>Type tab, uncheck the HW Connection box under Digital Input.
7. Because of how the pushbutton on the development kit is connected, go to the Pins>General tab and change the Drive Mode to Resistive Pull Up.
8. Press OK


Configure 'cy_pins'

Name: Pushbutton

Number of Pins: 1

[All Pins]
Pushbutton_0

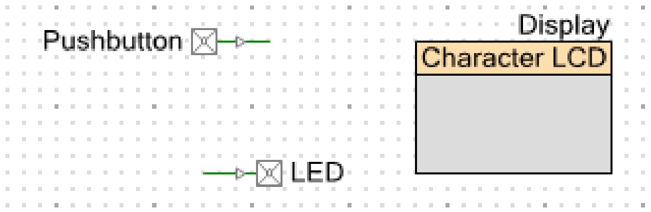
Drive Mode: Resistive Pull Up

Initial State: High (1)

Minimum Supply Voltage:

Figure 3: Configuring the Pushbutton Pin

9. Place a Ports and Pins>Digital Output Pin onto the schematic, uncheck HW Connection, and rename it to “LED” with the configuration window. Do not change the drive mode.
10. Place a Display>Character LCD onto the schematic. Rename it to “Display” with the configuration window.
11. The final schematic should look similar to what’s below. The components in the schematic should look exactly the same as what’s below, but not necessarily arranged in the same way.

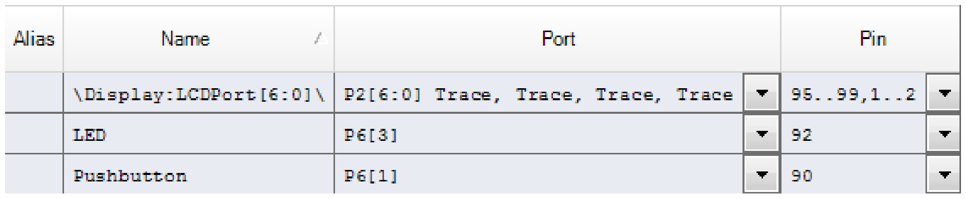


12. Save the schematic.

Part 3: Configure Design-Wide Resources

The next part of the lab is to configure Design-Wide Resources (DWR) file. The DWR allows the configuration of global PSoC resources, such as pin mappings, clocks, interrupts, and DMA, most of which will be covered in future labs. The DWR configures the parts of the PSoC that connect the hardware schematic and CPU to the actual hardware. The important part for this lab is to connect the Pushbutton, LED, and (hidden) Display I/O pins on the schematic to the physical pins on the chip.

1. Open the DWR file named Lab0.cydwr from the Workspace Explorer.
2. The Pins tab displays a diagram of the PSoC and all of its physical pins. Additionally, there is a list of the I/O pins from the schematic on the right.
3. Look at the PSoC development board (or look at Figure 5 below) and take note of the markings next to the pushbutton marked SW2, the LED marked LED4, and LCD port. All of these components are beneath the prototyping area (the white square with holes). The markings that start with a P indicate the physical port and pin on the PSoC that the components are connected to. The number after the P indicates the port number and the number after the underscore indicates the pin number in the port.
4. Using the Port column on the list of pins in the DWR view, configure Display to use ports P2[6:0], LED to use P6[3], and Pushbutton to use port P6[1]. These ports match the markings on the development kit (6:0 indicates pins 0-6 of that port). See Figure 4 for how the list should look after configuring it.



Alias	Name	Port	Pin
	\Display:LCDPort[6:0]\	P2[6:0] Trace, Trace, Trace, Trace	95..99,1..2
	LED	P6[3]	92
	Pushbutton	P6[1]	90

Figure 4: Pin Configuration List

5. Save the DWR file.

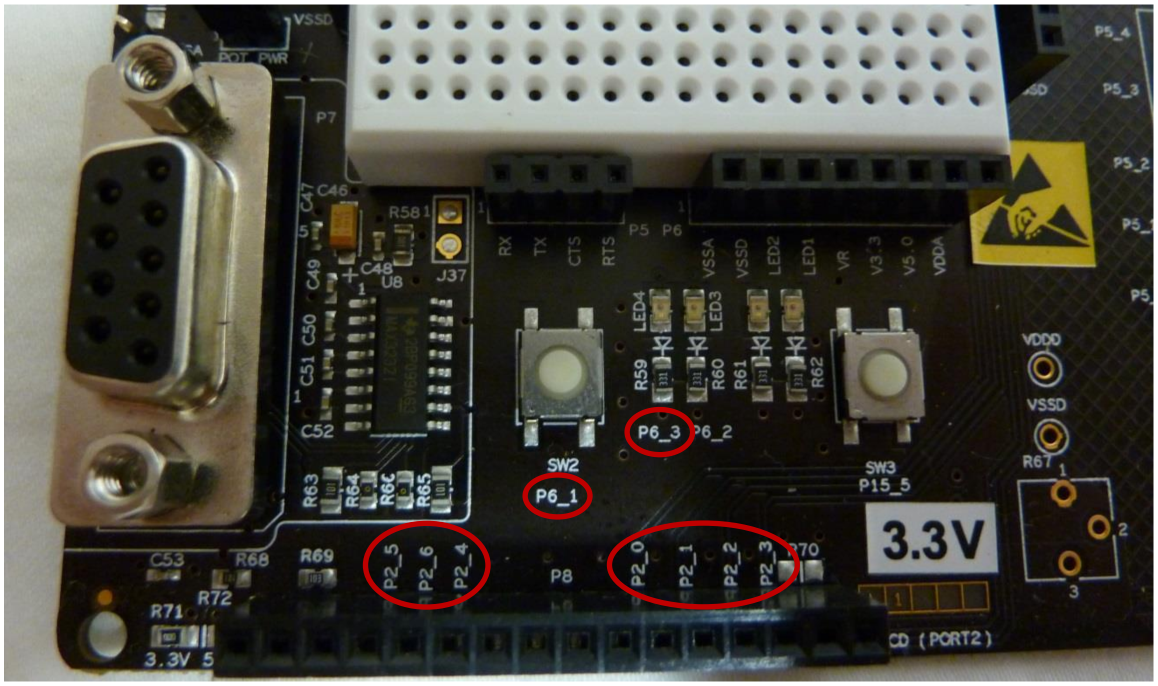


Figure 5: LED, pushbutton, and LCD port area of the PSoC 5LP development kit

Part 4: Programming the Firmware

The PSoC includes a CPU that can interface with the hardware configured in the schematic. The program loaded on it is called the firmware. In this lab, the firmware will continuously poll the pushbutton input pin, count how many times the pushbutton is pressed, and print the count to the LCD display. All firmware code is programmed in C.

1. Before writing any code, select Build>Generate Application from the main window menu. There should also be a button for this below the save button. This will produce all of the C files required for the firmware to interface with the hardware configuration in the schematic. Always perform this step after making additional changes to the schematic.

The names that are given to the components in the schematic become the names of the firmware interfaces to the components. Additionally, the names are used in the function names of the interface. For example, in the character LCD datasheet, all of the function names start with LCD_Char, such as LCD_Char_PrintString. The name of the actual component replaces LCD_Char, so in this lab all of the LCD functions start with Display, such as Display_PrintString.

Avoid changing the names of the components in the schematic after code is written that uses them, otherwise function calls to the renamed components will need to be renamed.

Most of the generated files should not be modified, as doing so could break their functionality. Additionally, generating the application again will replace any modifications. There are some exceptions, but for now the only file that can be safely modified is main.c.

To find out how to use the firmware interface for a component, right click the component in the schematic and click Open Datasheet. The datasheet also includes details on how to configure the components in the schematic and how it works.

2. Open main.c, which should include several comments and a main function with an infinite loop.
3. See the code snippet on the next page for an outline of the code to write.
4. Create a count variable with the type uint16 and a uint8 acting as a boolean to indicate when the button was pressed.
   Note: the bool type is not defined in the compiler, so a uint8 with 0 as false and 1 as true is used instead.
5. Initialize the LCD display before the infinite loop by:
   - Calling Display_Start() to initialize the LCD
   - Set the print position to row 0 column 0 using Display_Position(0, 0)
   - Print the string "Count" using Display_PrintString("Count")
   - Set the print position to row 1 column 0
   - Print 0 using Display_PrintNumber(0)
6. Initialize the LED's state using LED_Write(0).
7. In the infinite loop, poll the pushbutton using Pushbutton_Read(). If the button is pressed (remember, 0 is pressed, 1 is not), increment the count and update the display to show the new count. Use the boolean to ensure that every press is only counted once.
8. Additionally, while the pushbutton is depressed, turn on the LED using LED_Write(1). When the pushbutton is released, turn the LED off using LED_Write(0).
9. Build the project using Build>Build Lab0. The first button in the second row of the toolbar should also be the build button.

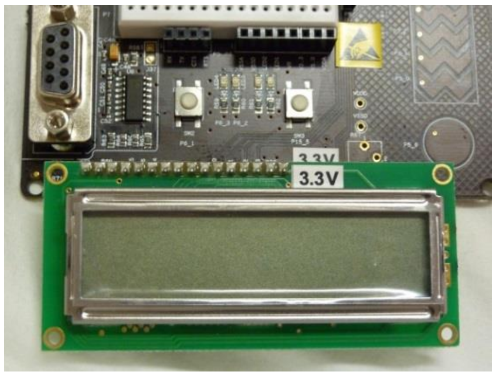
```
int main()
{
    //The number of times the pushbutton has been pressed
    uint16 count = 0;

    //A "boolean" to keep track if the button was read as pressed
    //in the previous loop iteration
    uint8 pressed = 0;


    //Initialize the display
    Display_Start();
    Display_Position(0, 0);
    Display_PrintString("Count");
    Display_Position(1, 0);
    Display_PrintNumber(count);

    //Initialize the LED to off
    LED_Write(0);

    //Infinitely loop
    for(;;)
    {
        //Insert the code from steps 7-8 here.
        //Use LED_Write(1) to turn on the LED when the button is pressed
        //and use LED_Write(0) turn the LED off otherwise.
        //The value of the pushbutton is read using Pushbutton_Read().
        //Pushbutton_Read() is 1 when not pressed, 0 when pressed.

        //Place the two lines below in your code to print the count.
        //Only print the count when it changes, otherwise the LCD may flicker
        Display_Position(1, 0);
        Display_PrintNumber(count);
    }
}
```

Part 5: Running the Project

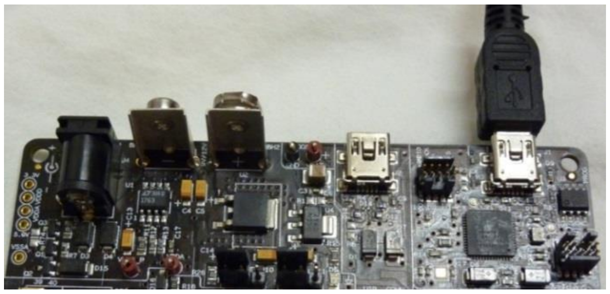
At this point, the project is ready to be loaded and run on the PSoC development kit.

Read the handling and safety guide for the PSoC before continuing.

1. Gently remove the PSoC development kit, LCD display, and USB cable from the development kit box.
2. Gently plug in the LCD panel into the bottom port of the development kit as shown.



3. Plug the USB cable into the programming port at the top right of the development kit.



4. Plug the USB cable into the computer.
5. Program the development kit using Debug>Program. The first time you program the board, you may have to wait a few seconds for Windows to finish automatically installing drivers. Also, if this isn't the first time programming the board since plugging it in, it could fail to program with an unknown error. Just disconnect and reconnect the USB cable if this happens and try again.
6. Test the program to make sure it works:
   - Pressing the SW2 should turn on LED4, and releasing the button should turn it off.
   - Every time the button is depressed, the counter should increment by one. Holding down the button should not cause any increments.
7. Note that sometimes a button press may cause the counter to increment more than once and a release may occasionally increment the counter as well. This occasional miscount is due to mechanical bouncing of the switch as it transitions between states. Lab 3 will solve this issue.