Introduction to PSoC Creator for 6.131 and 6.115

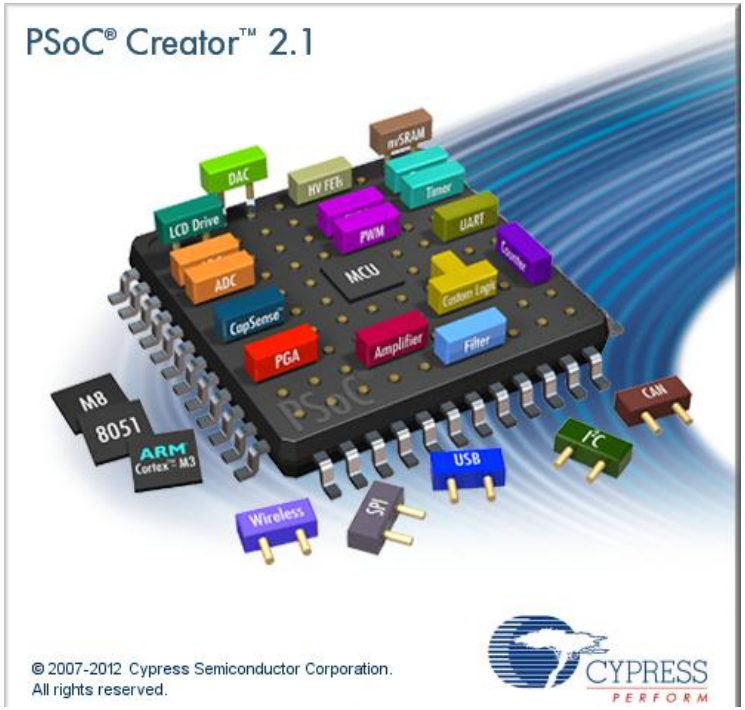
Massachusetts Institute of Technology

Department of Electrical Engineering and Computer Science

Section Objectives

You will be able to:

- Follow the PSoC Creator Design Flow and develop projects
- Find and use the tools available within the software IDE
- Compile, build and program PSoC 3/5 applications
- Debug PSoC 3/5 applications



PSoC Creator Design Flow

Configure

- Start a new project
- Place components
- Configure components
- Connect components

Develop

- Build hardware design and generate component APIs
- Write application code utilizing component APIs
- Compile, build and program

Debug

- Perform in-circuit debug using the MiniProg3 and PSoC Creator

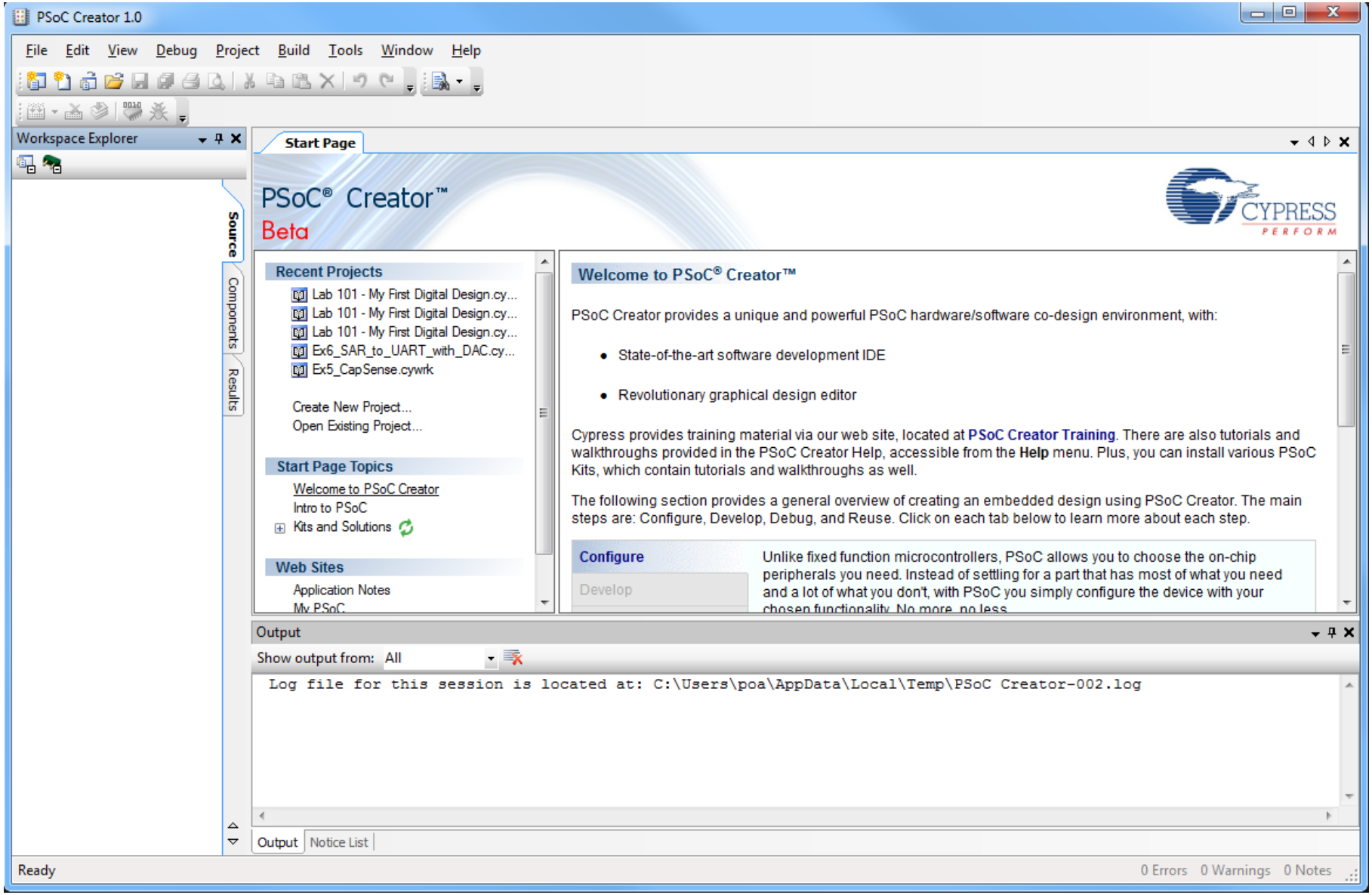
Reuse

- Capture working hardware/software designs as your own components for future use

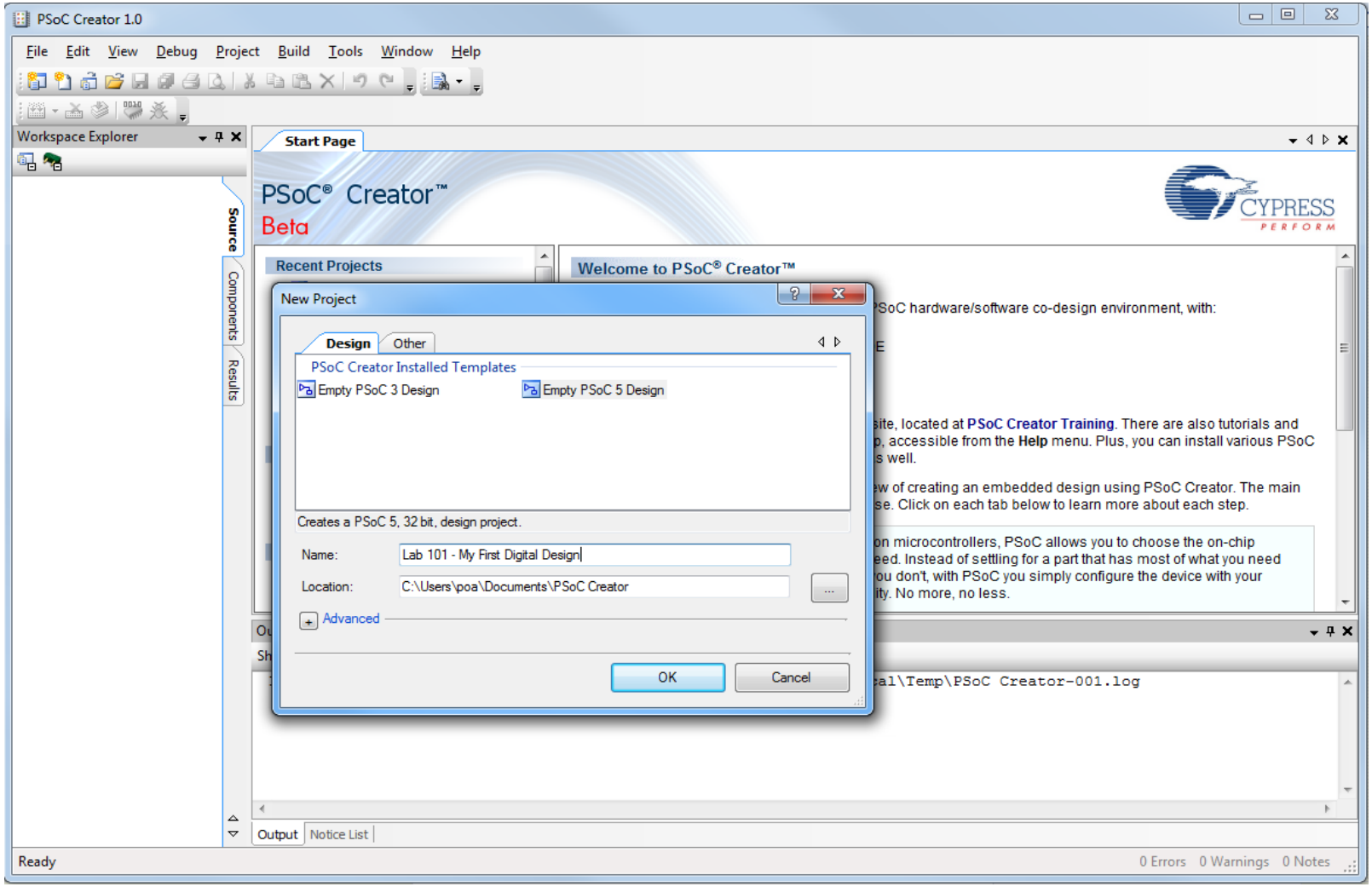
Step 1: Download PSoC Creator

1. Go to www.cypress.com
2. Go to Products→Programmable System-on-chip→PSoC Software
3. Click on PSoC Creator. (PSoC Creator is the IDE for PSoC 3 and 5 designs, PSoC Designer is for PSoC 1 designs)
4. Click download. Scroll to the bottom of the page and click on the first link.
5. You will be prompted to login. Create an account to do so.
6. Use the download manager. Click “launch.” Install the typical version of Creator. Run the update manager.

Step 2: Start PSoC Creator

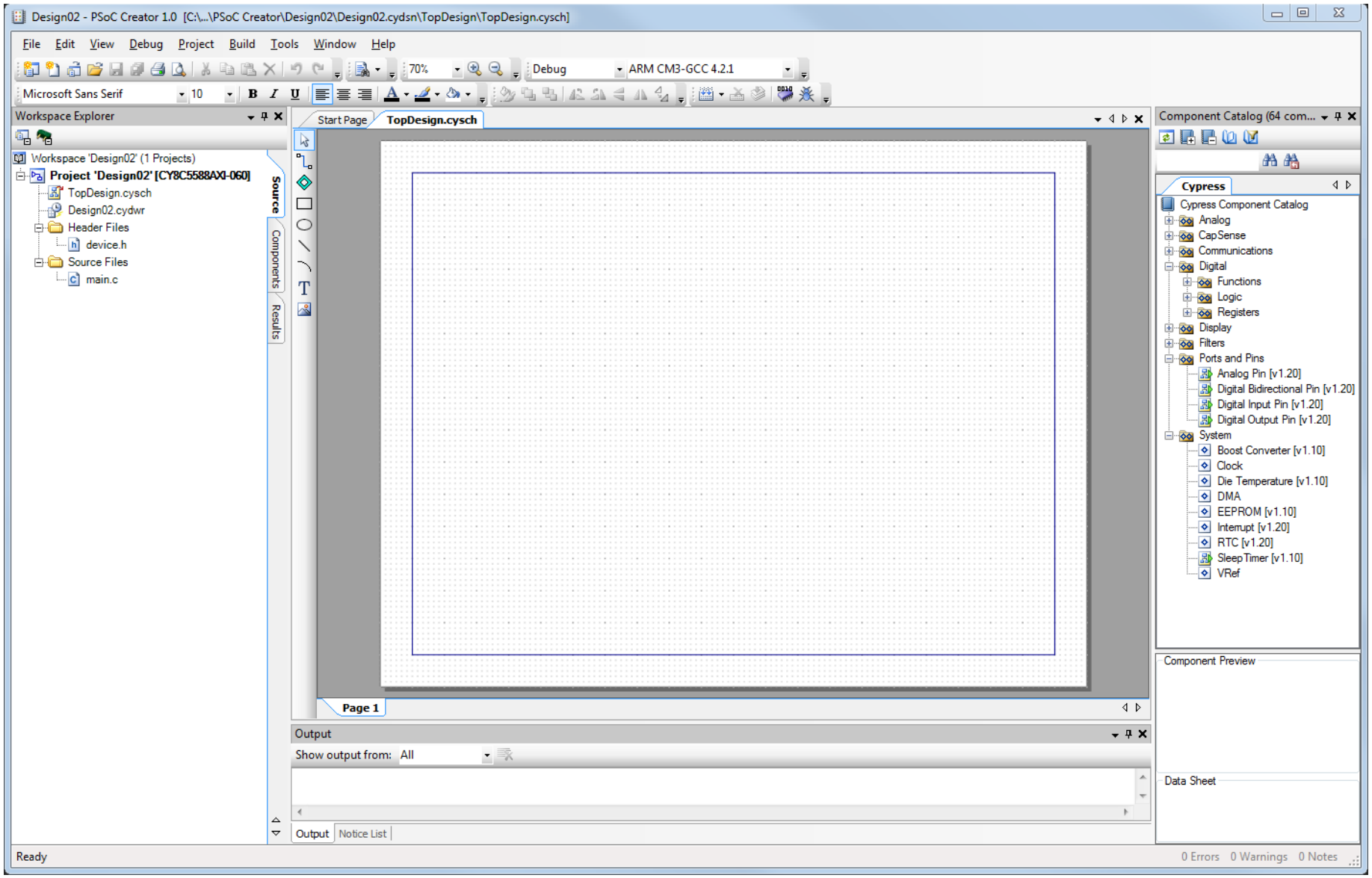


Step 3: Create a New Project

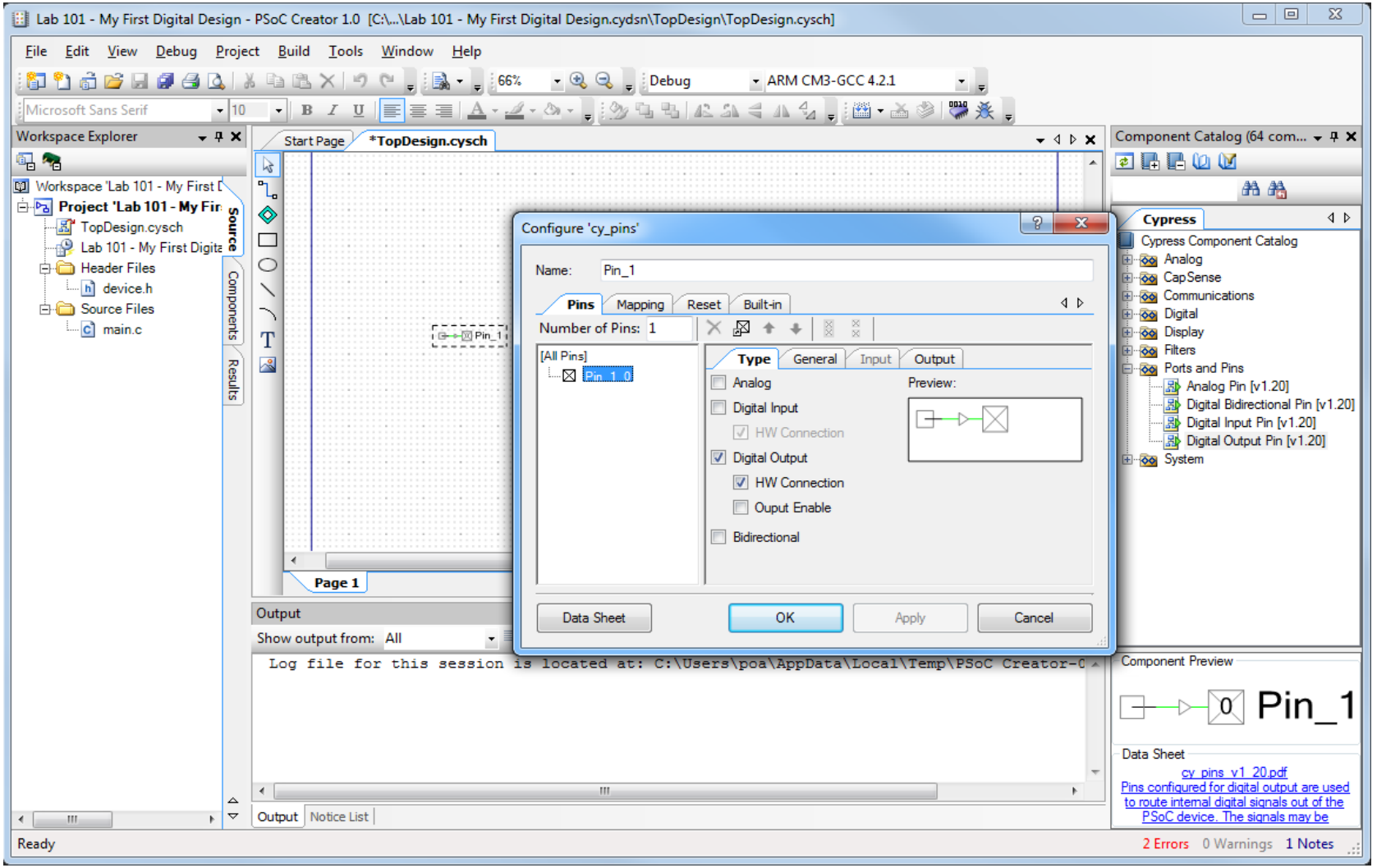


File→New→New Project

PSoC Creator Design Canvas



Step 4: Place/Configure Digital Pin



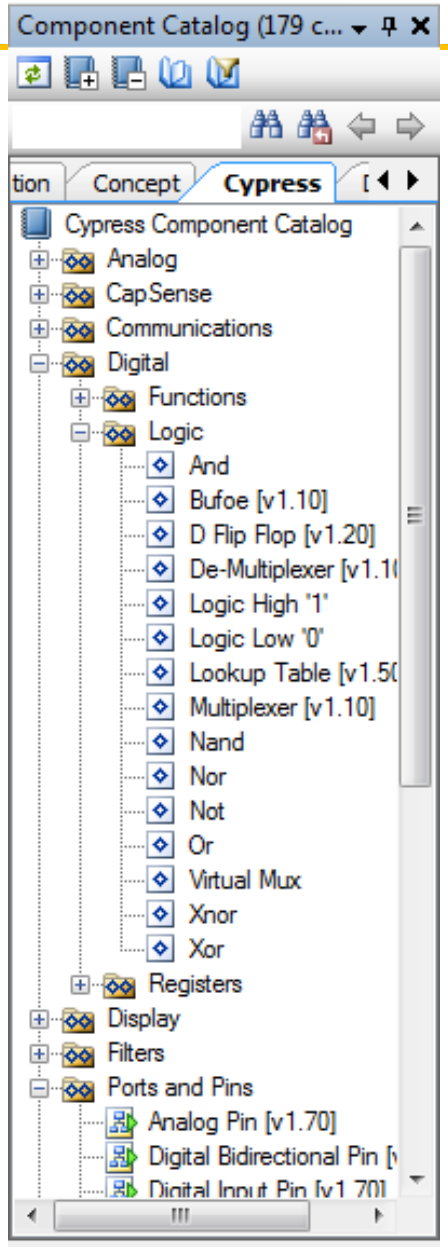
Drag pin from Component Catalog on right

Component Catalog

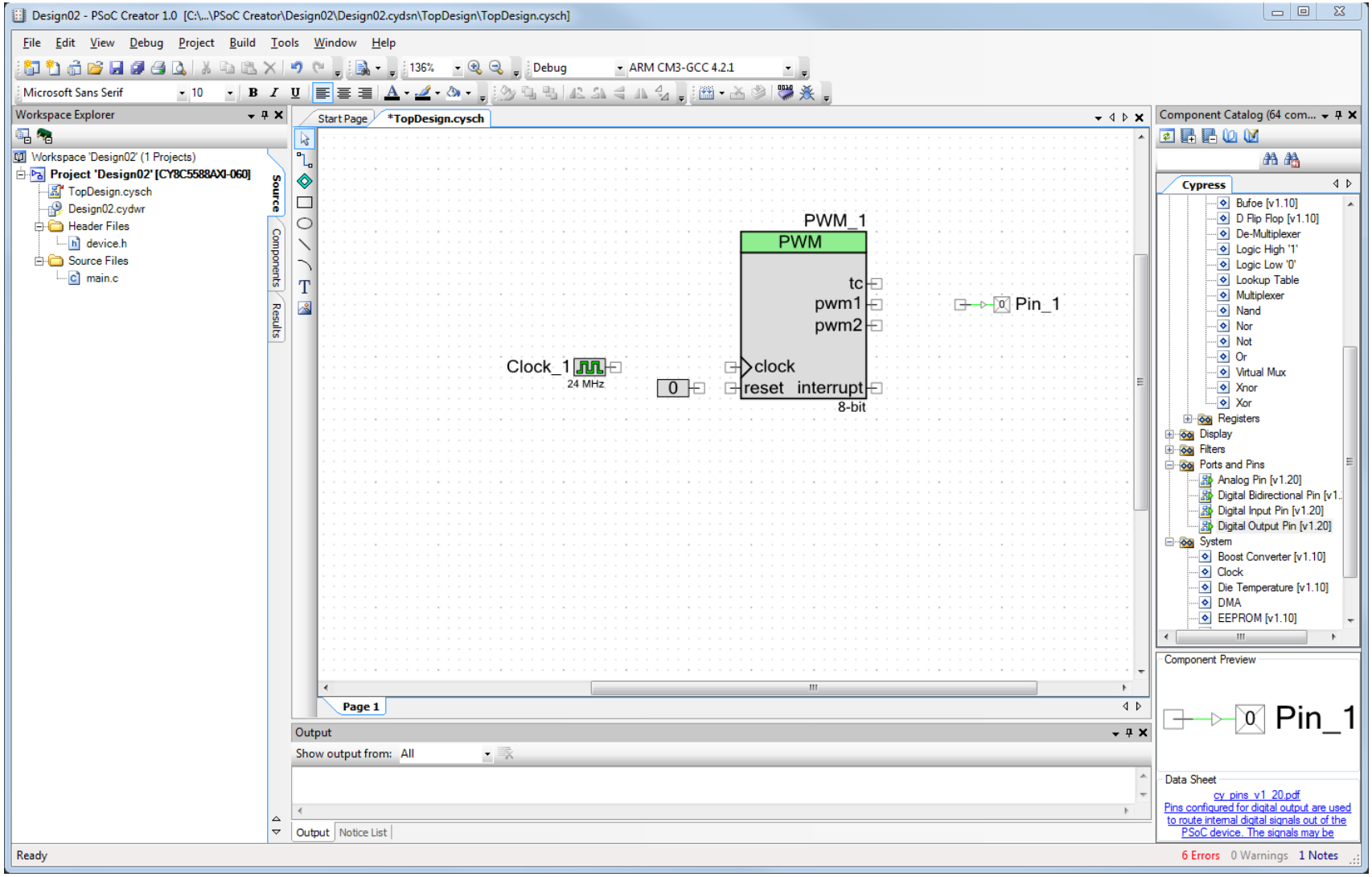
Catalog Folders

- Analog
  - ADC
  - Amplifier
  - DAC
- Digital
  - Registers
  - Functions
  - Logic
- Communication
  - UART
    - –Display
    - –System

Datasheet access



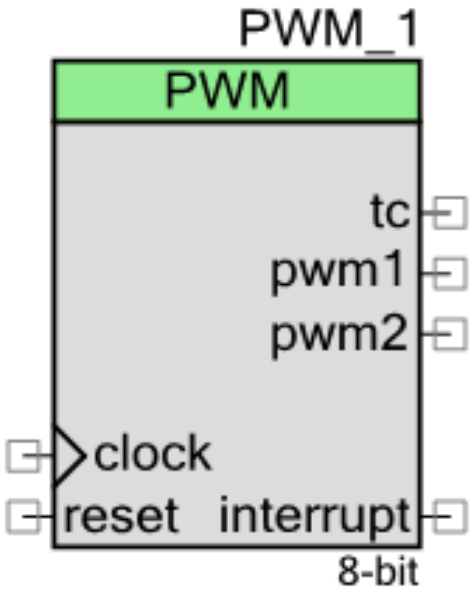
Step 5: Adding Other Components

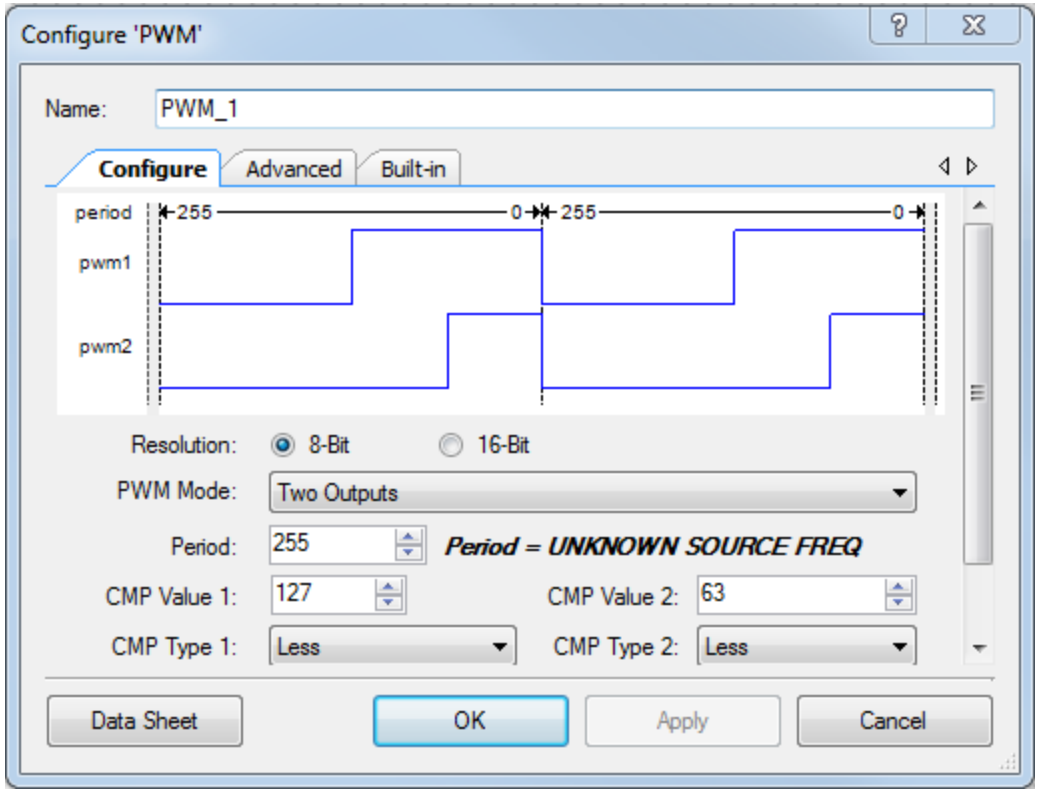


Drag needed components from Component Catalog

Step 6: Component Configuration

- Double-click on a component to open its component configuration dialog box
- Change configuration
- Click on Datasheet in bottom left corner for component description and APIs

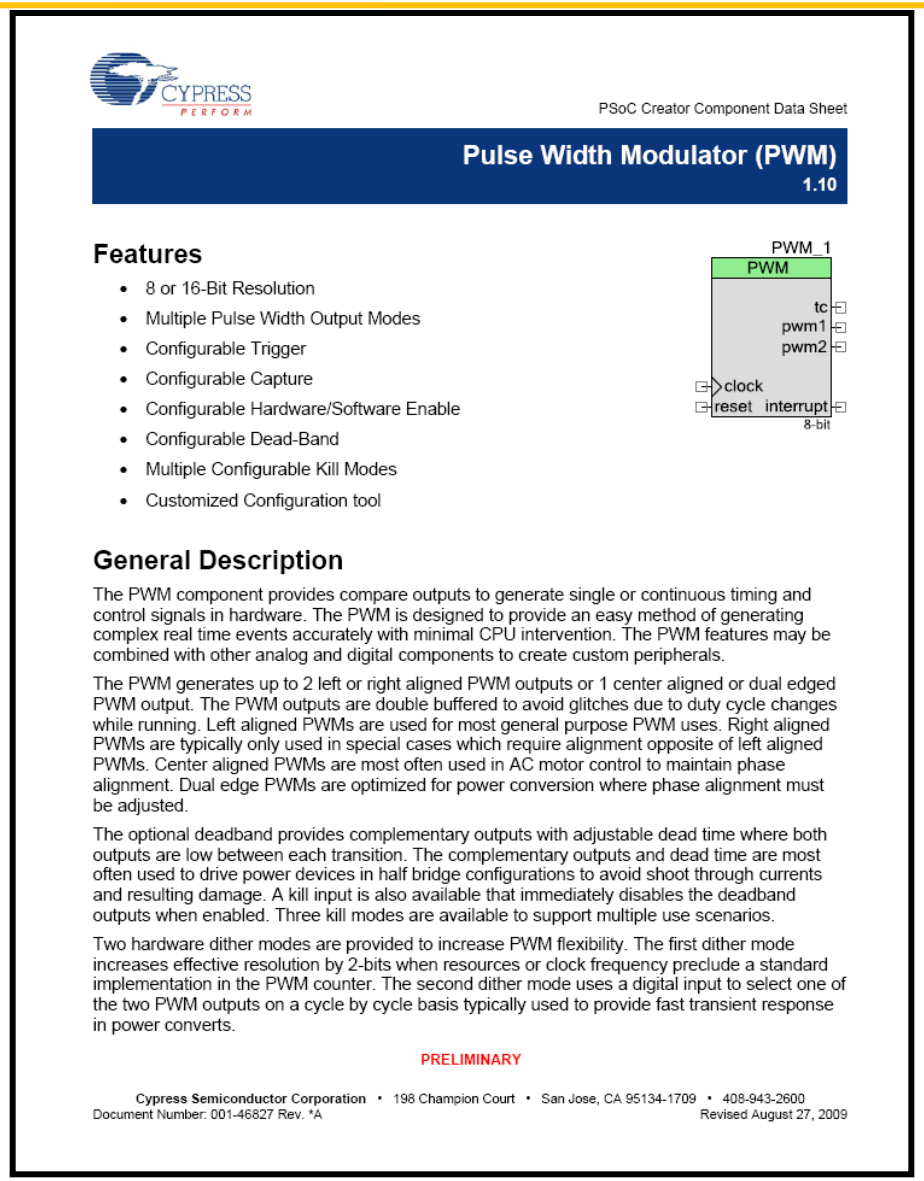




Component Datasheets

Contents:

- Features
- General description of component
- When to use component
- Input/Output connections
- Parameters and setup
- Application Programming Interface (API)
- Sample firmware source code
- Functional description
- DC and AC electrical characteristics



CYPRESS PERFORM

PSoC Creator Component Data Sheet

Pulse Width Modulator (PWM)
1.10

Features

- 8 or 16-Bit Resolution
- Multiple Pulse Width Output Modes
- Configurable Trigger
- Configurable Capture
- Configurable Hardware/Software Enable
- Configurable Dead-Band
- Multiple Configurable Kill Modes
- Customized Configuration tool

PWM_1
PWM
tc
pwm1
pwm2
clock
reset
interrupt
8-bit

General Description

The PWM component provides compare outputs to generate single or continuous timing and control signals in hardware. The PWM is designed to provide an easy method of generating complex real time events accurately with minimal CPU intervention. The PWM features may be combined with other analog and digital components to create custom peripherals.

The PWM generates up to 2 left or right aligned PWM outputs or 1 center aligned or dual edged PWM output. The PWM outputs are double buffered to avoid glitches due to duty cycle changes while running. Left aligned PWMs are used for most general purpose PWM uses. Right aligned PWMs are typically only used in special cases which require alignment opposite of left aligned PWMs. Center aligned PWMs are most often used in AC motor control to maintain phase alignment. Dual edge PWMs are optimized for power conversion where phase alignment must be adjusted.

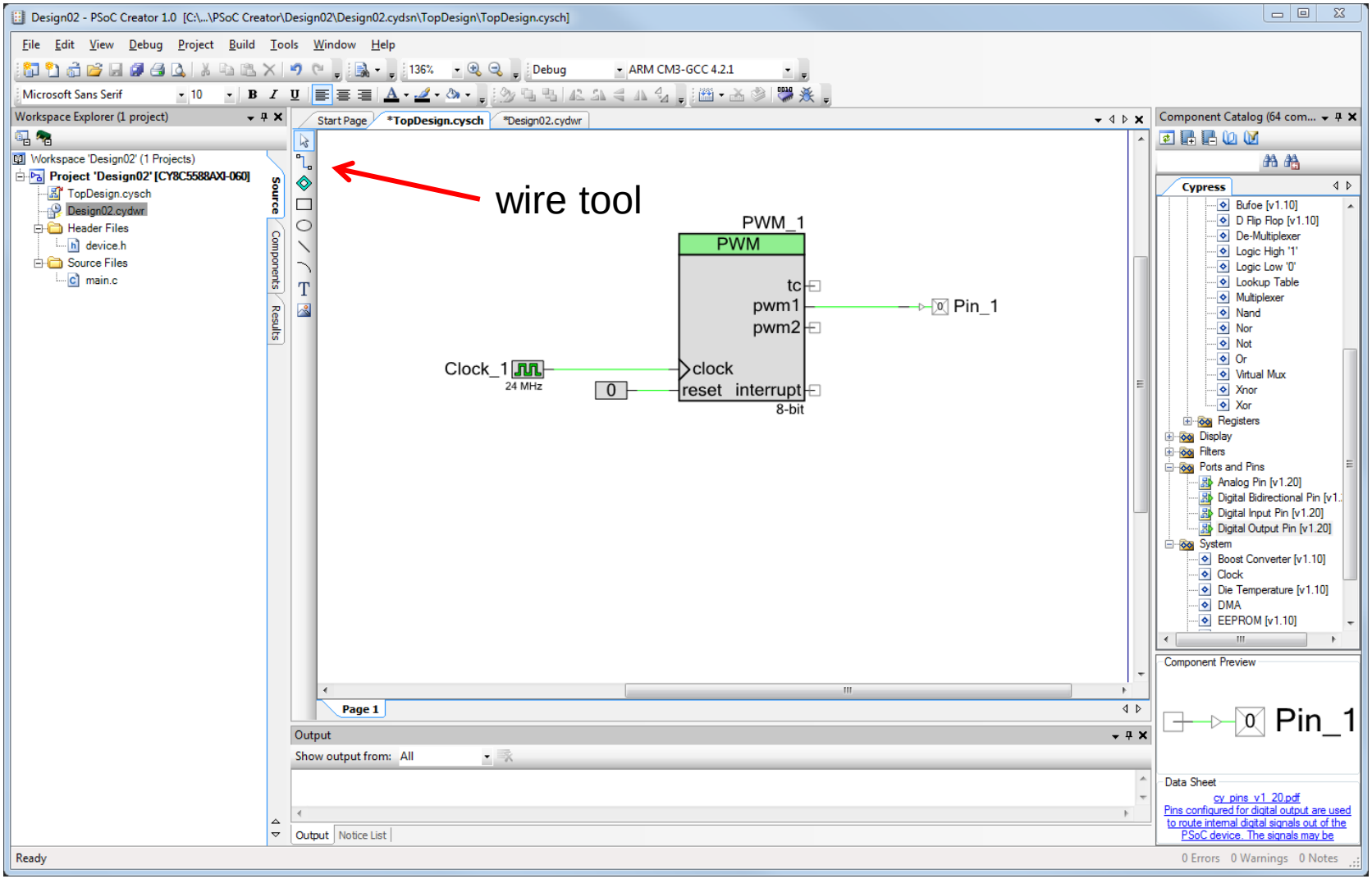
The optional deadband provides complementary outputs with adjustable dead time where both outputs are low between each transition. The complementary outputs and dead time are most often used to drive power devices in half bridge configurations to avoid shoot through currents and resulting damage. A kill input is also available that immediately disables the deadband outputs when enabled. Three kill modes are available to support multiple use scenarios.

Two hardware dither modes are provided to increase PWM flexibility. The first dither mode increases effective resolution by 2-bits when resources or clock frequency preclude a standard implementation in the PWM counter. The second dither mode uses a digital input to select one of the two PWM outputs on a cycle by cycle basis typically used to provide fast transient response in power converts.

PRELIMINARY

Cypress Semiconductor Corporation • 198 Champion Court • San Jose, CA 95134-1709 • 408-943-2600
Document Number: 001-46827 Rev. *A Revised August 27, 2009

Step 7: Connecting Components



Use wire tool (or shortcut key “w”) to connect components

Design-Wide Resource Manager (.cydwr)

Clocks

Interrupts

- Set priority and vector

DMA

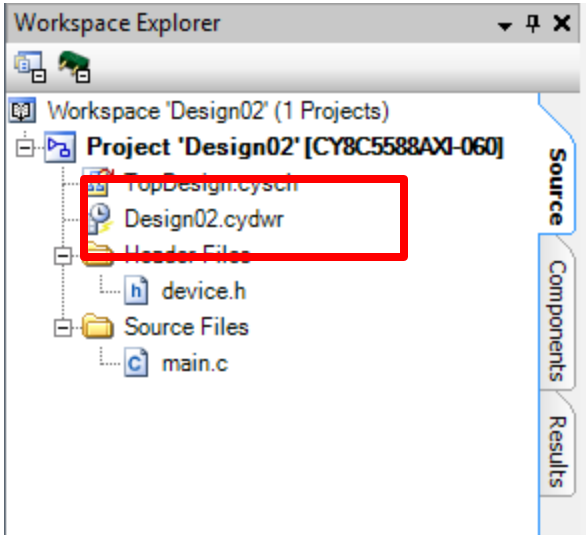
- Manage DMA channels

System

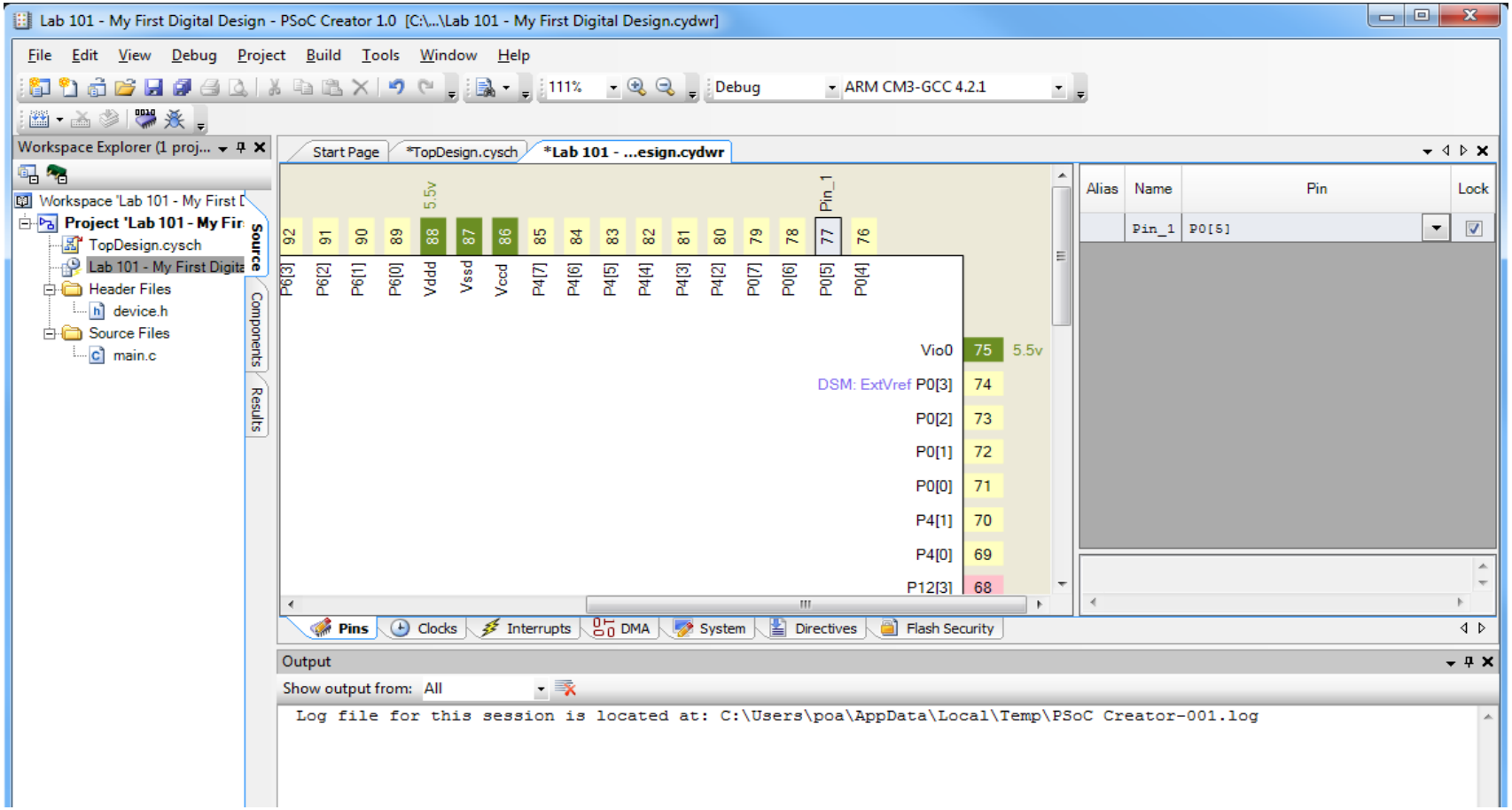
- Debug, boot parameters, sleep mode API generation, etc.

Pins

- Map I/O to physical pins and ports
- Over-ride default selections



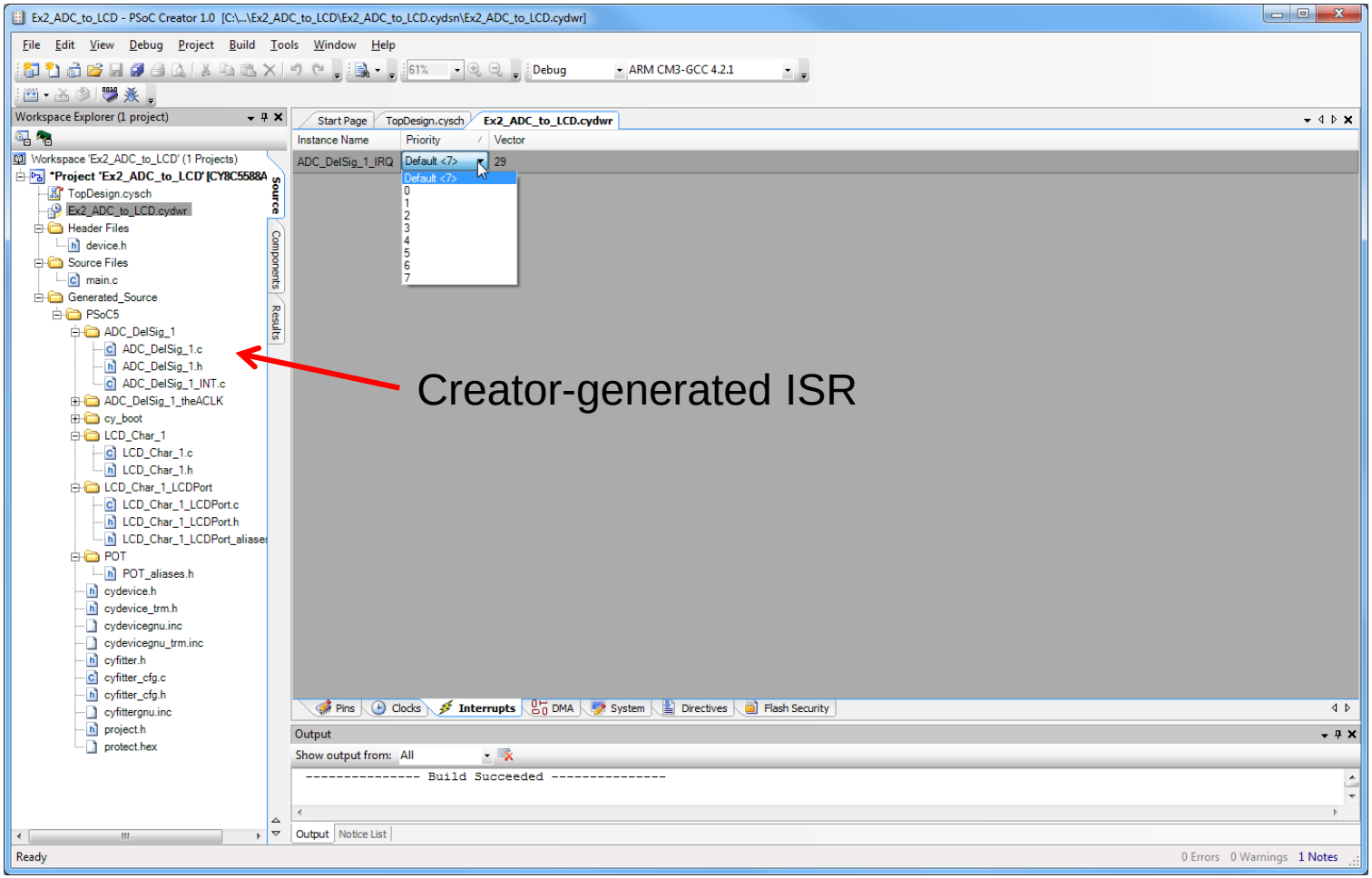
Step 8: Configure PSoC I/O



In .cydwr file, route schematic pins to physical pins on the PSoC chip (on right: name to pin mapping)

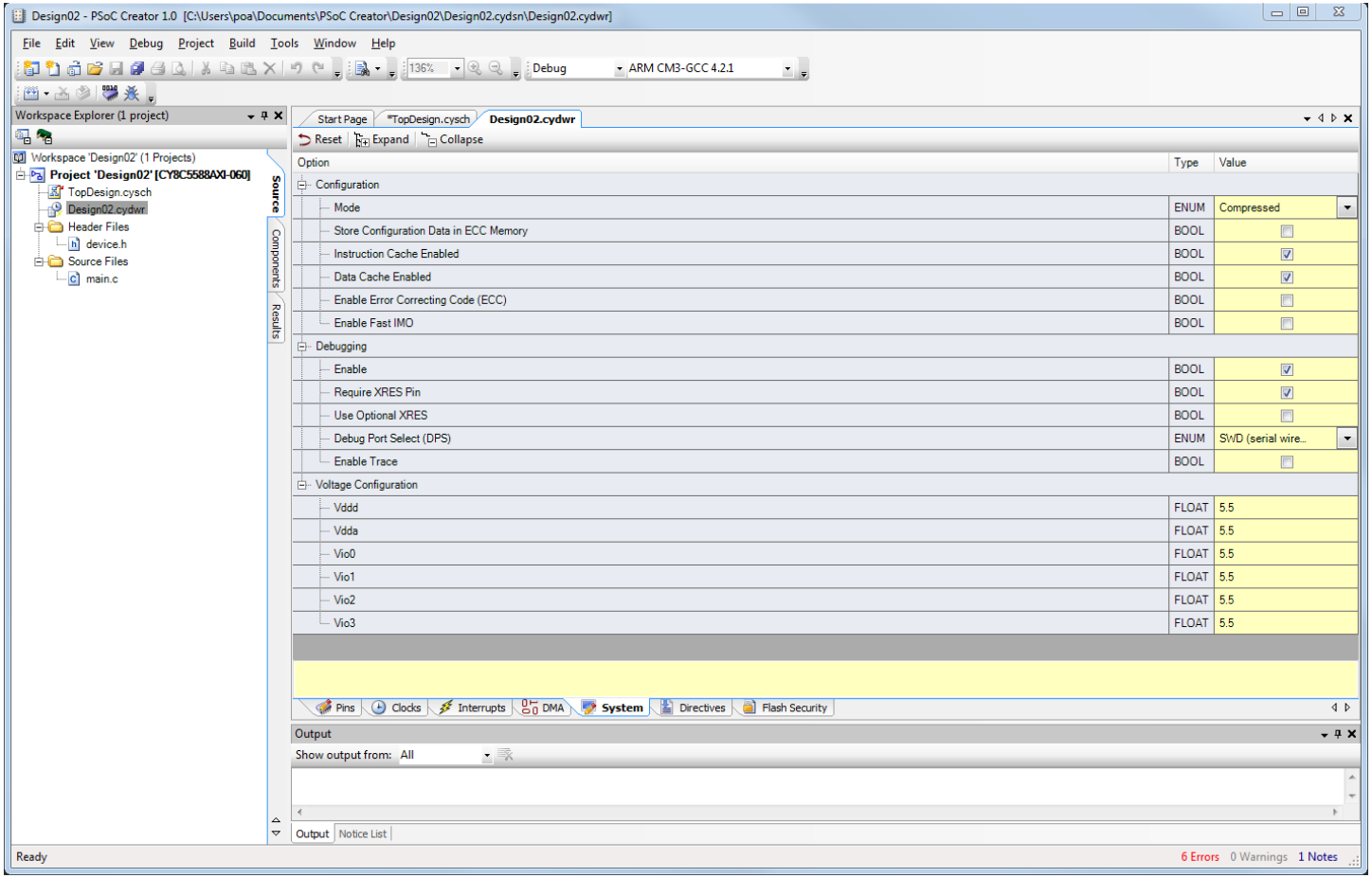
Interrupts

- Priority may be changed: defaults to 7 (lowest priority)
- Edit ISR code in interrupt Creator-generated .c file



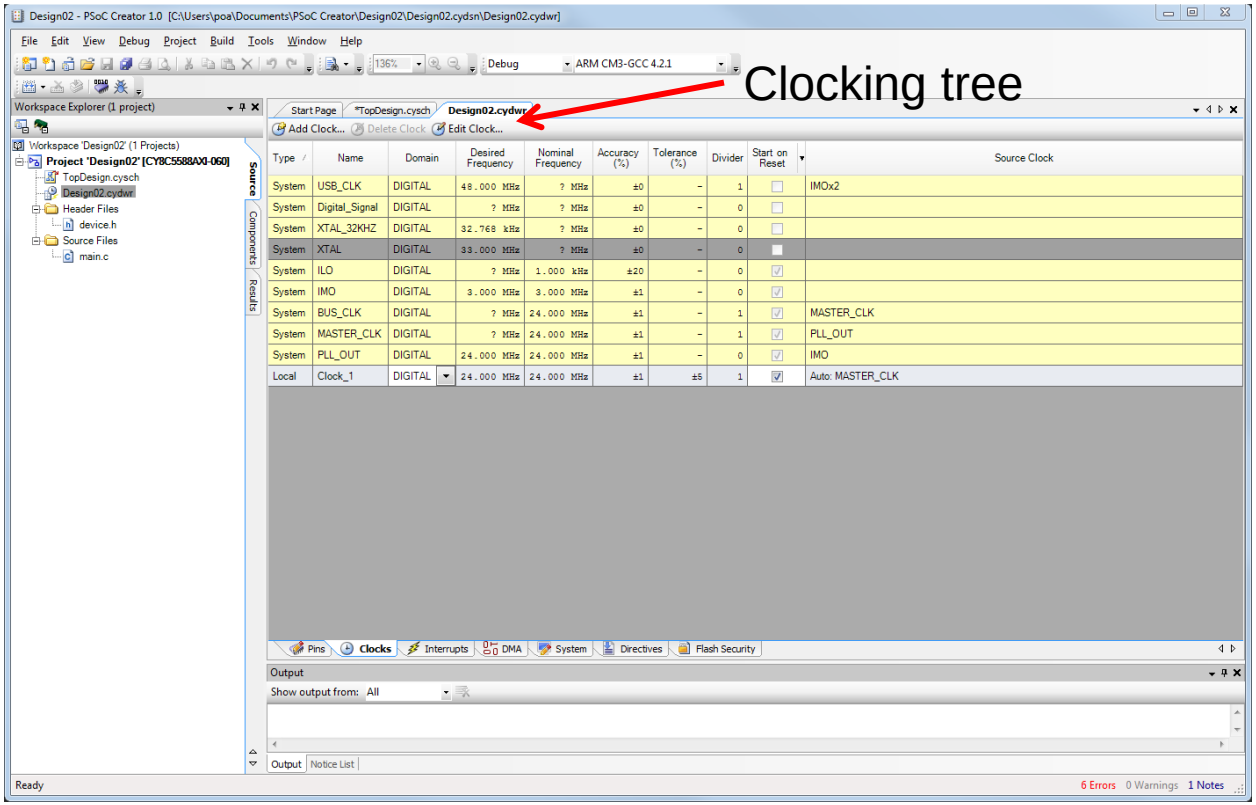
System

- System settings
- Debug settings
- Voltage Configuration

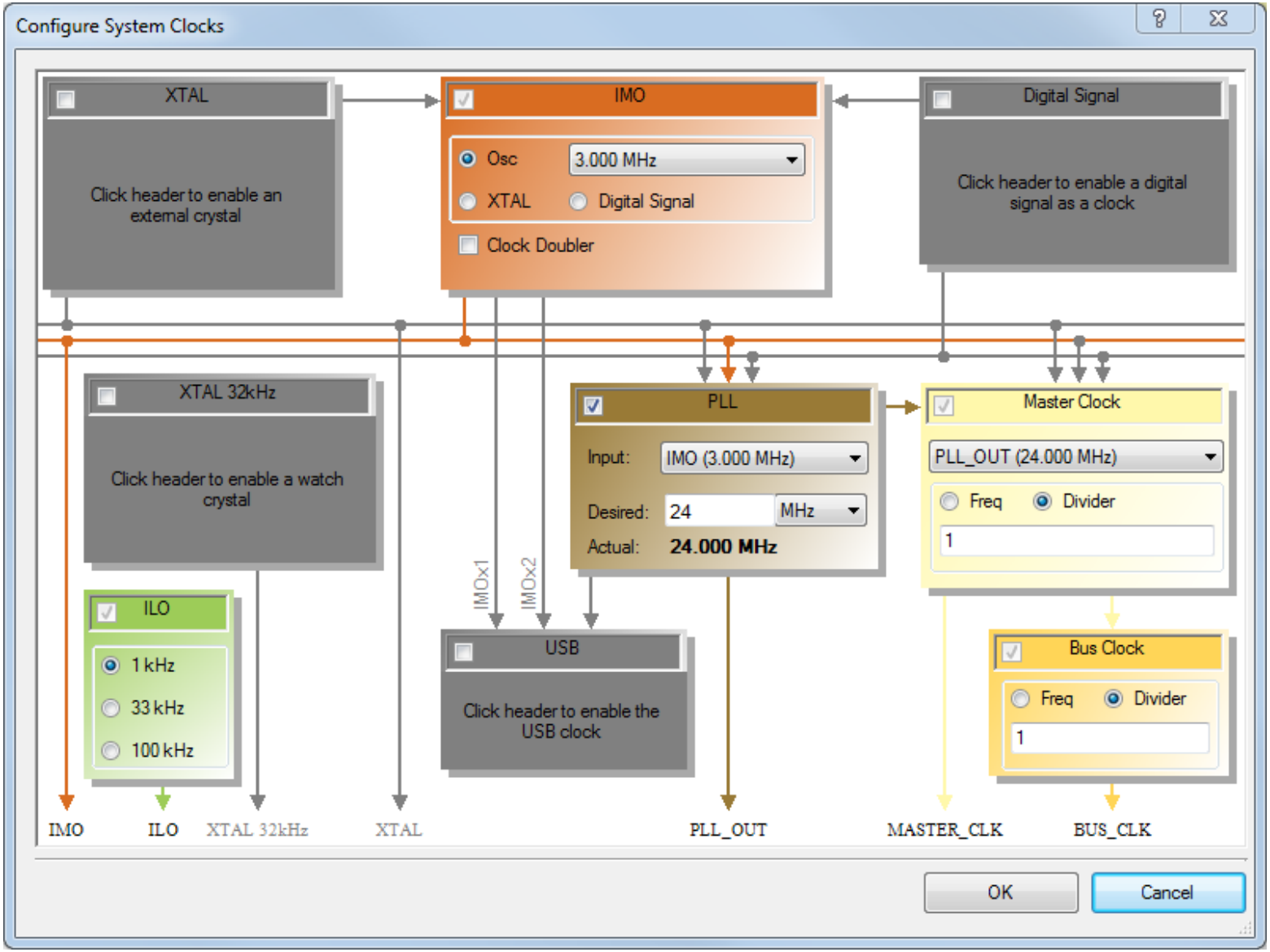


Clock Configurations

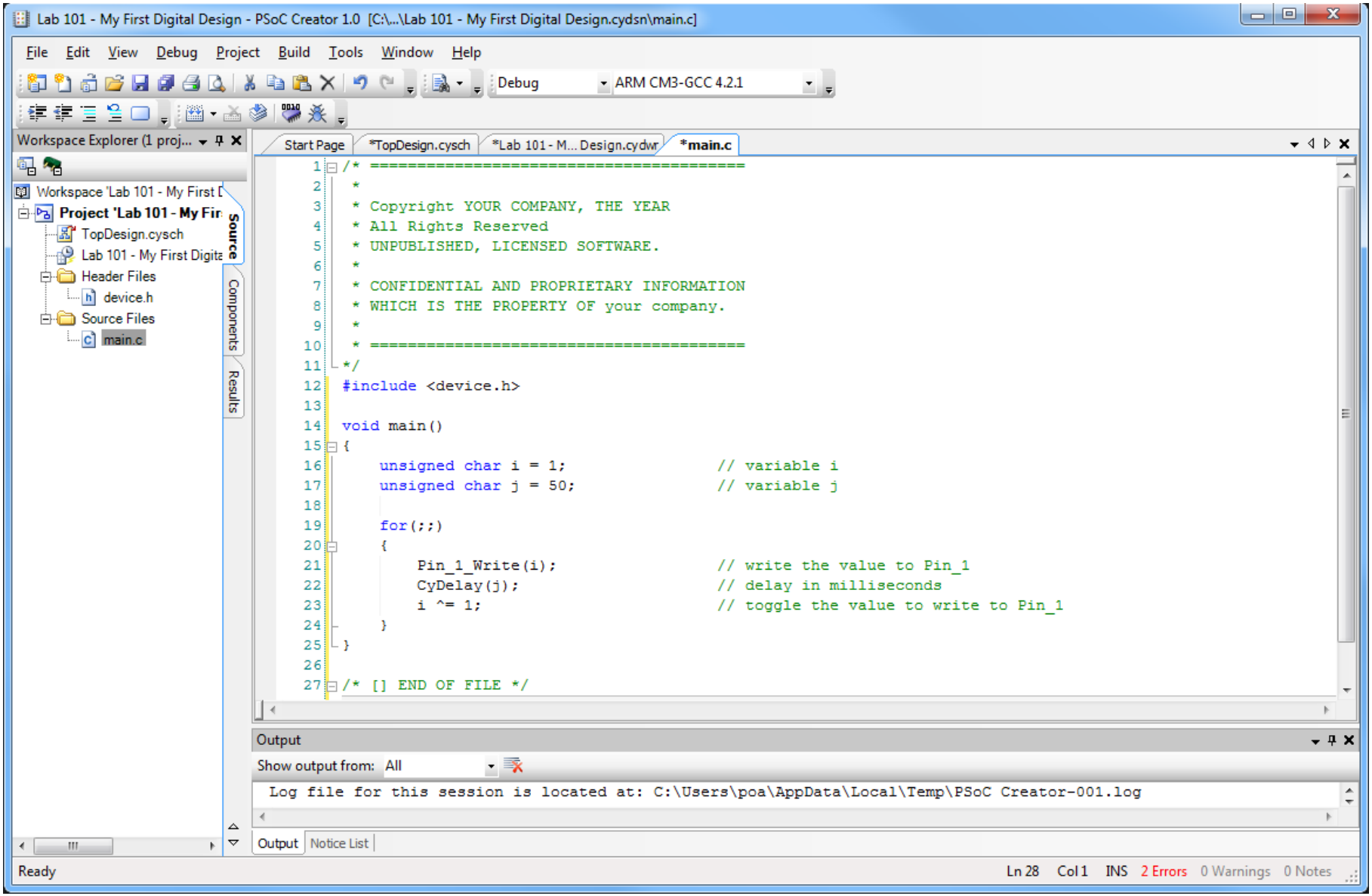
- Clocks are allocated to slots in the clock tree
- 8 digital, 4 analog
- Clocks have APIs
- Reuse existing clocks to preserve resources, if possible



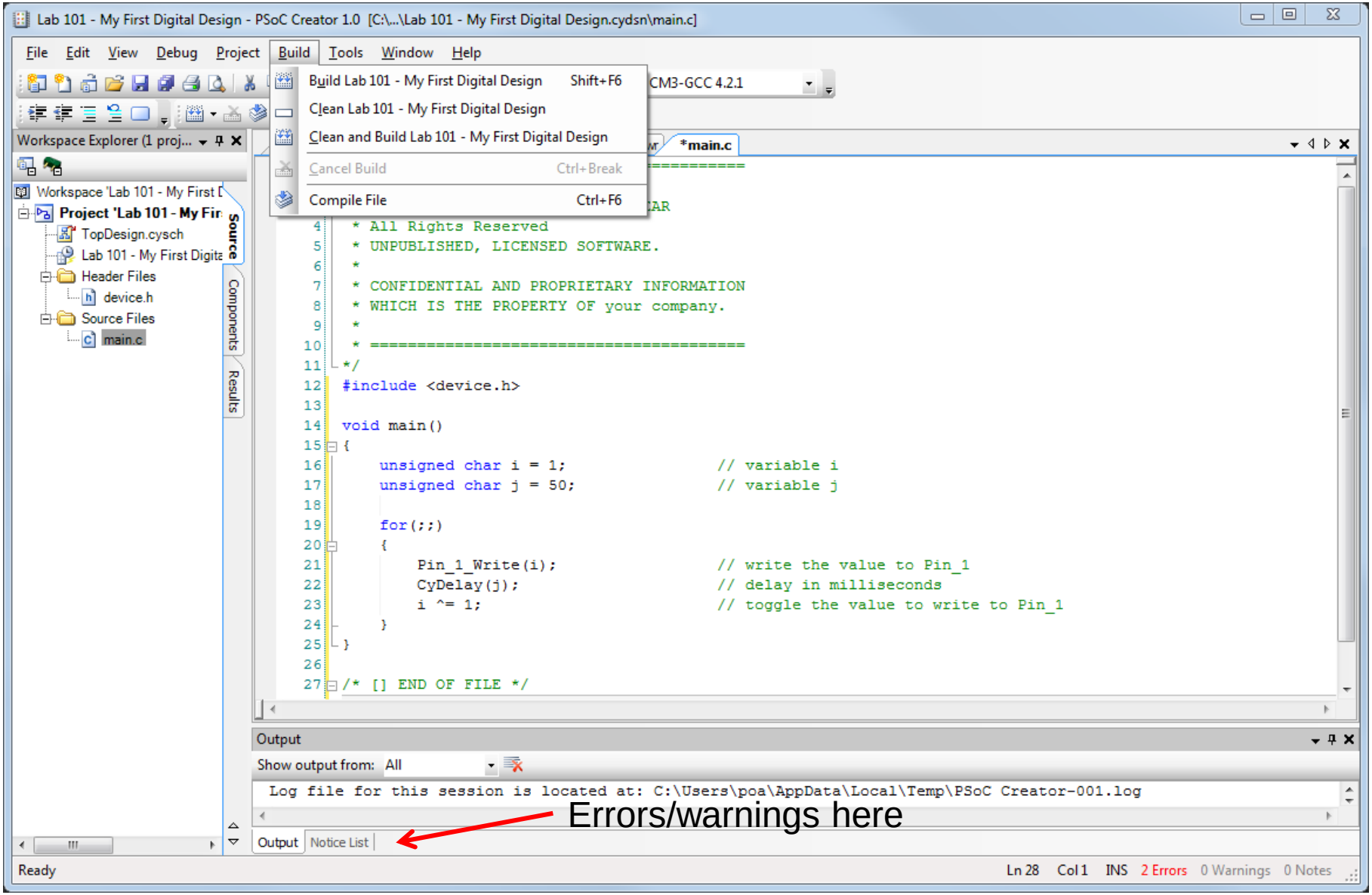
System Clocking Tree



Step 8: Add main.c Code



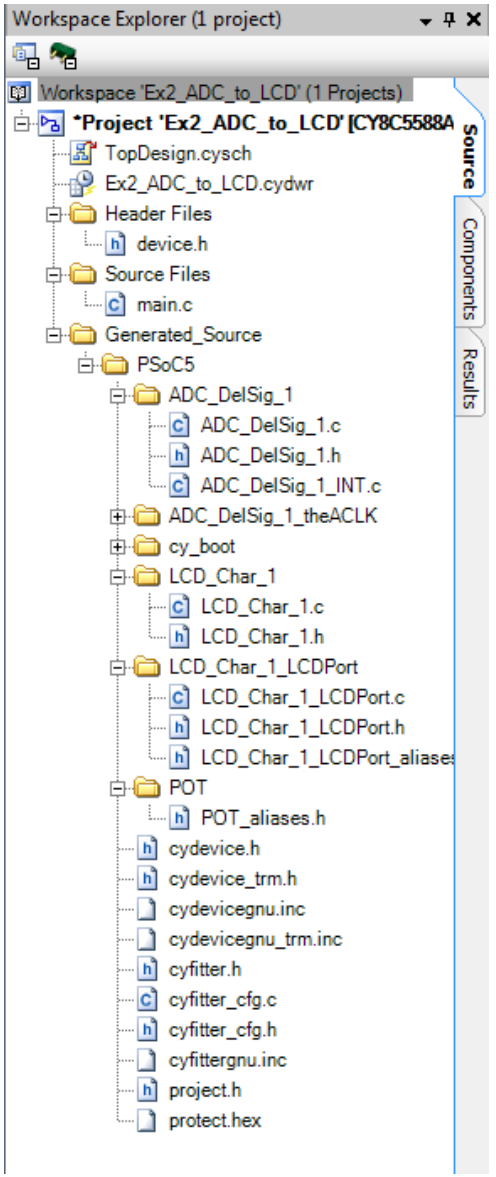
Step 9: Build Project



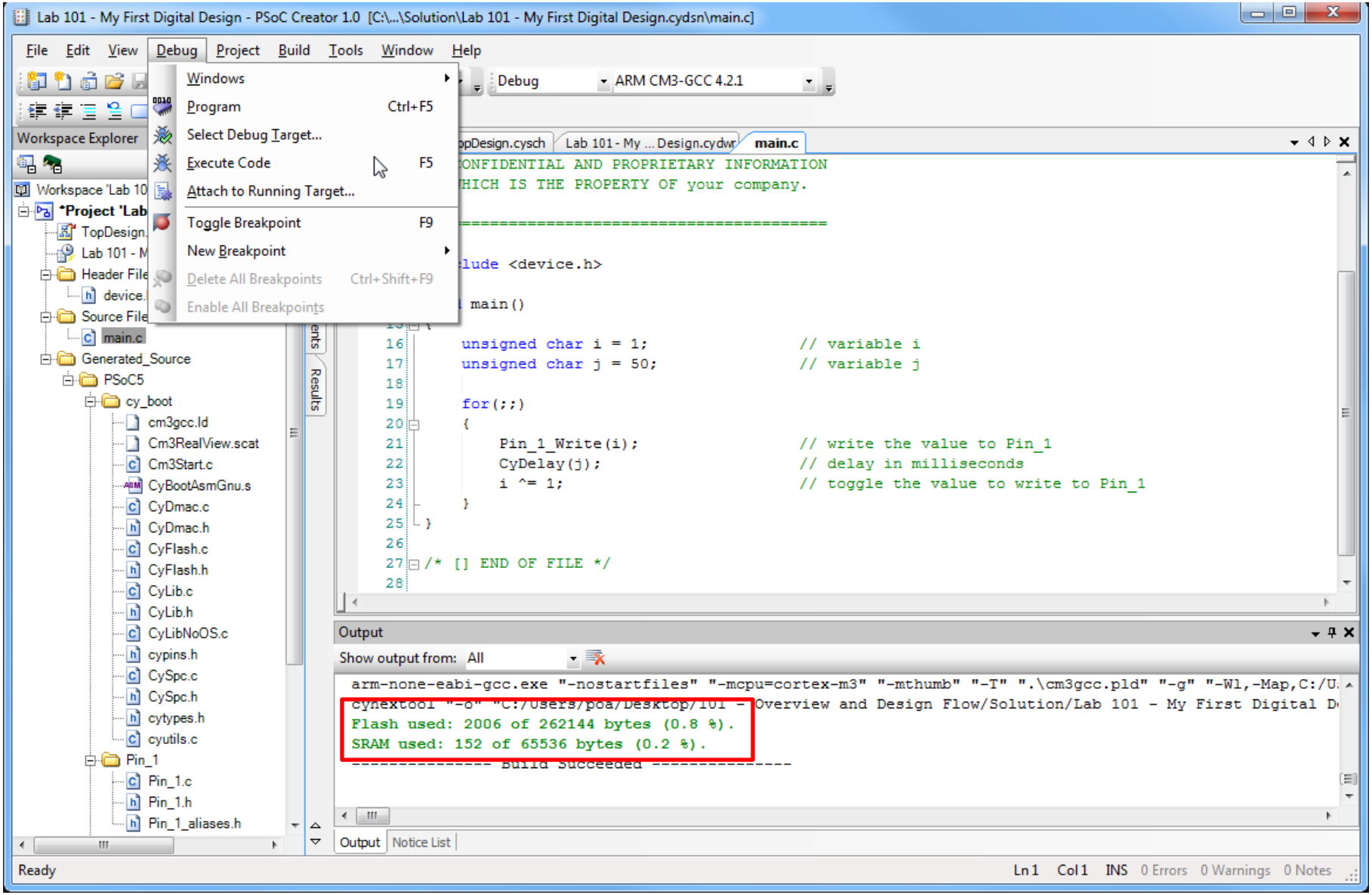
Step 9: Build Process

- API Generation
- Compilation
- Configuration Generation
- Configuration Verification

Generated APIs



Step 10: Program Device



Step 11: Debug