CE4920 PSOC TUTORIAL

INTRODUCTION

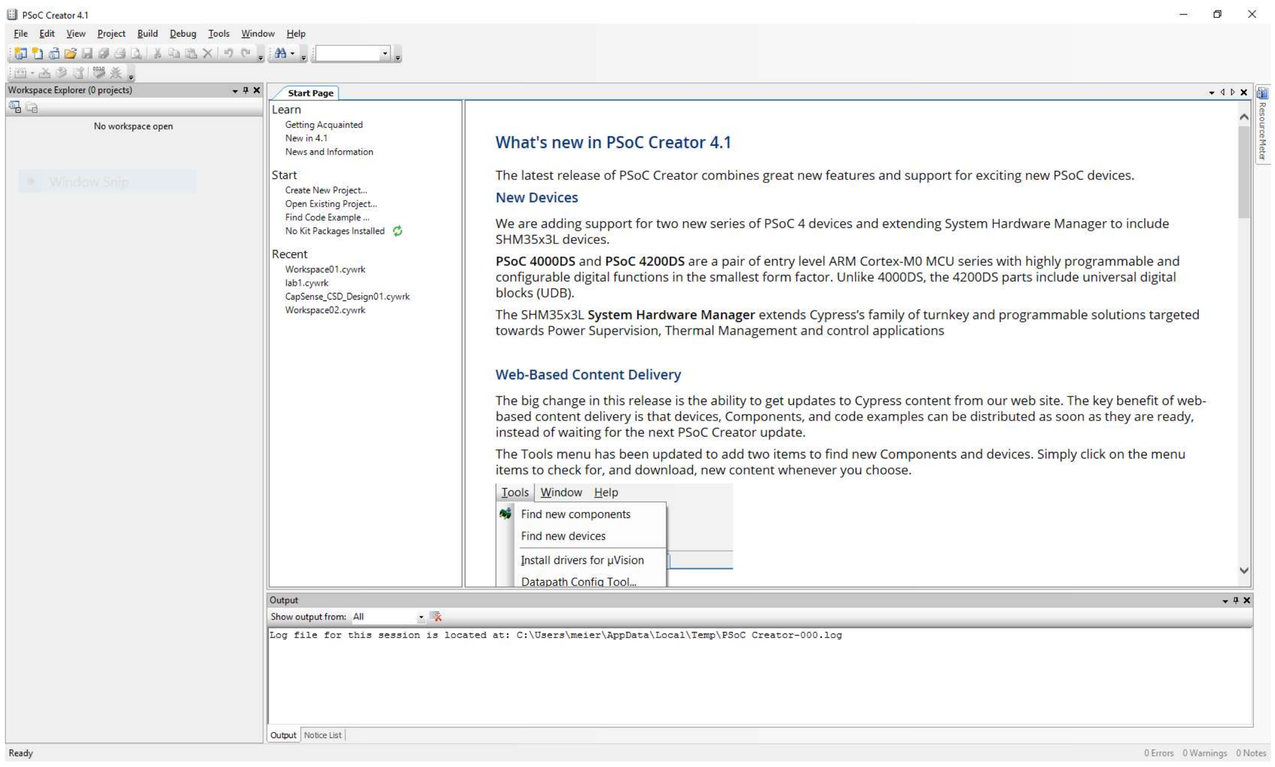
The MSOE embedded systems sequence rigorously prepares students for design engineering jobs in industry. Over the past three years, students wrote firmware in both Assembly language as well as the high-level C programming language to control memory-mapped peripheral devices that sensed and controlled the system environment. Students also designed, simulated, and built combinational logic circuits, sequential logic circuits, as well as standard interfacing circuits such as amplifiers, filters, ADCs, DACs, resistive pull-ups and transistor drivers. CE4920 is the final course in the MSOE embedded systems sequence. This project course requires students to integrate their analog and digital circuit skills to complete the design of a treadmill used in physical therapy or recreational exercise. The project requires the use of a Cypress Semiconductor Programmable System-on-a-Chip (PSOC) technology. PSOC devices provide a reconfigurable mixed-signal fabric and a hard-core processor that together form a comprehensive single-chip solution.

Use this tutorial to learn about the design flow used in the PSOC Creator computer-aided-design software. The tutorial documents PSOC Creator 4.1.

CE4920 PSOC TUTORIAL

TUTORIAL

1. **Start** the PSOC Creator 4.1 software.
    a. A start page shows push-content from Cypress Semiconductor.
    b. Cypress provides tutorials under the Getting Started and Examples headings.



What's new in PSoC Creator 4.1

The latest release of PSoC Creator combines great new features and support for exciting new PSoC devices.

New Devices

We are adding support for two new series of PSoC 4 devices and extending System Hardware Manager to include SHM35x3L devices.

PSoC 4000DS and **PSoC 4200DS** are a pair of entry level ARM Cortex-M0 MCU series with highly programmable and configurable digital functions in the smallest form factor. Unlike 4000DS, the 4200DS parts include universal digital blocks (UDB).

The SHM35x3L **System Hardware Manager** extends Cypress's family of turnkey and programmable solutions targeted towards Power Supervision, Thermal Management and control applications

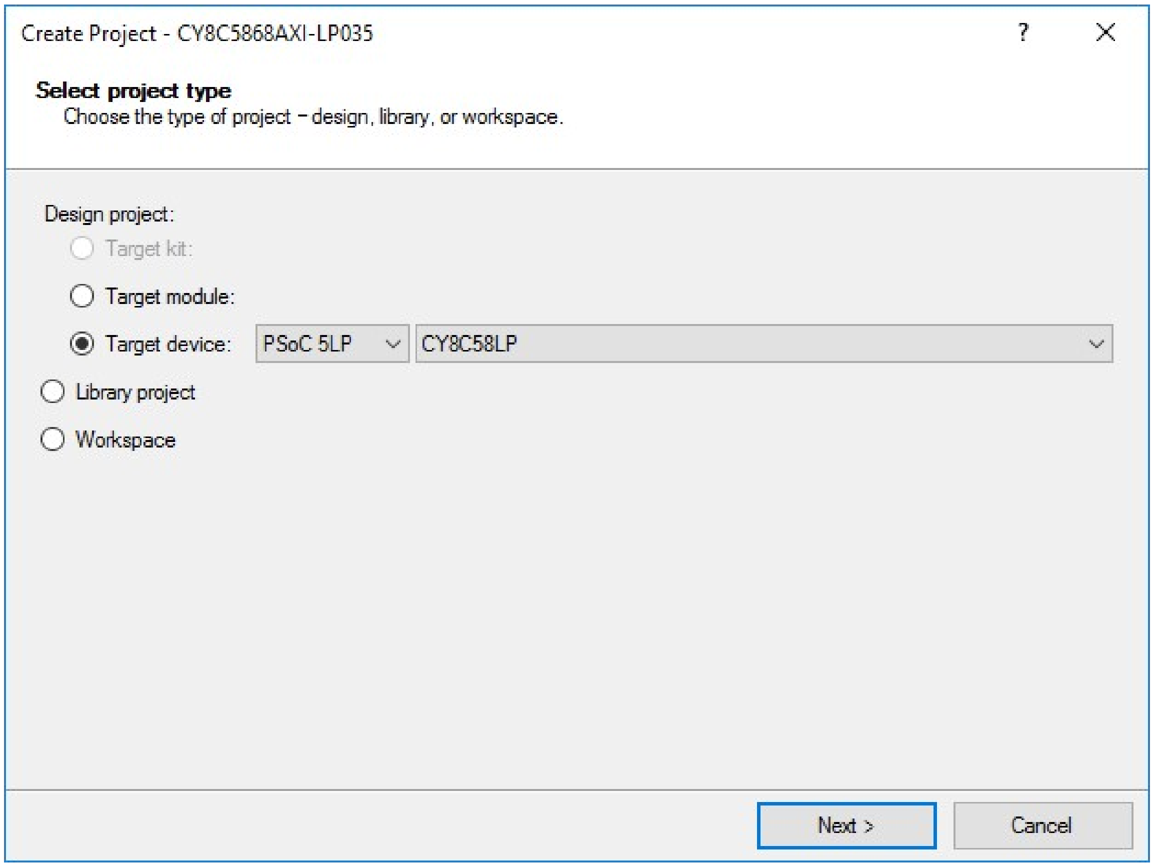
Web-Based Content Delivery

The big change in this release is the ability to get updates to Cypress content from our web site. The key benefit of web-based content delivery is that devices, Components, and code examples can be distributed as soon as they are ready, instead of waiting for the next PSoC Creator update.

The Tools menu has been updated to add two items to find new Components and devices. Simply click on the menu items to check for, and download, new content whenever you choose.

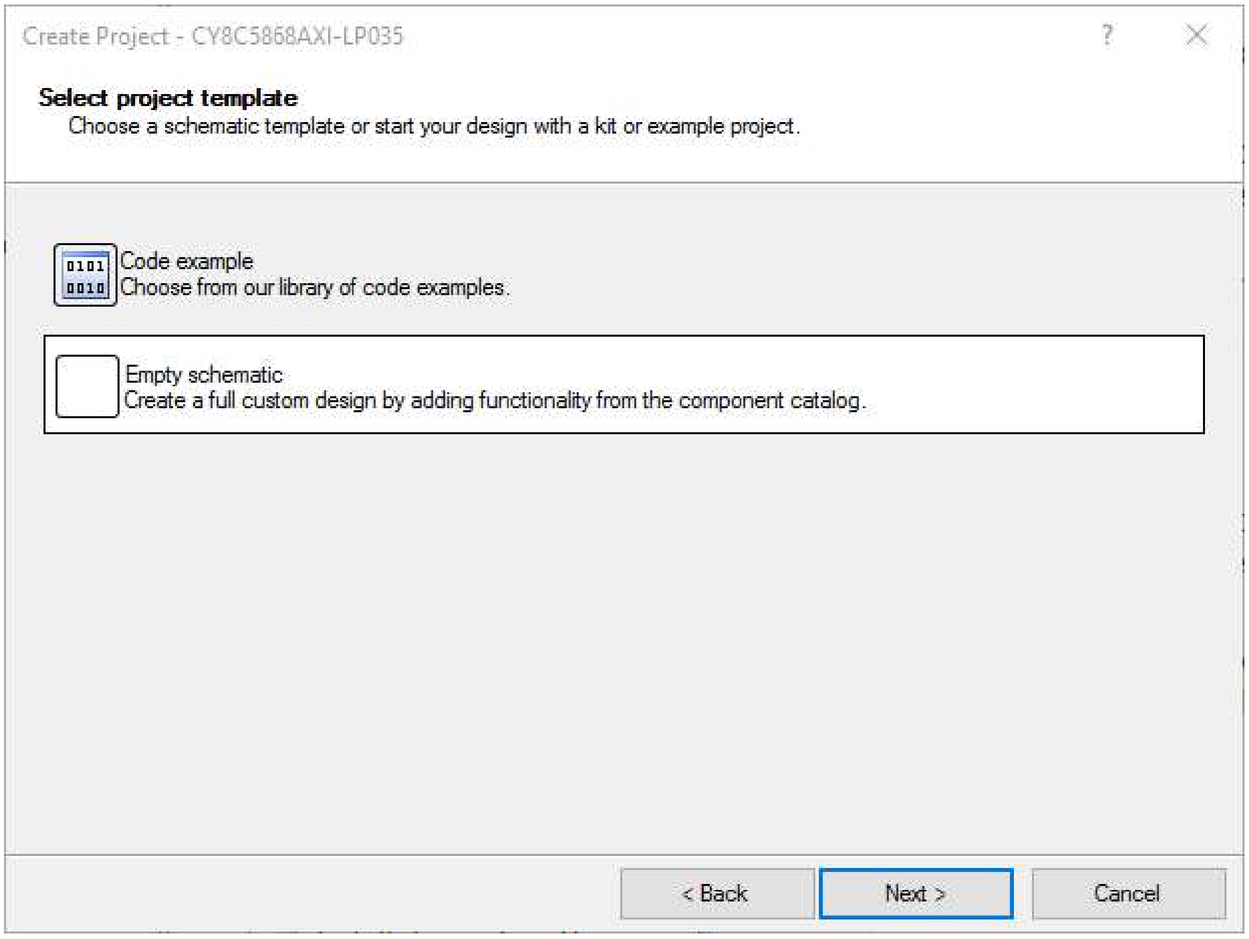
```
Log file for this session is located at: C:\Users\meier\AppData\Local\Temp\PSoC Creator-000.log
```

2. **Make** a new PSOC 5LP project in an appropriate folder.


Create Project - CY8C5868AXI-LP035

Select project type
Choose the type of project – design, library, or workspace.

Design project:
- ( ) Target kit:
- ( ) Target module:
- (●) Target device: PSoC 5LP | CY8C58LP
- ( ) Library project
- ( ) Workspace

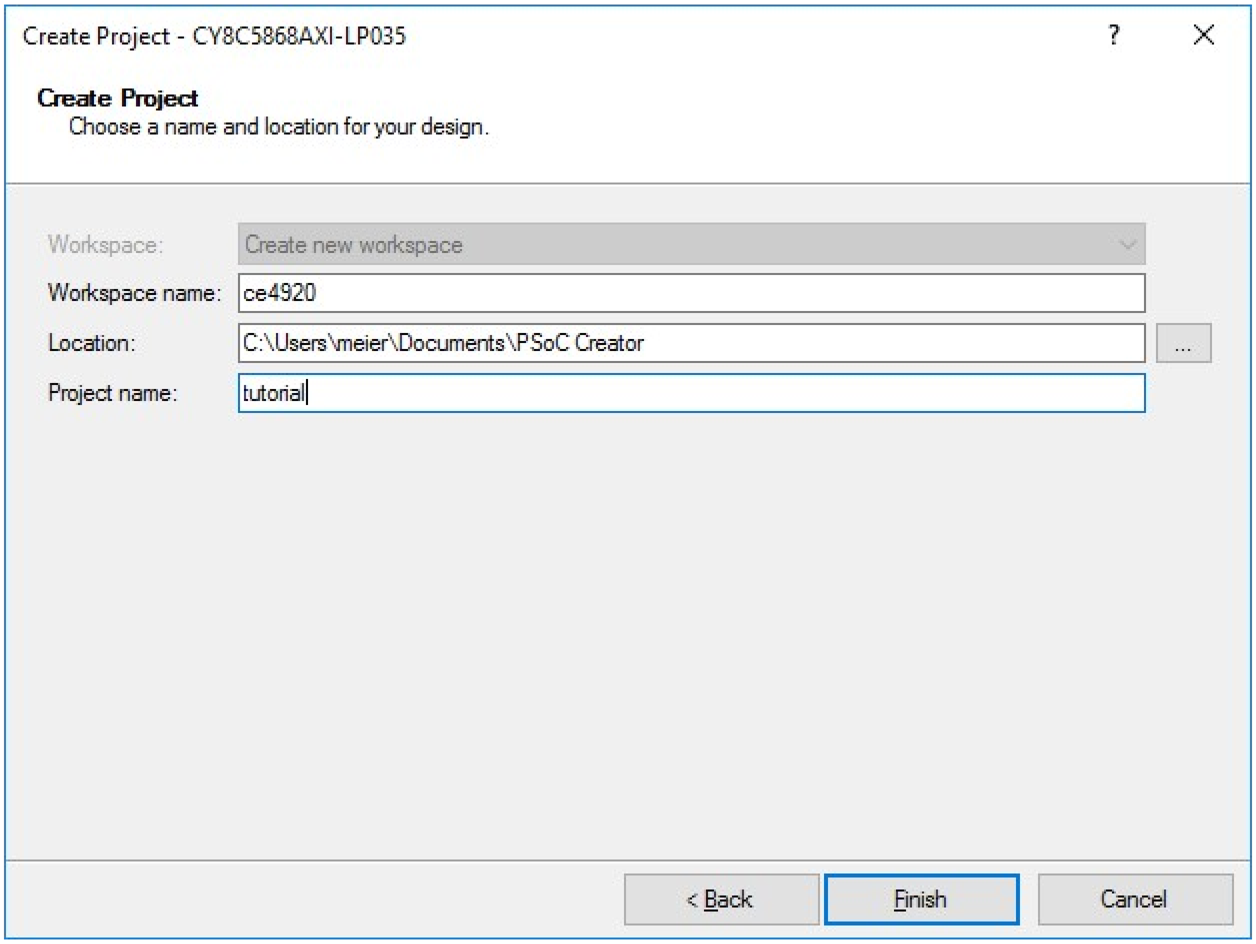


Select project template

Choose a schematic template or start your design with a kit or example project.

Code example
Choose from our library of code examples.

Empty schematic
Create a full custom design by adding functionality from the component catalog.



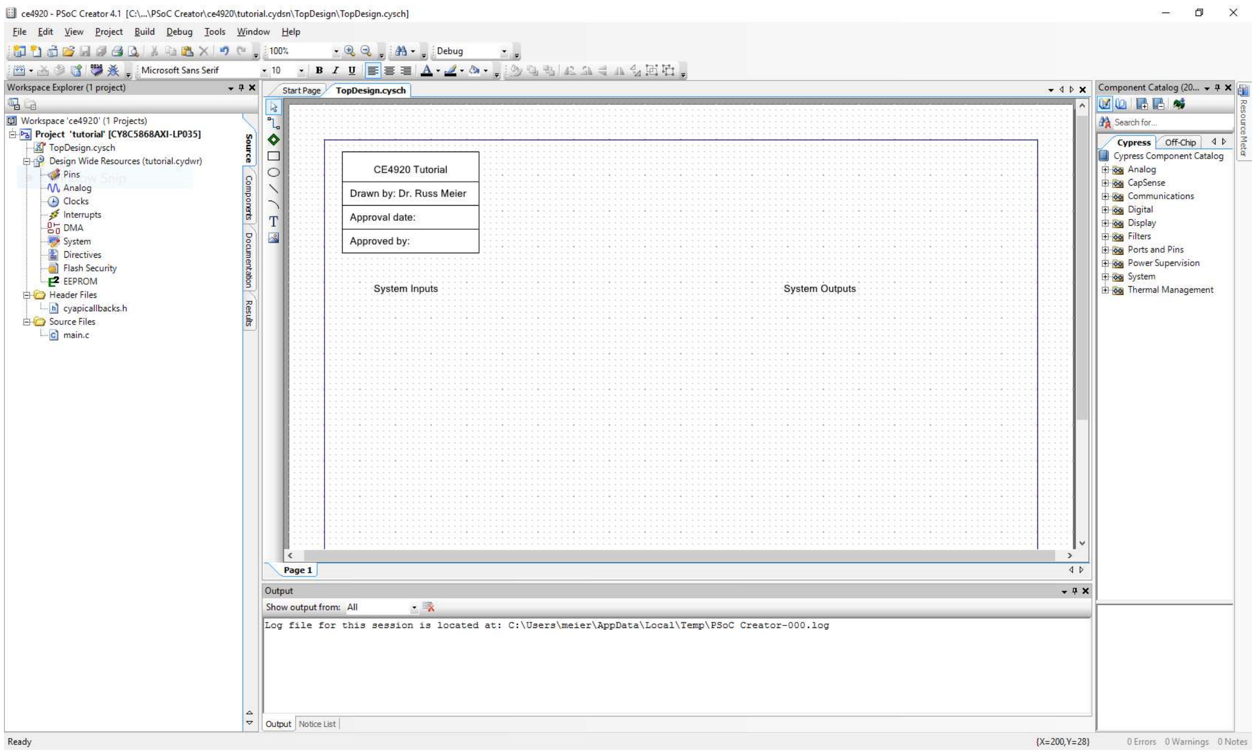
Create Project - CY8C5868AXI-LP035

Create Project
Choose a name and location for your design.

Workspace:	Create new workspace
Workspace name:	ce4920
Location:	C:\Users\meier\Documents\PSoC Creator
Project name:	tutorial

Click the **Finish** button.

3. **Zoom** the top-level schematic diagram to 100% and add a title block.
    a. **Use** the rectangle, line, and text tools located along the left-hand side of the schematic to draw your title block shape and add text labels.
    b. **Note** that this example shows the title block in the upper left hand corner.
    c. **Note** that both system input and system output text labels also have been added.



CE4920 Tutorial
Drawn by: Dr. Russ Meier
Approval date:
Approved by:

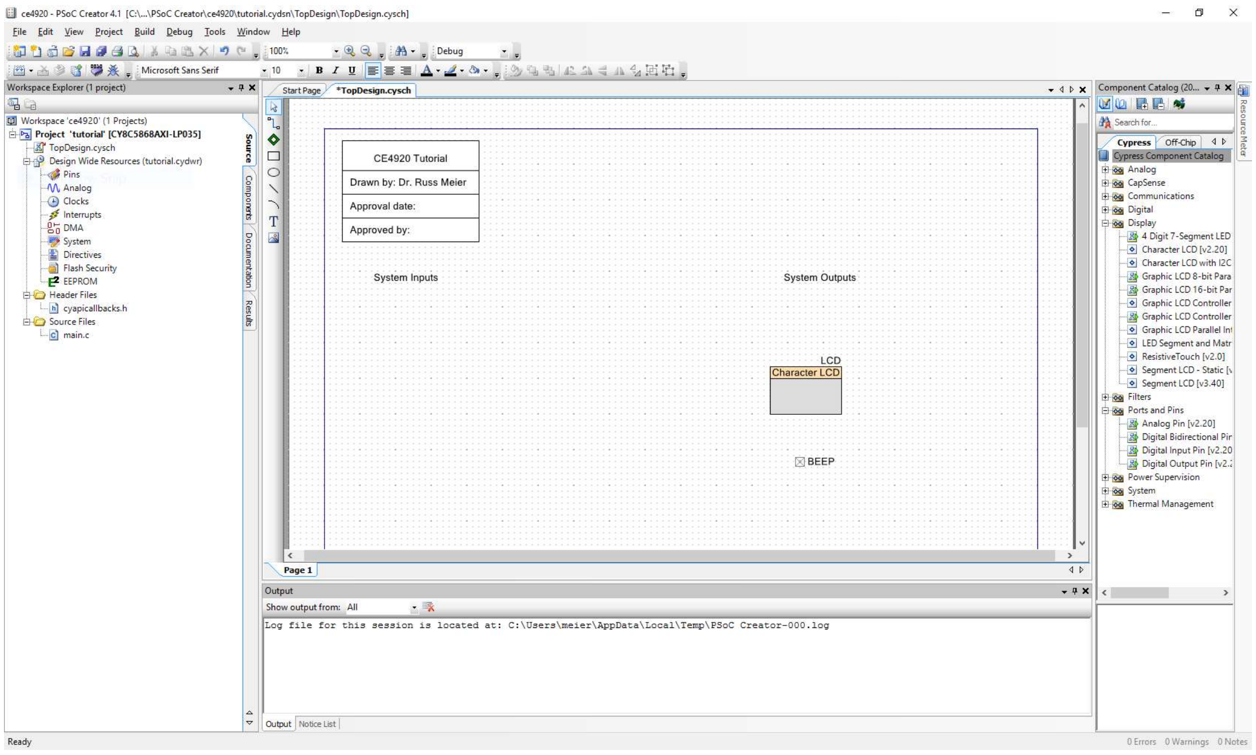
System Inputs

System Outputs

Show output from: All

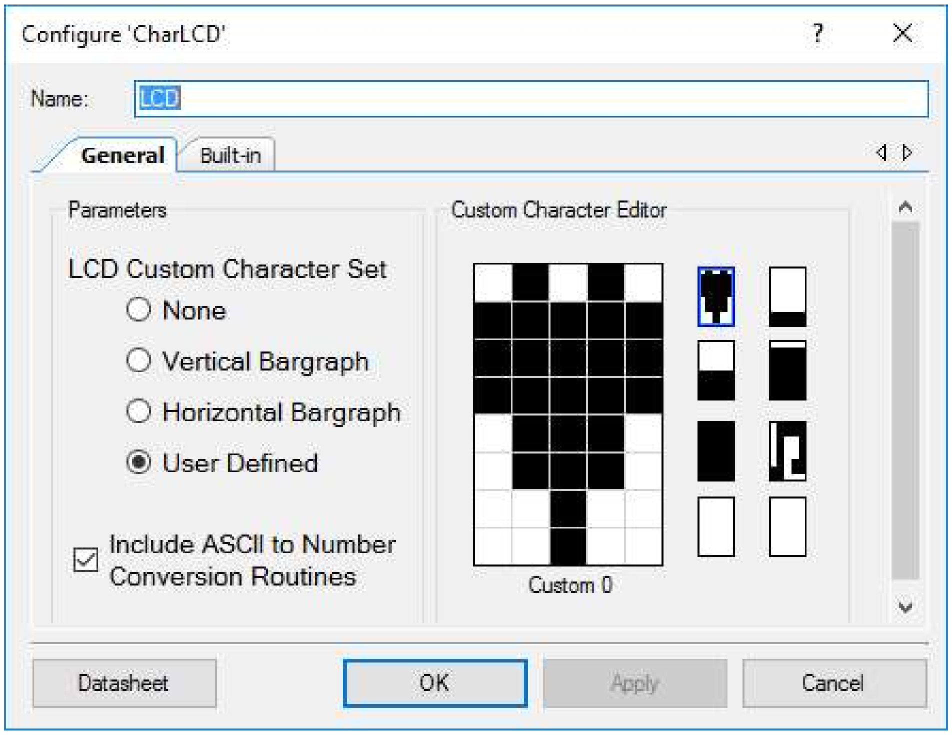
```
Log file for this session is located at: C:\Users\meier\AppData\Local\Temp\PSoC Creator-000.log
```

4. **Add** a character LCD and a digital output pin to drive a piezo-speaker.
    a. **Expand** the Display group in the Cypress Component Catalog panel.
    b. **Expand** the Ports and Pins group in the Cypress Component Catalog panel.
    c. **Drag** the character LCD panel and a digital output pin onto the schematic.
    d. **Double-click** the component to configure and rename.
    e. **Note** that the name of the component becomes the object name in software.
    f. **Note** that LCD panels allow custom character sets. This example shows one.
    g. **Double-click** the digital output pin and enable firmware control by turning off hardware connection.



```
Log file for this session is located at: C:\Users\meier\AppData\Local\Temp\PSoC Creator-000.log
```

See additional images for step 4 on the next page



Configure 'CharLCD'

Name: LCD

General | Built-in

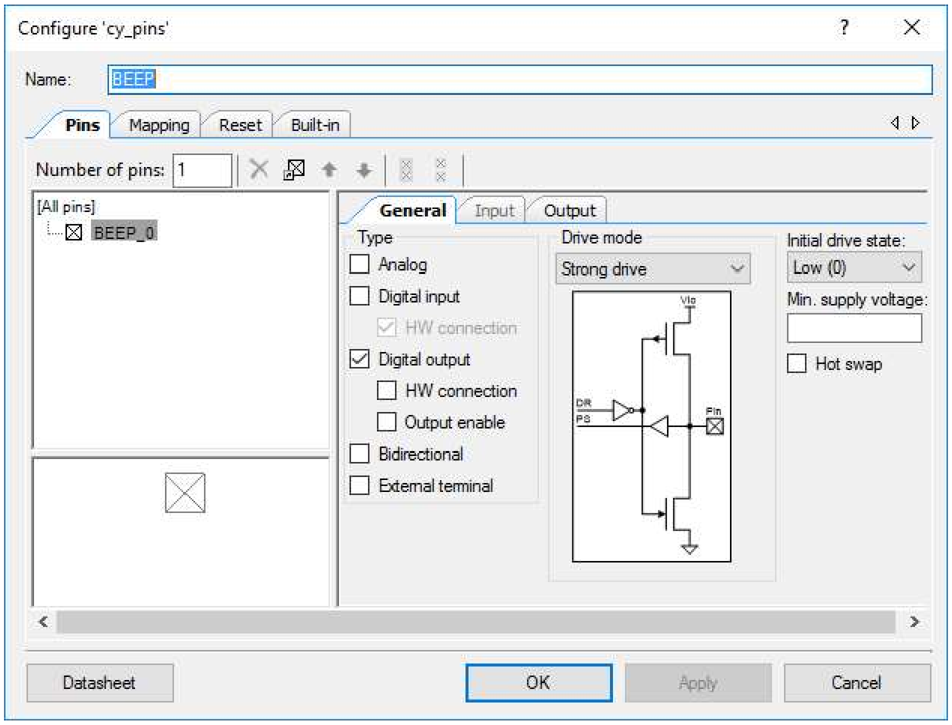
Parameters

LCD Custom Character Set
- None
- Vertical Bargraph
- Horizontal Bargraph
- [✓ User Defined]

[✓] Include ASCII to Number Conversion Routines

Custom Character Editor

Custom 0



Configure 'cy_pins'

Name: BEEP

Pins | Mapping | Reset | Built-in

Number of pins: 1

[All pins]
BEEP_0

General | Input | Output

Type
- [ ] Analog
- [ ] Digital input
  - [✓] HW connection
- [✓] Digital output
  - [ ] HW connection
  - [ ] Output enable
- [ ] Bidirectional
- [ ] External terminal

Drive mode: Strong drive

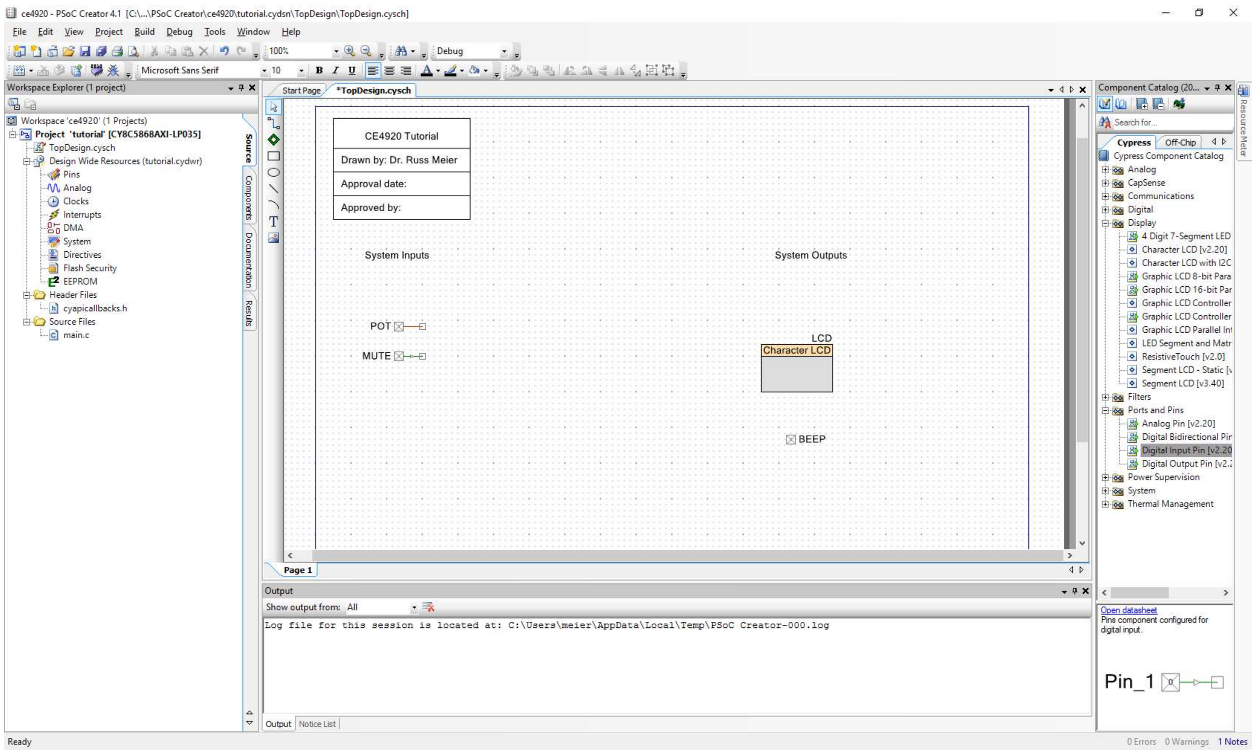
Initial drive state: Low (0)

Min. supply voltage:

[ ] Hot swap

CE4920 PSOC TUTORIAL

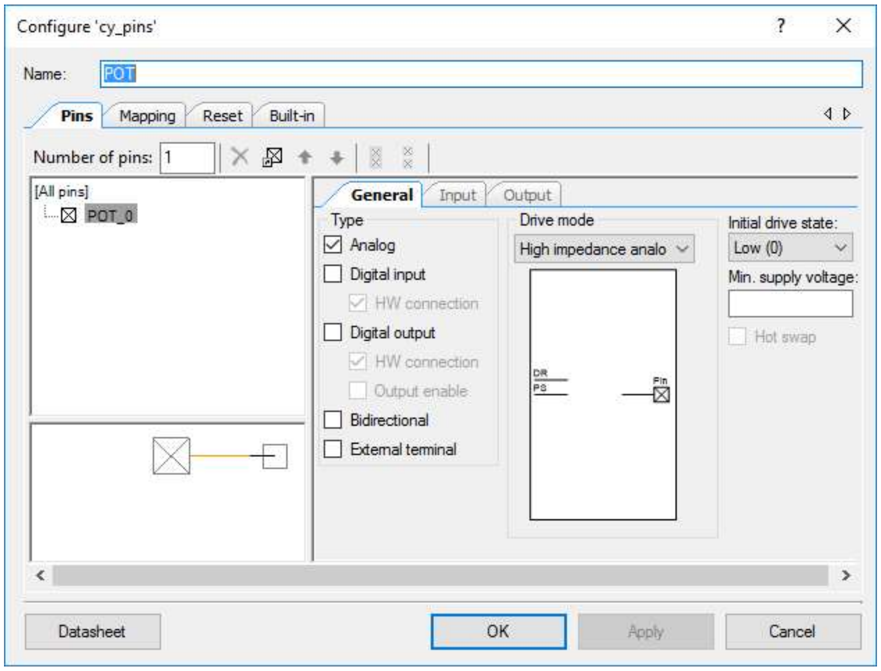
5. **Add** an analog input pin and a digital input pin.
    a. Analog-to-digital conversion of a potentiometer will vary tone frequency.
    b. **Rename** the analog input POT.
    c. A pushbutton switch will mute and unmute tones.
    d. **Configure** the digital input with resistive pull-up on switch release.
    e. **Note** that "hardware connection" is checked for this input pin as the values are driven from hardware and not from firmware.



```
Log file for this session is located at: C:\Users\meier\AppData\Local\Temp\PSoC Creator-000.log
```

See additional images for step 5 on the next page

CE4920 PSOC TUTORIAL



Configure 'cy_pins'

Name: POT

Pins | Mapping | Reset | Built-in

Number of pins: 1

[All pins]
POT_0

General | Input | Output

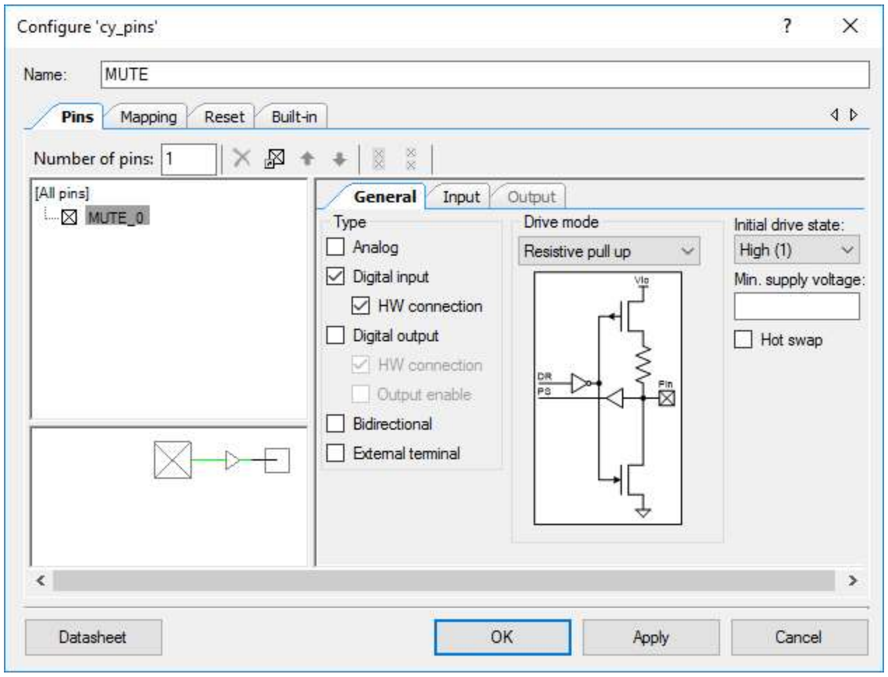
Type
- [x] Analog
- [ ] Digital input
  - [x] HW connection
- [ ] Digital output
  - [x] HW connection
  - [ ] Output enable
- [ ] Bidirectional
- [ ] External terminal

Drive mode: High impedance analo

Initial drive state: Low (0)

Min. supply voltage:

- [ ] Hot swap



Configure 'cy_pins'

Name: MUTE

Pins | Mapping | Reset | Built-in

Number of pins: 1

[All pins]
MUTE_0

General | Input | Output

Type
- [ ] Analog
- [x] Digital input
  - [x] HW connection
- [ ] Digital output
  - [x] HW connection
  - [ ] Output enable
- [ ] Bidirectional
- [ ] External terminal

Drive mode: Resistive pull up

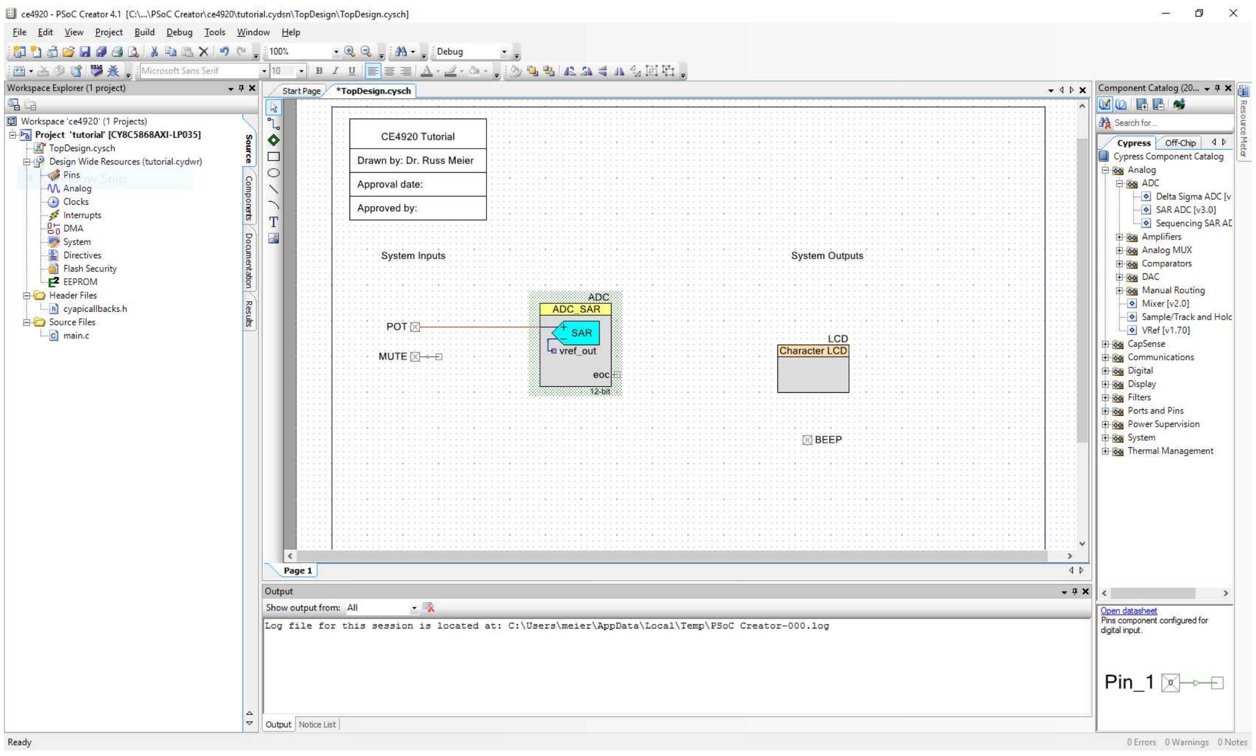
Initial drive state: High (1)

Min. supply voltage:

- [ ] Hot swap

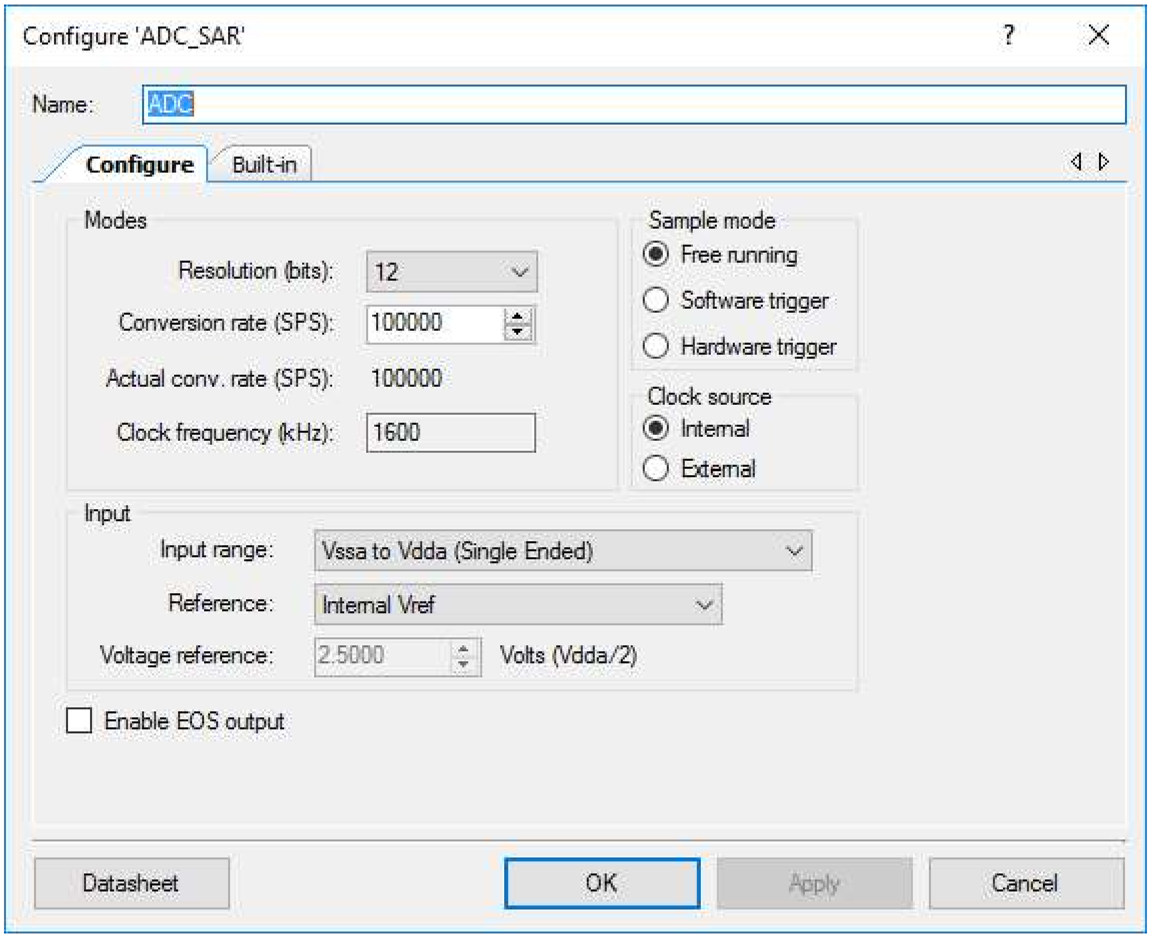
CE4920 PSOC TUTORIAL

6. **Add** an SAR ADC and configure it as shown.
    a. **Consult** the component data sheet for information about settings.
    b. **Access** the data sheet by double-clicking on the component.
    c. **Use** the wire tool from the left icon set to connect the potentiometer input to the ADC.
    d. **Note** the throughout this tutorial, components are renamed by double-clicking and typing a simple one-word name. **Remember** that these names become the object names when writing code. In this case, the ADC has been renamed “ADC”.



```
Log file for this session is located at: C:\Users\meier\AppData\Local\Temp\PSoC Creator-000.log
```

See additional images for step 5 on the next page



Configure 'ADC_SAR'

Name: ADC

Configure | Built-in

Modes

Resolution (bits): 12

Conversion rate (SPS): 100000

Actual conv. rate (SPS): 100000

Clock frequency (kHz): 1600

Sample mode

● Free running

○ Software trigger

○ Hardware trigger

Clock source

● Internal

○ External

Input

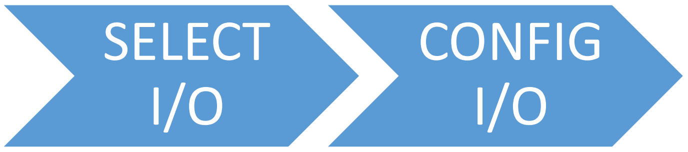
Input range: Vssa to Vdda (Single Ended)

Reference: Internal Vref

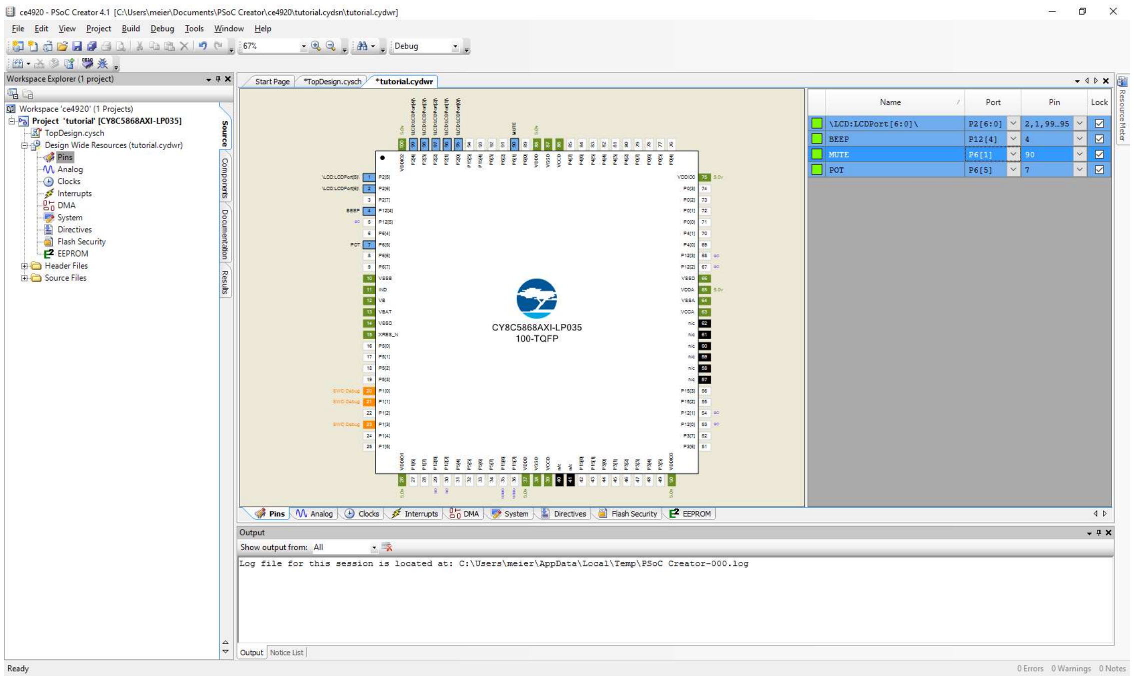
Voltage reference: 2.5000 Volts (Vdda/2)

☐ Enable EOS output

- **This completes selection and configuration of the system inputs and outputs. These are the first two steps in the Cypress PSOC design flow. The next step is pin assignment.**



7. **Assign** the pins of the design using the **Pins** section under Design Wide Resources.
    a. Development board silkscreen shows assigned pins for most components.
    b. **Note** that the potentiometer pin is not silkscreened.
    c. **Consult** the development board user-manual for pin assignments.
    d. The LCD header is connected to pins P2[6:0].
    e. A female header at pin P12[4] is selected to host the piezo-speaker BEEP signal.
    f. Pushbutton SW2 connected to pin P6[1] is selected to host the MUTE signal.
    g. The user-manual documents potentiometer connection at pin P6[5].



ce4920 - PSoC Creator 4.1 [C:\Users\meier\Documents\PSoC Creator\ce4920\tutorial.cydsn\tutorial.cydwr]

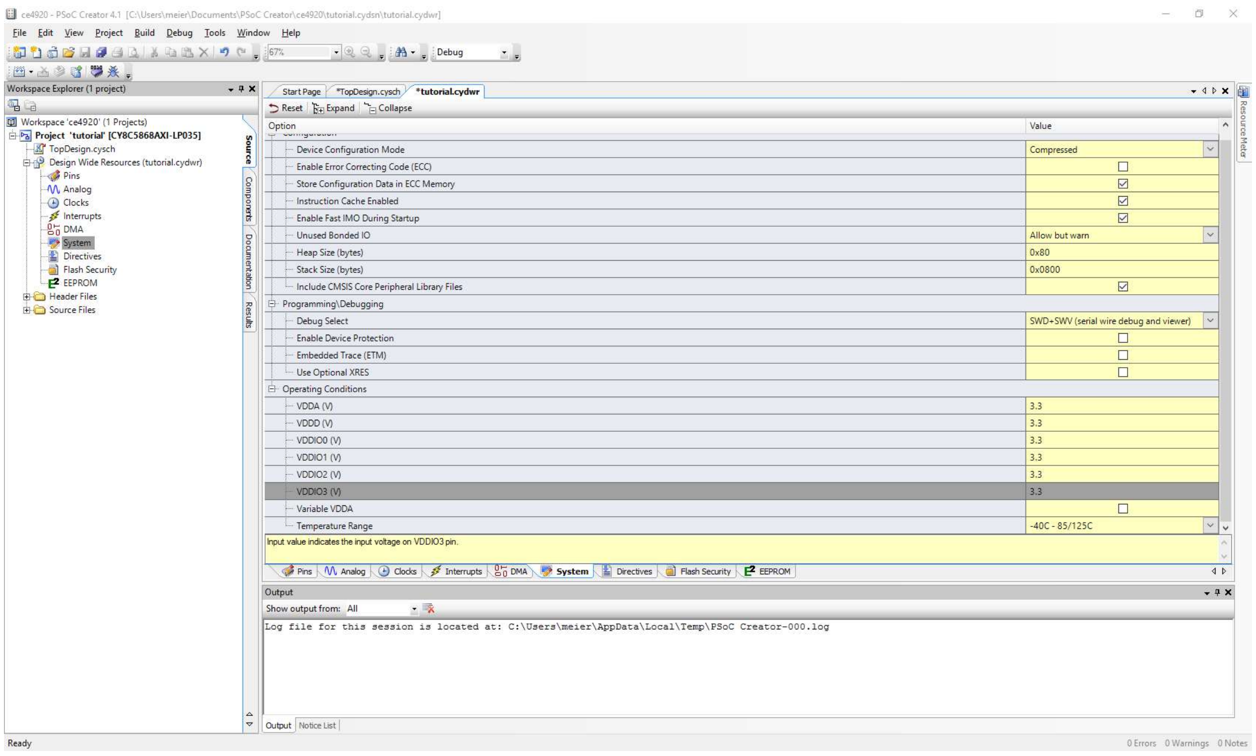
Workspace 'ce4920' (1 Projects)
Project 'tutorial' [CY8C5868AXI-LP035]

	Name	Port	Pin	Lock
🟩	\LCD:LCDPort[6:0]\	P2[6:0]	2,1,99..95	✓
🟩	BEEP	P12[4]	4	✓
🟩	MUTE	P6[1]	90	✓
🟩	POT	P6[5]	7	✓

```
Log file for this session is located at: C:\Users\meier\AppData\Local\Temp\PSoC Creator-000.log
```

SELECT I/O
CONFIG I/O
ASSIGN PINS

8. **Configure** the power supplies of the design using the **System** section under Design Wide Resources.
   a. **Locate** supply-voltage settings in the system section.
   b. **Set** the supply-voltages to 3.3V.


ce4920 - PSoC Creator 4.1 [C:\Users\meier\Documents\PSoC Creator\ce4920\tutorial.cydsn\tutorial.cydwr]

Workspace Explorer (1 project)

- Workspace 'ce4920' (1 Projects)
  - Project 'tutorial' [CY8C5868AXI-LP035]
    - TopDesign.cysch
    - Design Wide Resources (tutorial.cydwr)
      - Pins
      - Analog
      - Clocks
      - Interrupts
      - DMA
      - System
      - Directives
      - Flash Security
      - EEPROM
    - Header Files
    - Source Files

Option	Value
Device Configuration Mode	Compressed
Enable Error Correcting Code (ECC)	☐
Store Configuration Data in ECC Memory	☑
Instruction Cache Enabled	☑
Enable Fast IMO During Startup	☑
Unused Bonded IO	Allow but warn
Heap Size (bytes)	0x80
Stack Size (bytes)	0x0800
Include CMSIS Core Peripheral Library Files	☑
Programming\Debugging	
Debug Select	SWD+SWV (serial wire debug and viewer)
Enable Device Protection	☐
Embedded Trace (ETM)	☐
Use Optional XRES	☐
Operating Conditions	
VDDA (V)	3.3
VDDD (V)	3.3
VDDIO0 (V)	3.3
VDDIO1 (V)	3.3
VDDIO2 (V)	3.3
VDDIO3 (V)	3.3
Variable VDDA	☐
Temperature Range	-40C - 85/125C

Input value indicates the input voltage on VDDIO3 pin.

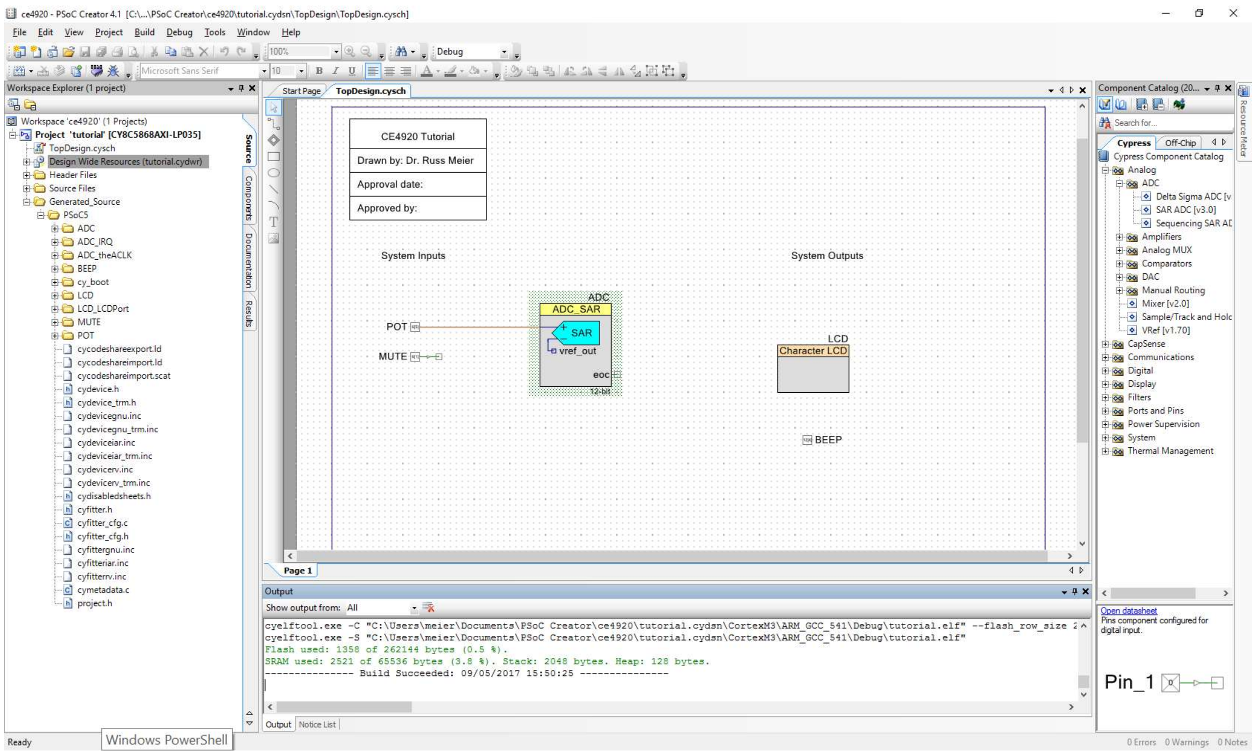
```
Log file for this session is located at: C:\Users\meier\AppData\Local\Temp\PSoC Creator-000.log
```

Ready — 0 Errors 0 Warnings 0 Notes

SELECT I/O
CONFIG I/O
ASSIGN PINS
SELECT POWER

CE4920 PSOC TUTORIAL

9. **Build** the initial design. There are three ways to start a build.
   a. **Use** the build icon in the toolbar.
   b. **Use** the keyboard shortcut Shift+F6.
   c. **Use** the build option in the build menu.
   d. PSOC Creator generates significant C source code during the initial design build.
   e. PSOC Creator leaves a main control function in the file **main.c**.
   f. **Write** your firmware in the **main.c** file.
   g. **Add** and **use** additional header and C files when appropriate.



```
cyelftool.exe -C "C:\Users\meier\Documents\PSoC Creator\ce4920\tutorial.cydsn\CortexM3\ARM_GCC_541\Debug\tutorial.elf" --flash_row_size 2
cyelftool.exe -S "C:\Users\meier\Documents\PSoC Creator\ce4920\tutorial.cydsn\CortexM3\ARM_GCC_541\Debug\tutorial.elf"
Flash used: 1358 of 262144 bytes (0.5 %).
SRAM used: 2521 of 65536 bytes (3.8 %). Stack: 2048 bytes. Heap: 128 bytes.
--------------- Build Succeeded: 09/05/2017 15:50:25 ---------------
```

SELECT I/O
CONFIG I/O
ASSIGN PINS
SELECT POWER
CODE

```c
/* ========================================
 * FILENAME: subroutines.h
 * AUTHOR:   meier@msoe.edu <Dr. Meier>
 * PROVIDES:
 * - header file for PSOC tutorial
 * ========================================
*/

#ifndef CE4920_SUBROUTINES
#define CE4920_SUBROUTINES

void initializeSystem(void);
void printWelcomeMessage(void);
void printUserInstructions(void);
int calculatePeriodDelay(int frequency);

#endif
```

```
/* ========================================
 * FILENAME: subroutines.c
 * AUTHOR:   meier@msoe.edu <Dr. Meier>
 * PROVIDES:
 * - support subroutines for PSOC tutorial
 * ========================================
*/

#include "subroutines.h"
#include <project.h>

void initializeSystem(void)
{
  LCD_Start();
  ADC_Start();
  ADC_StartConvert();
}

void printWelcomeMessage(void)
{
    // artifacts:
    // - clears display
    // - leaves message on top row
    LCD_ClearDisplay();
    LCD_Position(0,1);
    LCD_PrintString("TONE GENERATOR");
    CyDelay(1000);
}

void printUserInstructions(void)
{
    // position on 16x2 LCD
    // delay values in ms
    // artifacts:
    // - overwrites screen row 1
    // - locks for multiple seconds
    LCD_Position(1,0);
    LCD_PrintString("  Potentiometer");
    CyDelay(1000);
    LCD_Position(1,0);
    LCD_PrintString("  Adjusts the  ");
    CyDelay(1000);
    LCD_Position(1,0);
    LCD_PrintString("Tone Frequency!");
    CyDelay(1000);
    LCD_Position(1,0);
    LCD_PrintString("Push switch SW2");
    CyDelay(1000);
    LCD_Position(1,0);
    LCD_PrintString("To Mute Sounds!");
    CyDelay(1000);
    LCD_Position(1,0);
    LCD_PrintString("                ");
    LCD_Position(1,0);
    LCD_PrintString("Freq = ");
}

int calculatePeriodDelay(int frequency)
{
    // put a boundary on the frequency
    if(frequency < 300) frequency = 300;
    if(frequency > 3000) frequency = 3000;
    return (int)(1000000.0/(2.0*frequency)); // type cast to int
}

/* [] END OF FILE */
/* ========================================
 * FILENAME: main.c
 * AUTHOR:   meier@msoe.edu
 * DATE:     14 Sep 2015
 * PROVIDES:
 * - simple firmware to help CE4920
 *   students learn to use PSOC 5LP
 *
 * - main is a three state FSM
 * - state advancement
 * - software pushbutton debouncing
 * - LCD panel updates
 * - user subroutines
 * ========================================
*/


#include <project.h>
#include "subroutines.h"


enum state { reset, playTone, mute } systemState;


// WARNING: This code was quickly sketched.
//          It may still contain bugs.

int main()
{
 //   CyGlobalIntEnable; /* Enable global interrupts. */
    int toneFrequency = 0;
    int toneHalfPeriod = 0;

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */

    for(;;)
    {
        switch(systemState)
        {
            case reset:
                             initializeSystem();
                             printWelcomeMessage();
                             printUserInstructions();
                               // turn on music note icon
                             LCD_Position(1,14);
                             LCD_PutChar(LCD_CUSTOM_5);

                             systemState = playTone;
                             break;
            case playTone:
                             toneFrequency = ADC_GetResult16();
```

```
                                    LCD_Position(1,7);
                                    LCD_PrintString("     ");
                                    LCD_Position(1,7);
                                    LCD_PrintNumber(toneFrequency);
                                    toneHalfPeriod = calculatePeriodDelay( toneFrequency
);

                                    do
                                    {
                                        BEEP_Write(1);
                                        CyDelayUs(toneHalfPeriod);
                                        BEEP_Write(0);
                                        CyDelayUs(toneHalfPeriod);
                                        toneFrequency--; // make this many per second
                                    }while(toneFrequency > 0);


                                    if(MUTE_Read()==0) // check for pushbutton
                                    {
                                       int count=0;
                                       do
                                       {
                                         if(MUTE_Read()==0) count=0; // reset to 0 if
bounces back down

                                         count++;
                                       }while(count < 100); // good software debounce.

                                       systemState=mute;
                                       LCD_Position(1,14);
                                       LCD_PrintString(" "); // turn off music note
                                    }

                                    break;
            case mute:
                                    if(MUTE_Read()==0)
                                    {
                                      int count=0;
                                       do
                                       {
                                         // reset to 0 if bounces back down
                                         if(MUTE_Read()==0) count=0;
                                         count++;
                                       }while(count < 100); // good software debounce.

                                        // turn on music note icon
                                      LCD_Position(1,14);
                                      LCD_PutChar(LCD_CUSTOM_5);

                                      systemState=playTone;

                                    }
                                   break;
```

```
        }

        /* Place your application code here. */
    }
}

/* [] END OF FILE */
```