PSoC Technology

Team 1

Cecilia Acosta
Brett Donlon
Matt Durak
Aaron Thomson
Nathan Ward

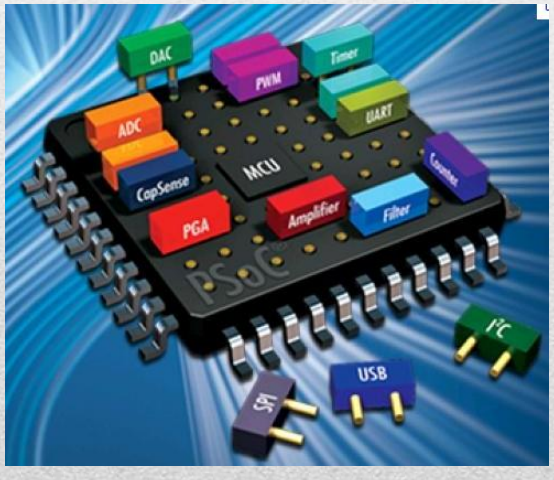
MICHIGAN STATE UNIVERSITY

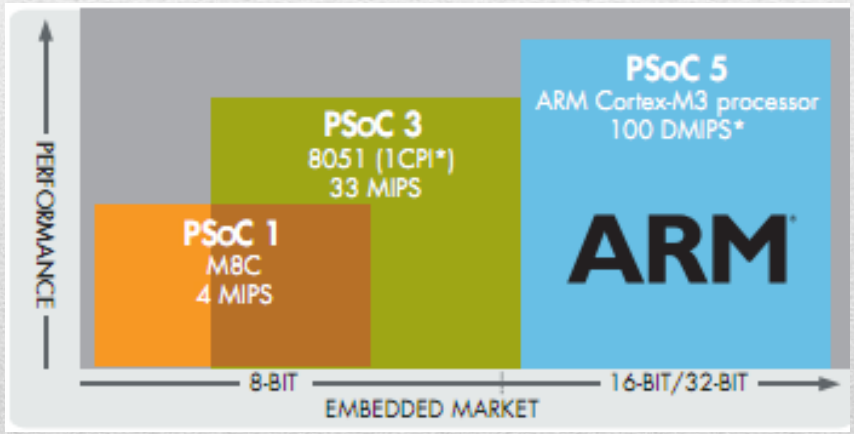
Outline

- Introduction
- Applications
- Hardware
- Software
  - PSoC Creator Overview
  - Digital Blocks
  - Analog Blocks
- Design in PSoC Creator
- Questions

Introduction

- Programmable System on Chip.
- It is the only programmable analog and digital embedded design platform.
- Contains a CPU and Programmable Hardware.
- It has subsystems in a single chip.
- Flexible and easy to integrate.



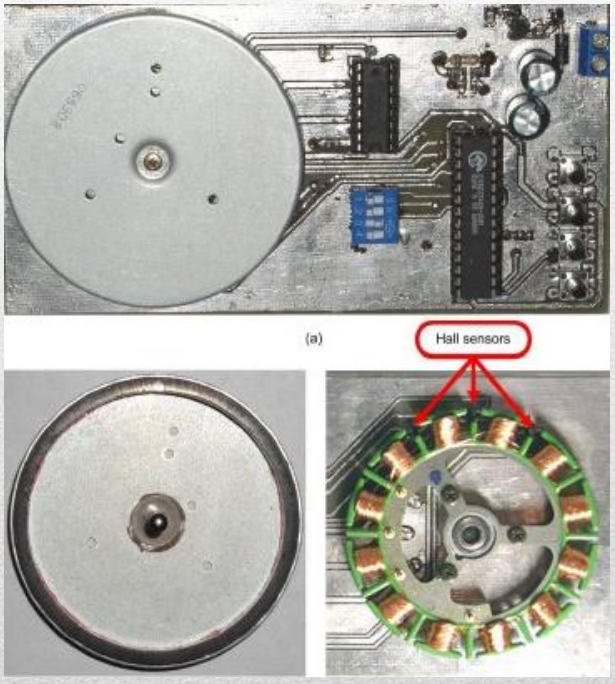


Applications

- General applications
  - Sensing (touch, light, proximity, temperature)
  - Motor control
  - Voltage monitoring and sequencing

Temp. sensor board

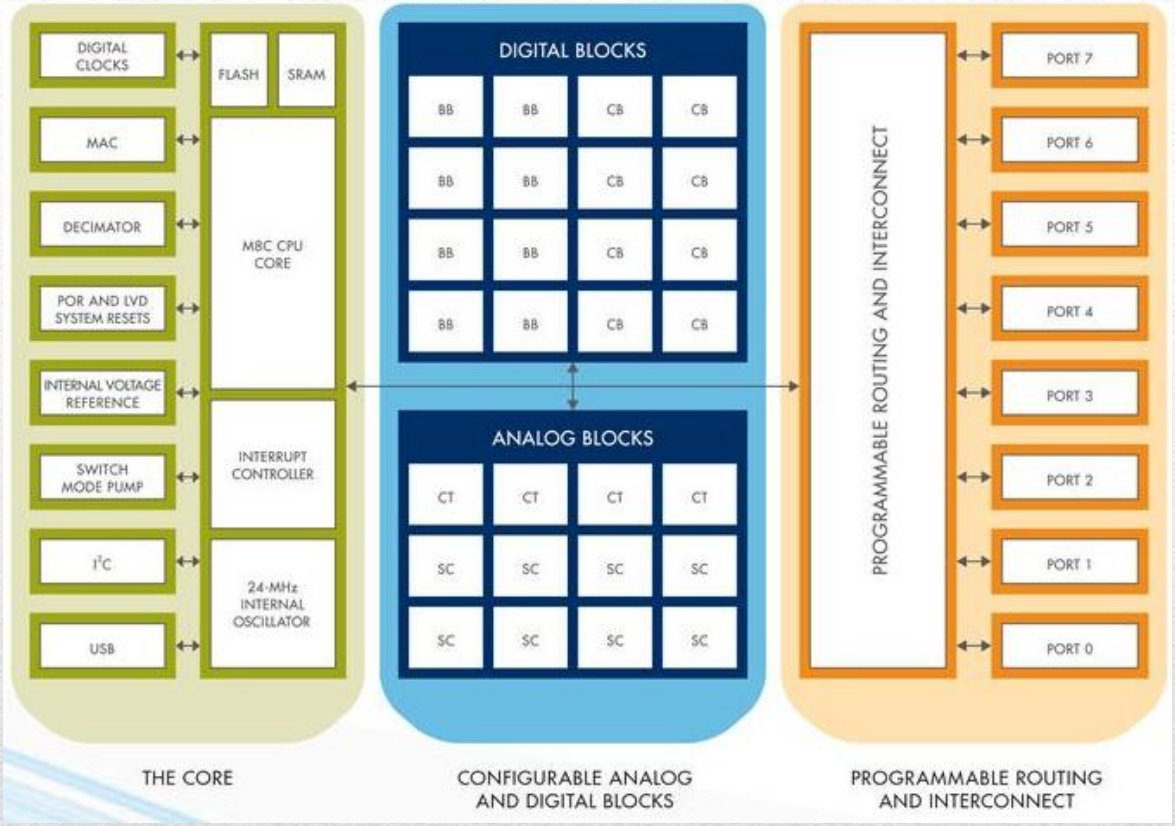
Cap-sensor



Brushless DC motor Drive

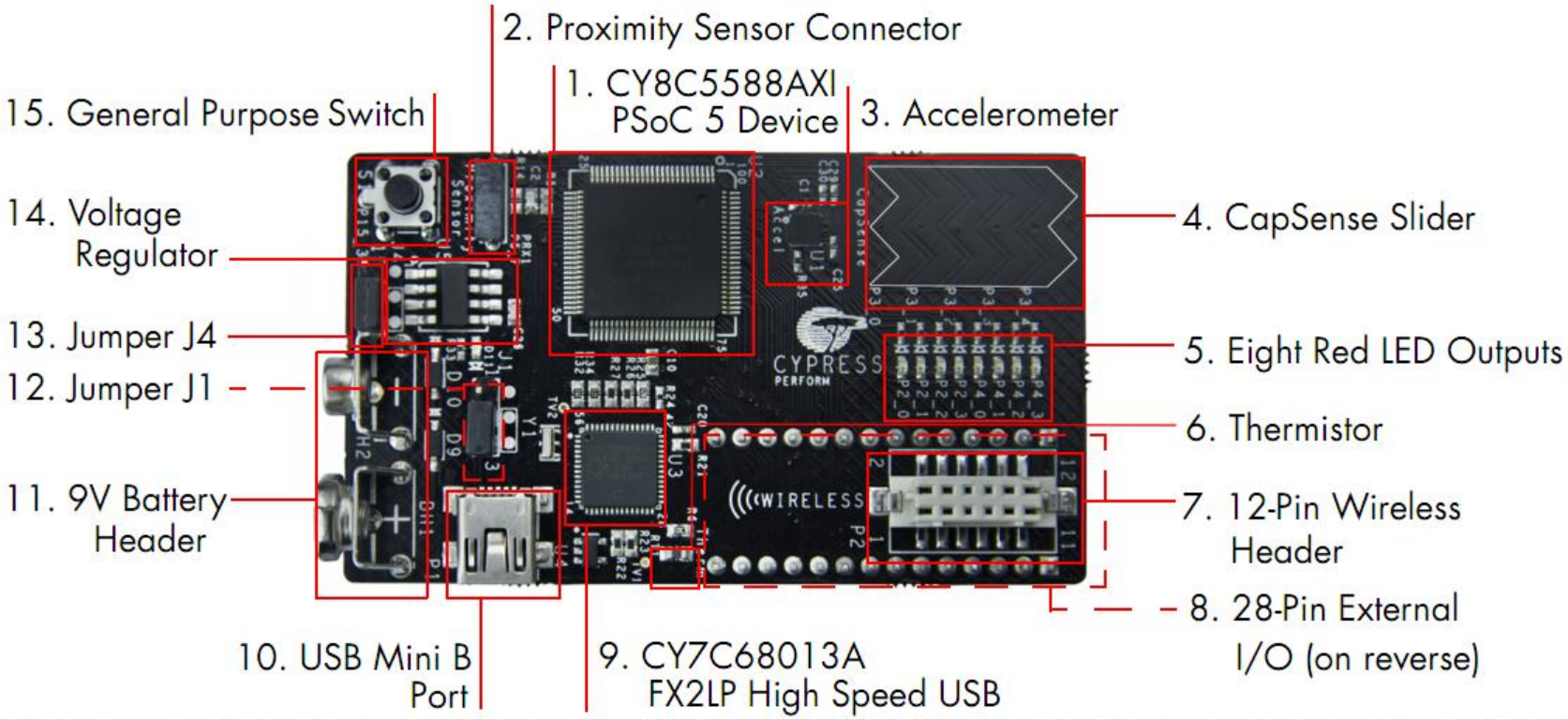
Applications

- Examples of applications
  - Portable medical devices such as blood pressure monitor and oximeter
  - Toothbrush
  - Adida running shoes
  - TiVo
  - Touch sensitive scroll wheel on iPod
  - Touch screen controller in NOOK color eReader
  - Washing machines



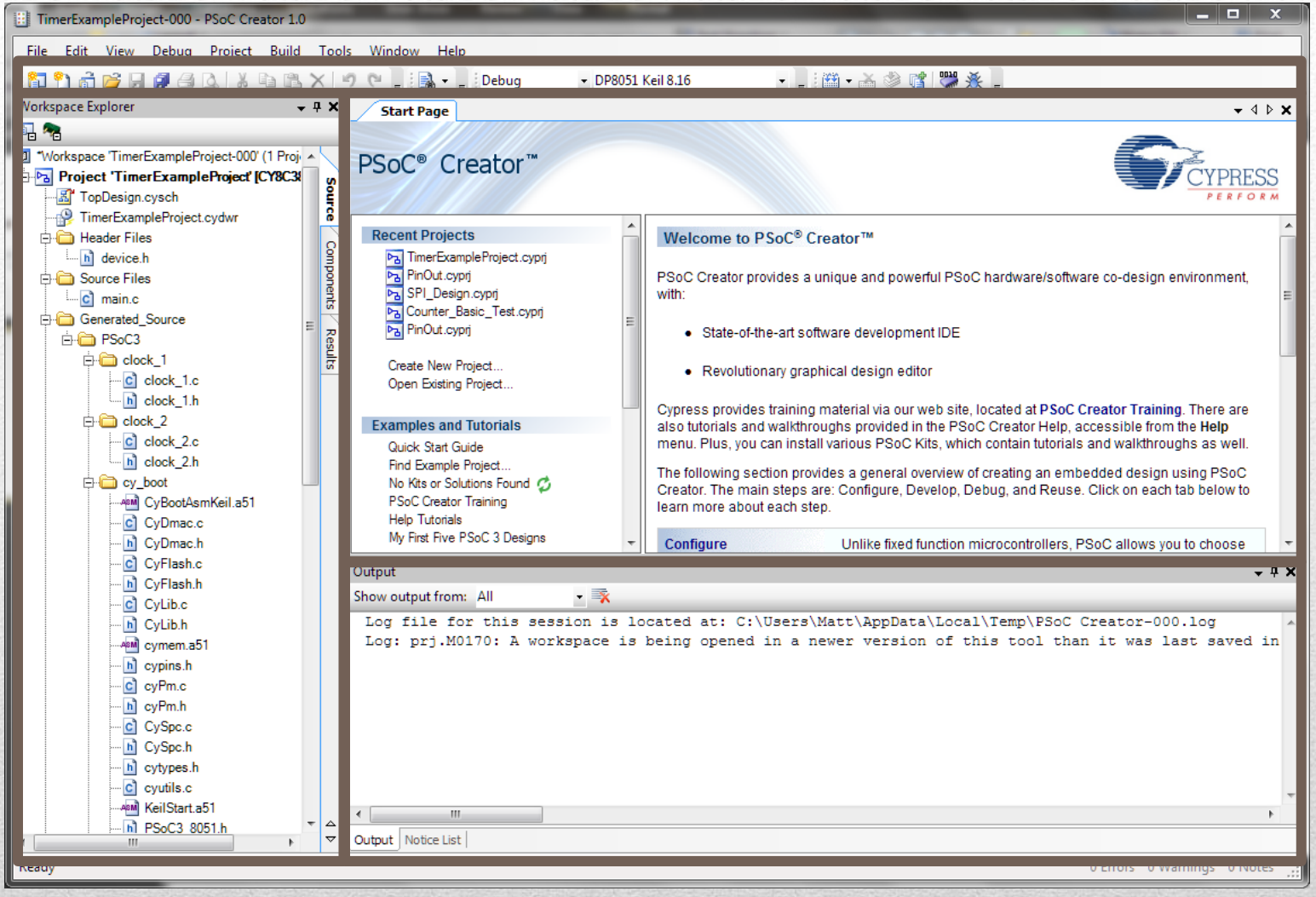
http://pdf.directindustry.com/pdf/cypress-semiconductor/cypress-psoc-programmable-system-on-chip-brochure/34220-70363-_8.html

PSoC Technology



http://www.cypress.com/?docID=27015

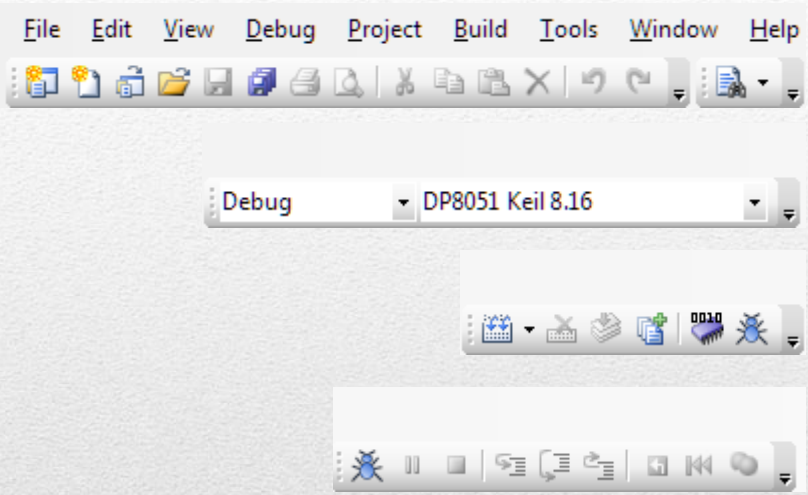
Hardware



Software: PSoC Creator Overview

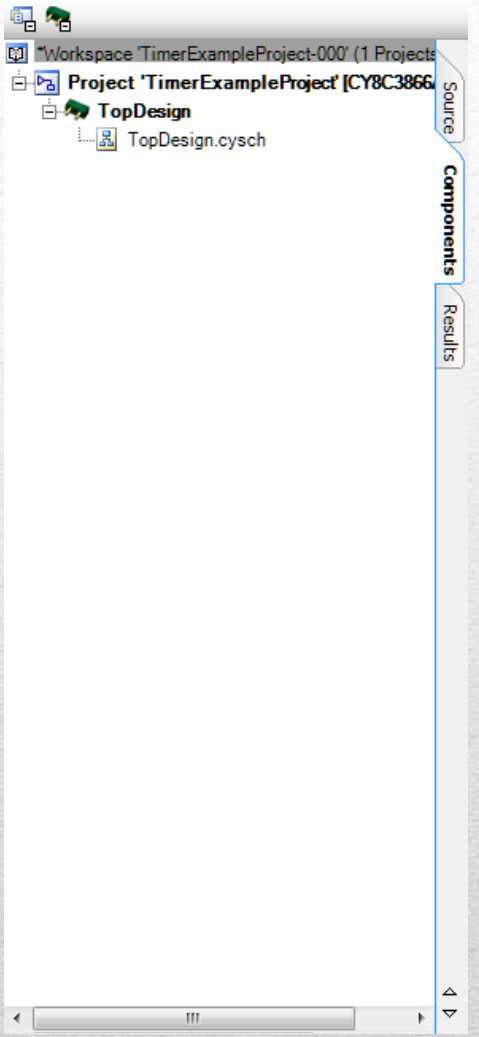
Toolbar

- Basic File Operation
- Compiler Options
- Build/Program
- Debug/Step
- Similar to Visual Studio

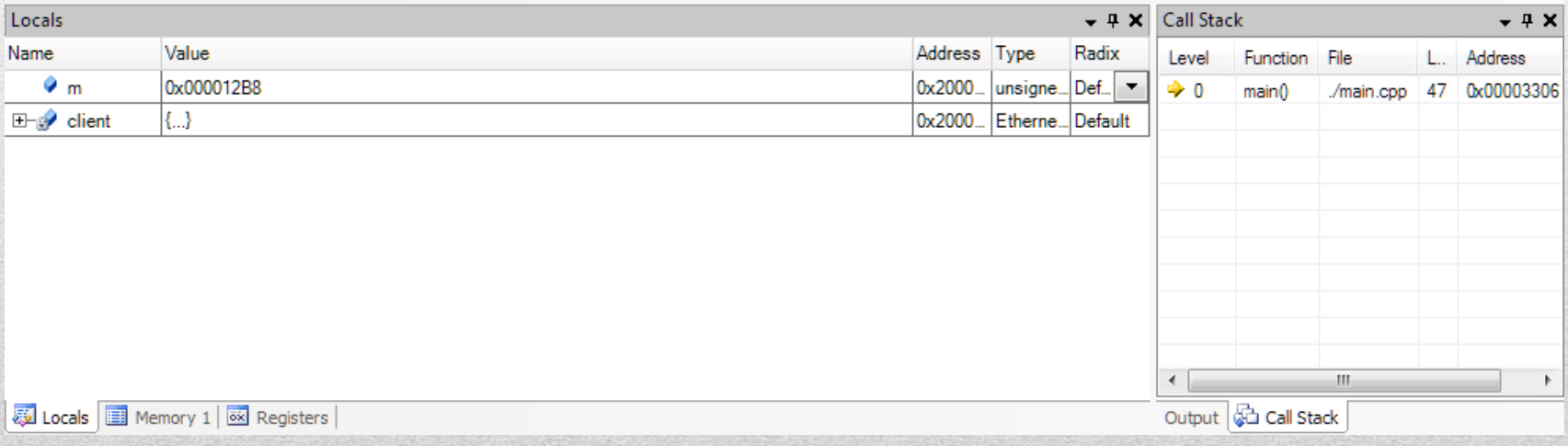


Workspace

- File Explorer
  - Organized into folders
  - Includes applications codes
  - Generated sources
  - Schematic file
- Component Tab

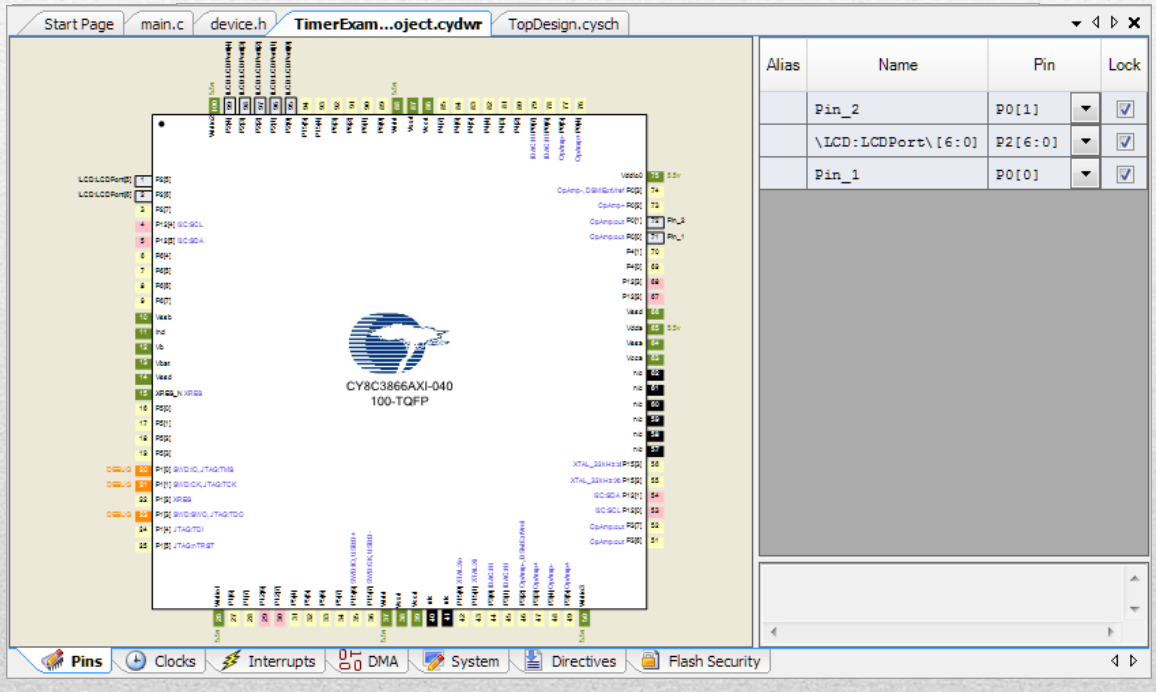


- Output of Build/Program
- Notice List includes Compiler/Linker errors
- Breakpoints for debugging
- Variable and memory values, stack



Output Window

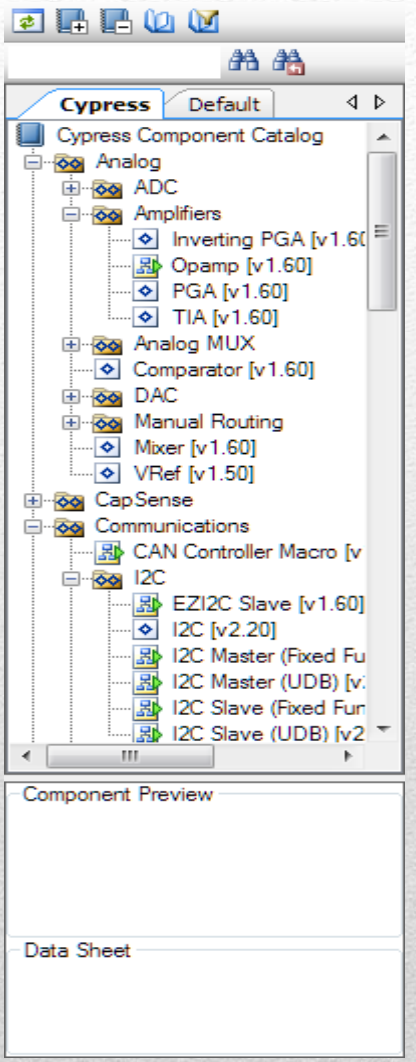
- C Source Files/Header Files
- Schematic Layout
- Pin assignment and configuration



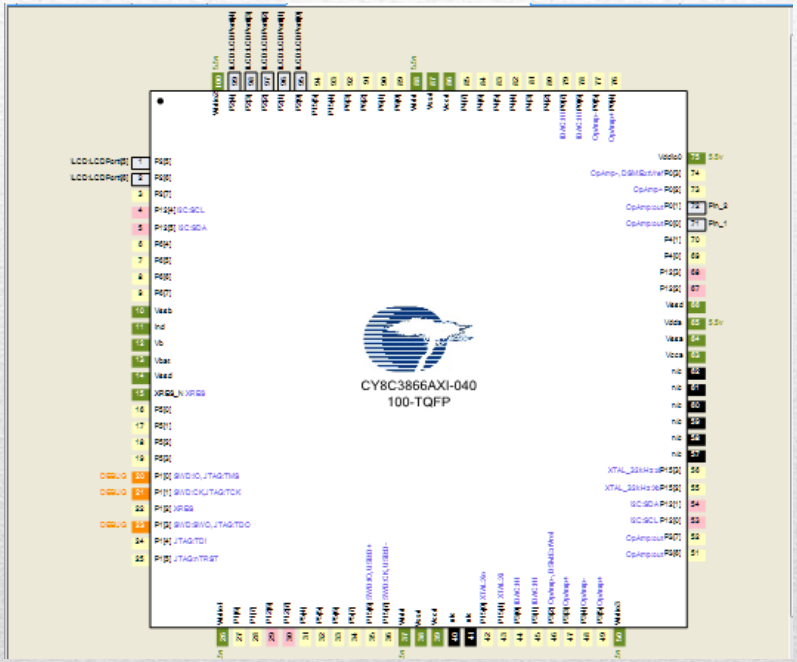
Document Pane

Schematic Layout

- Wires
- Annotations and Labels
- Components from library
- Generate symbols to add to library

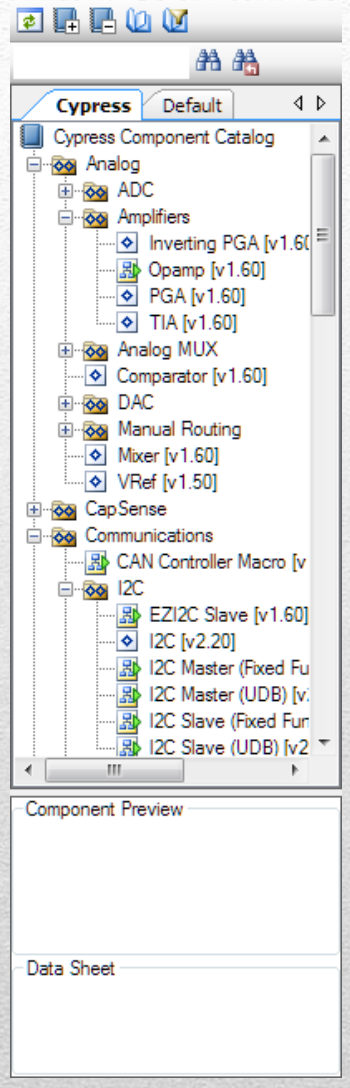


- Assign virtual pins to hardware pins
- Configure Clocks
- Configure Interrupts
- Configure DMA
- Other Configuration



Pin Assignment and Configuration

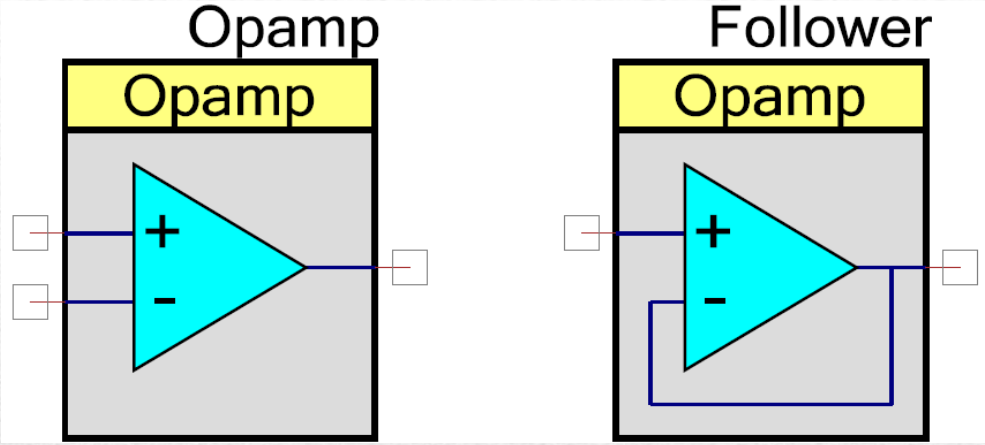
- A/D Converter
- D/A Converter
- Op-Amps
- MUXs
- Voltage and Current Sources



Analog Library

Analog OpAmps

- OpAmp or Voltage Follower
- 4 OpAmps in 1



	Non-inverting input	Inverting input	Output
opamp_0	P0[2]	P0[3]	P0[1]
opamp_1	P3[5]	P3[4]	P3[6]
opamp_2	P0[4]	P0[5]	P0[0]
opamp_3	P3[3]	P3[2]	P3[7]

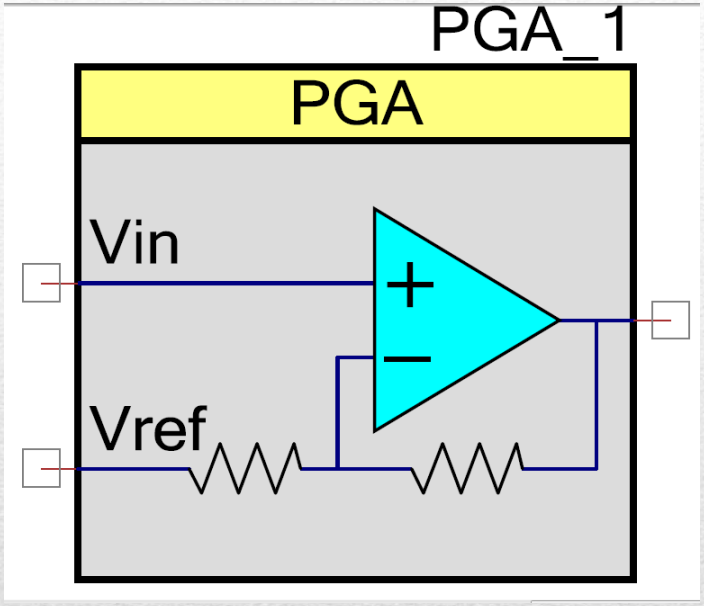
OpAmp Function

- Init
- Enable
- Start

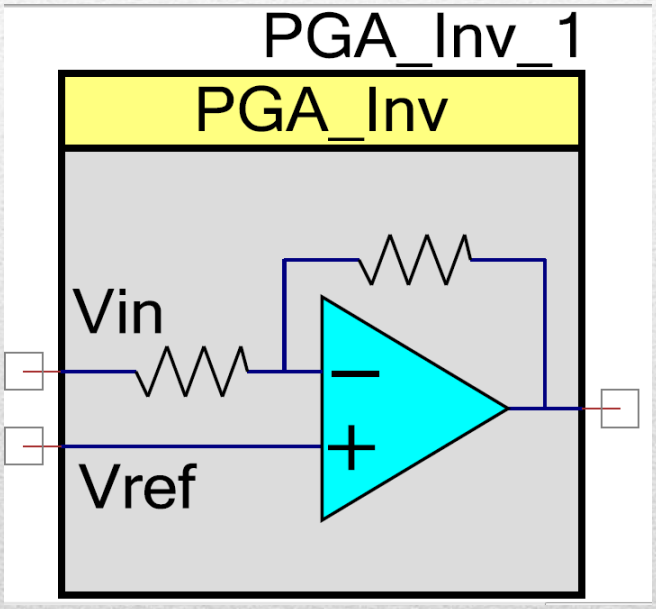
Function	Description
void Opamp_Start(void)	Turns on the Opamp and sets the power level to the value chosen during the parameter selection.
void Opamp_Stop(void)	Disable Opamp (power down)
void Opamp_SetPower(uint8 power)	Set the power level.
void Opamp_Sleep(void)	Stops and saves the user configuration.
void Opamp_Wakeup(void)	Restores and enables the user configuration.
void Opamp_SaveConfig(void)	Empty function. Provided for future usage.
void Opamp_RestoreConfig(void)	Empty function. Provided for future usage.
void Opamp_Init(void)	Initializes or restores default Opamp configuration.
void Opamp_Enable(void)	Enables the Opamp.

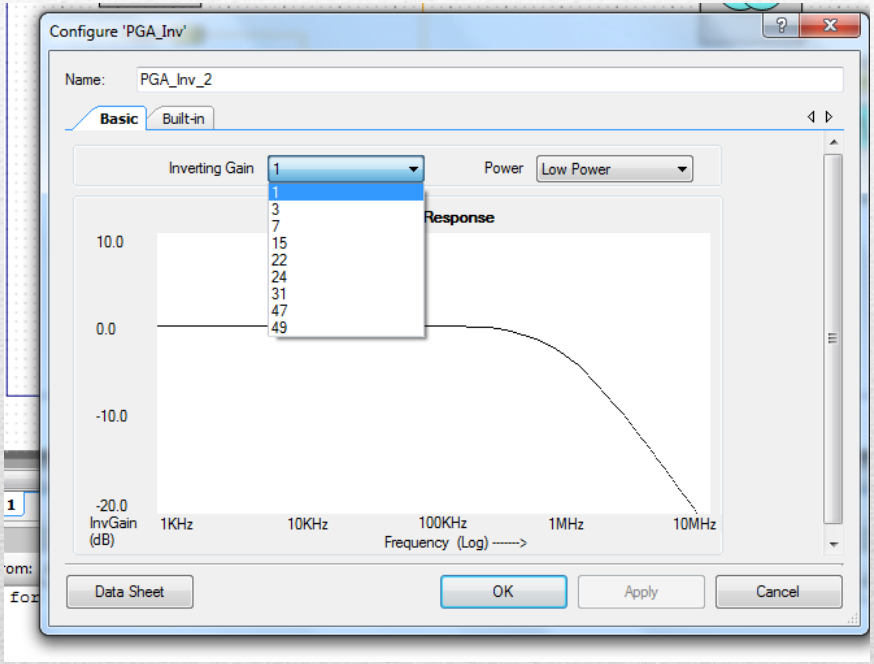
Programmable Gain Amp

- Gain between 1 and 50



- Gain between -1 and -49





Inverting PGA

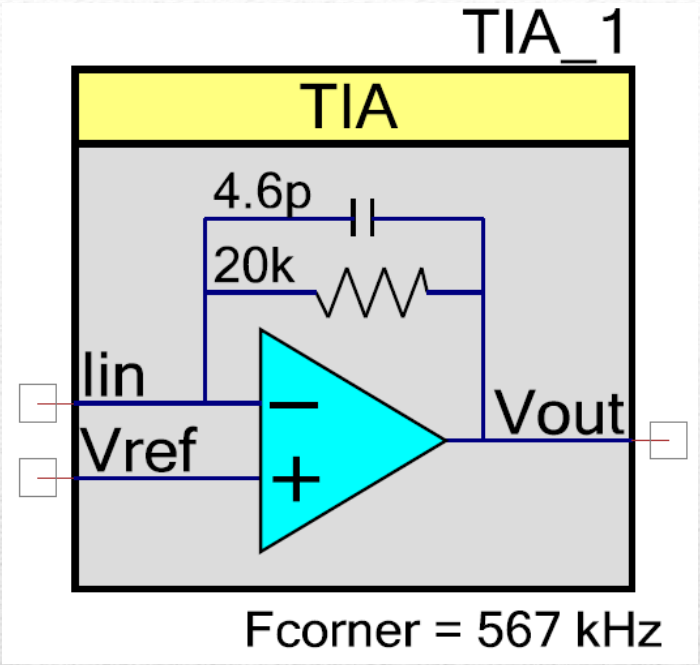
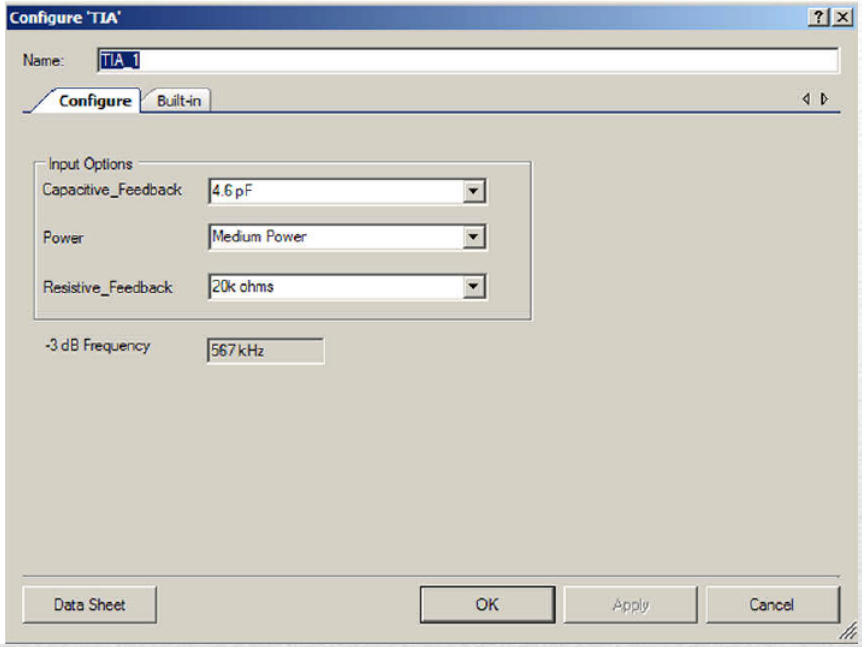


Figure 1. Configure TIA Dialog

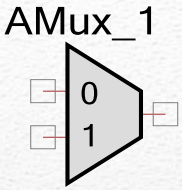


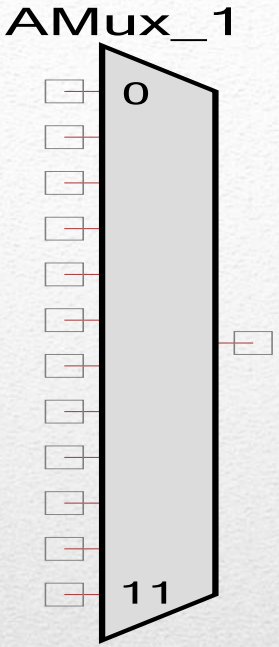
- Vout = Vref – Iin * Rfb

Trans-Impedance Amp

Analog MUX

- Between 2 and 32 inputs
- Single and Differential Inputs
- Software Controlled

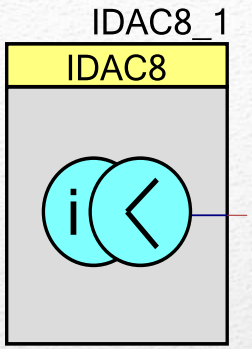


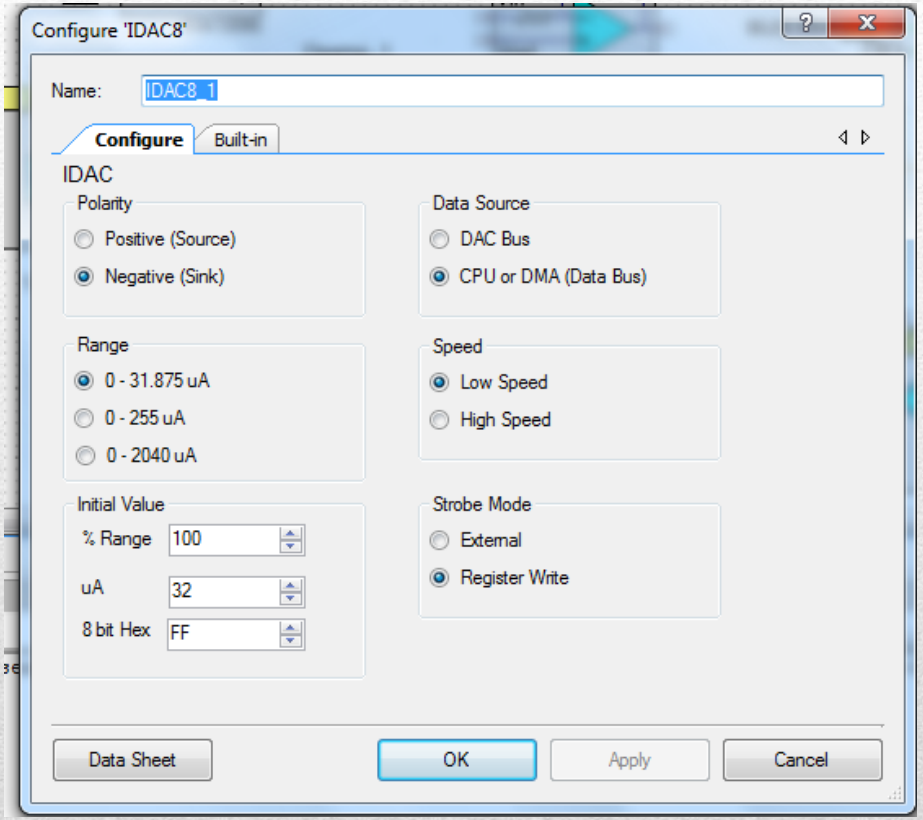


Function	Description
void AMux_Init(void)	Disconnect all channels
void AMux_Start(void)	Disconnect all channels
void AMux_Stop(void)	Disconnect all channels
void AMux_Select(uint8 chan)	Disconnect all channels, then connect "chan"
void AMux_Connect(uint8 chan)	Connect "chan" signal, but do not disconnect other channels.
void AMux_Disconnect(uint8 chan)	Disconnect only "chan" signal
void AMux_FastSelect(uint8 chan)	Disconnect the last channel that was selected by the AMux_Select() or AMux_FastSelect() function, then connect the new signal "chan".
void AMux_DisconnectAll(void)	Disconnect all channels

Analog Current DAC

- Current Source or Sink
- 3 Current Ranges





Analog Current DAC

- Start
- Enable
- Init

Function	Description
void IDAC8_Start(void)	Initialize the IDAC8 with default customizer values. Enable and power up the IDAC8.
void IDAC8_Stop(void)	Disables the IDAC8 and sets it to the lowest power state.
Void IDAC8_SetSpeed(uint8 speed)	Set DAC speed.
void IDAC8_SetPolarity(uint8 polarity)	Sets the output mode to current sink or source.
void IDAC8_SetRange(uint8 range)	Sets full scale range for IDAC8.
void IDAC8_SetValue(uint8 value)	Sets value between 0 and 255 with the given range.
void IDAC8_SaveConfig(void)	Empty function. Provided for future use
void IDAC8_RestoreConfig(void)	Empty function. Provided for future us.
void IDAC8_Sleep(void)	Stops and saves the user configuration.
void IDAC8_WakeUp(void)	Restores and enables the user configuration.
void IDAC8_Init(void)	Initializes or restores default IDAC8 configuration
void IDAC8_Enable(void)	Enables the IDAC8.

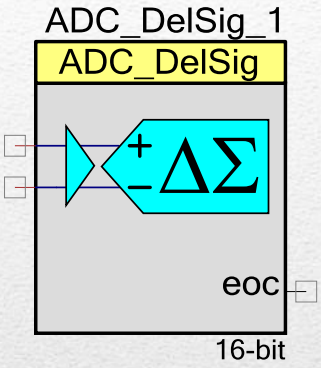
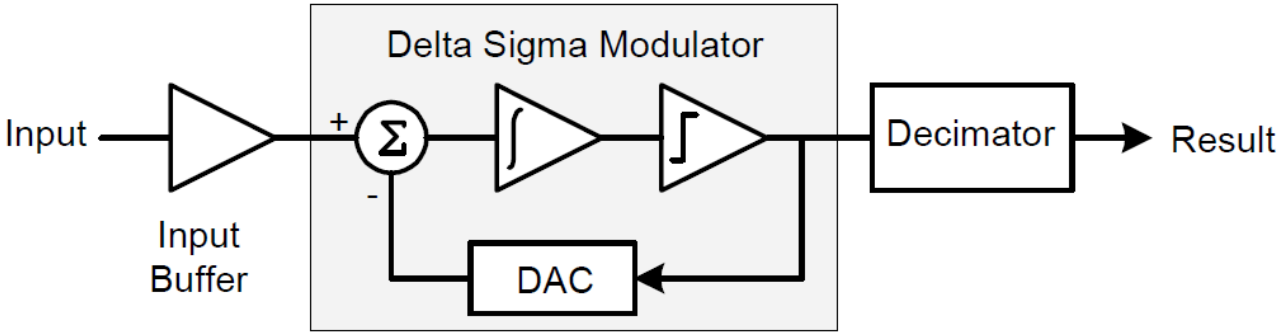
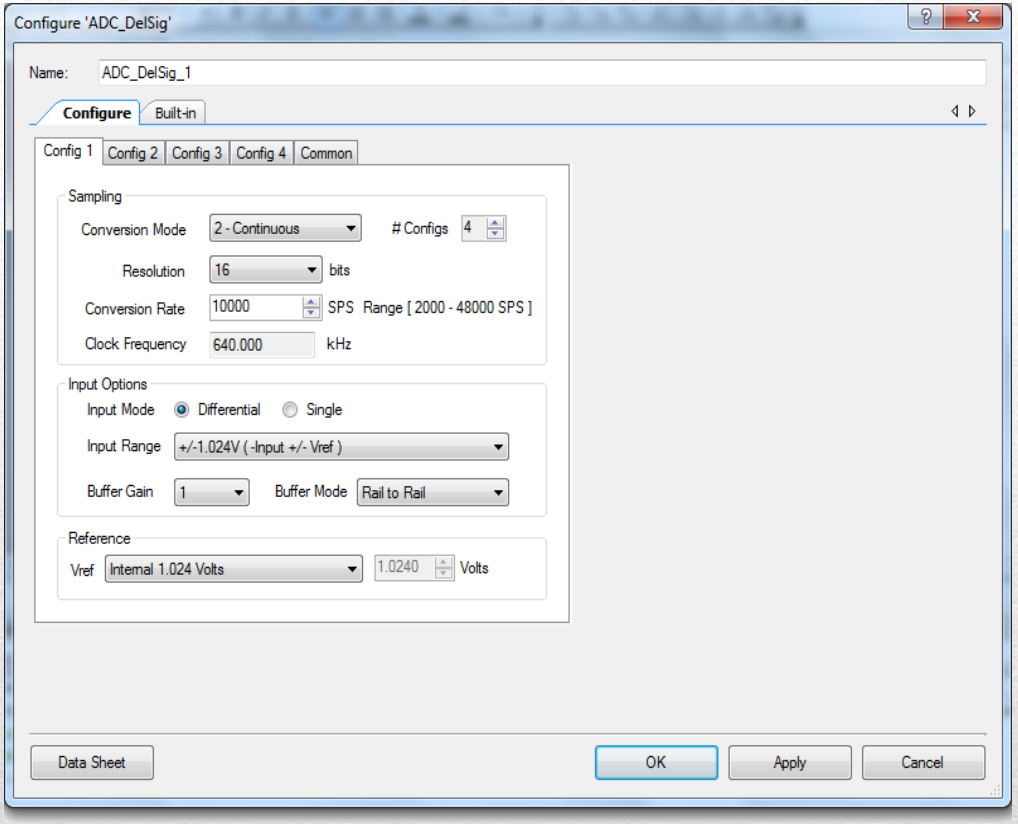


Figure 1: ADC_DelSig Block Diagram

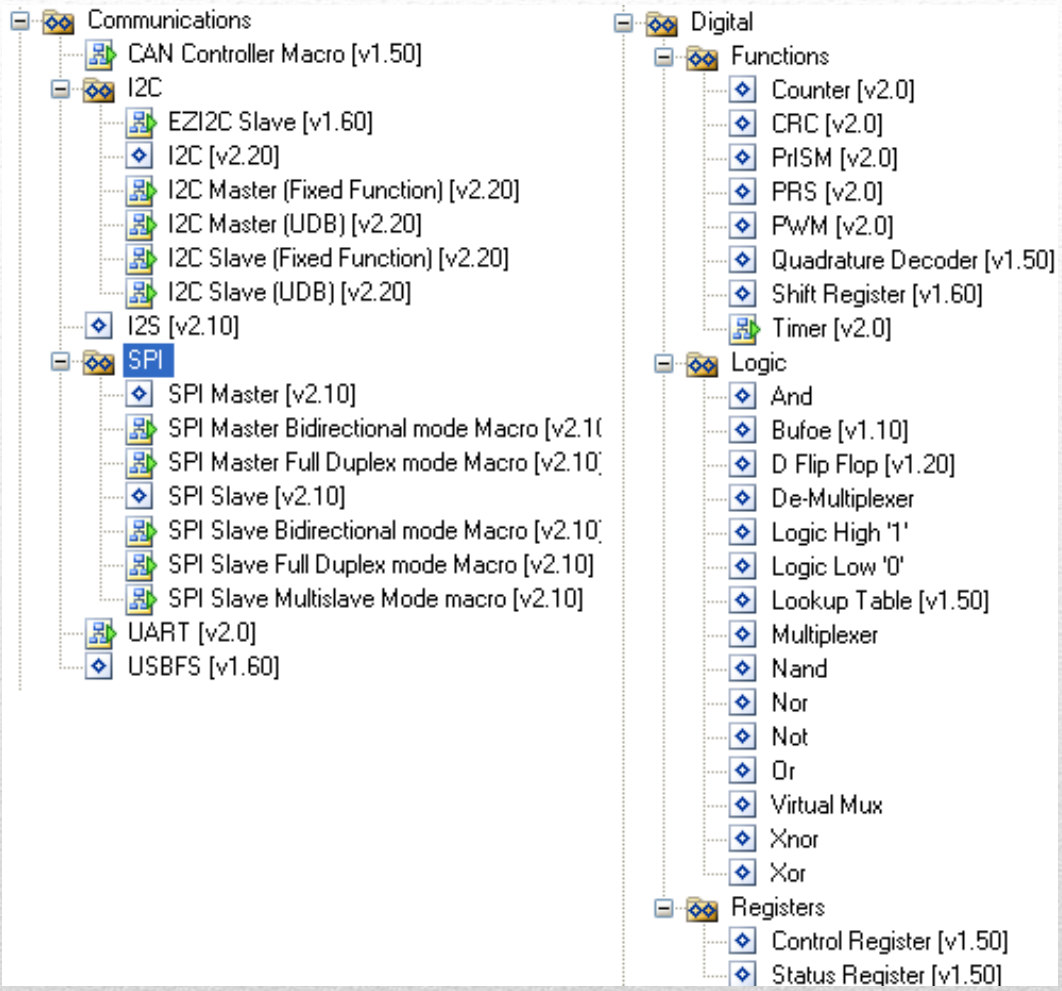


Delta Sigma ADC

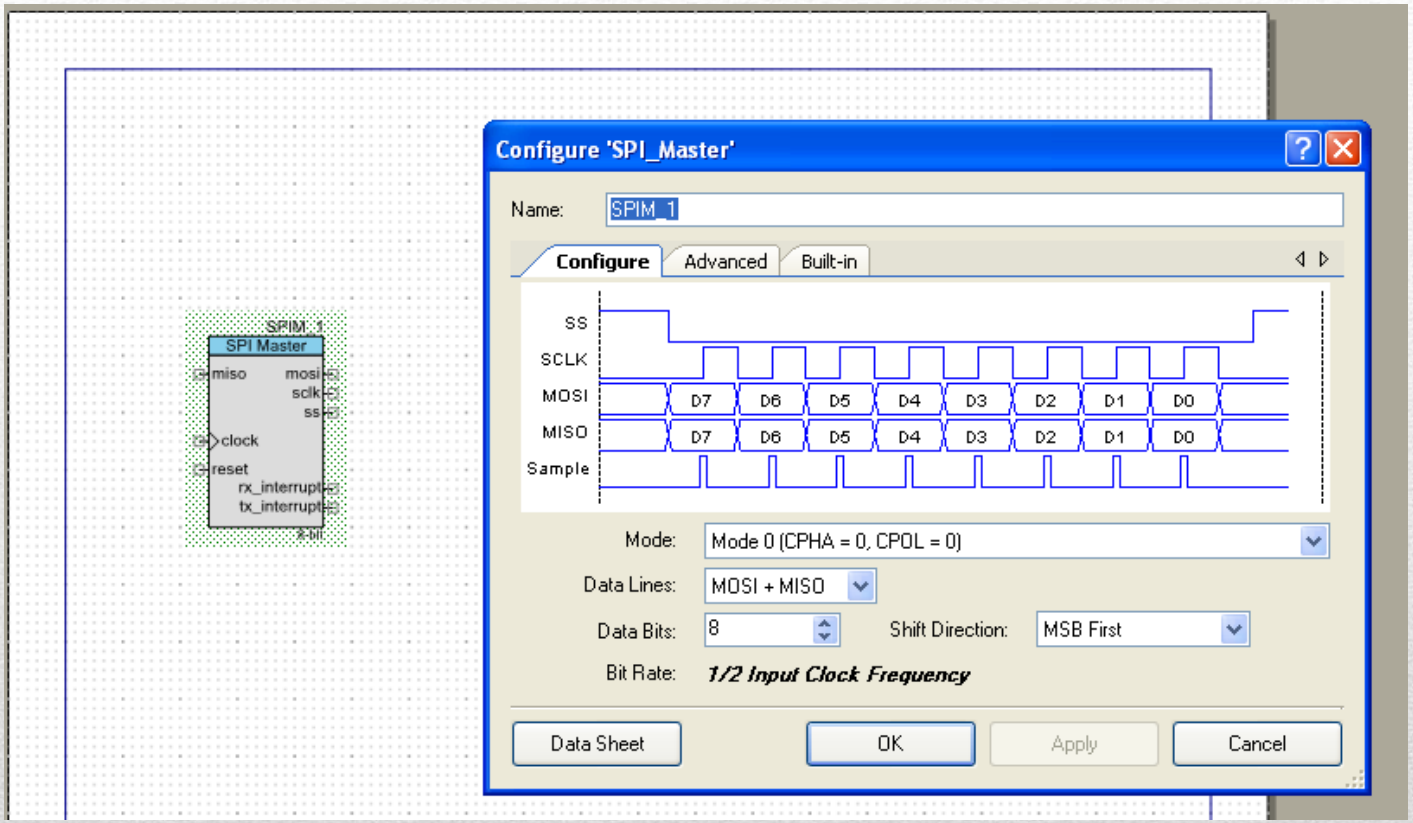
- Four Different Configurations



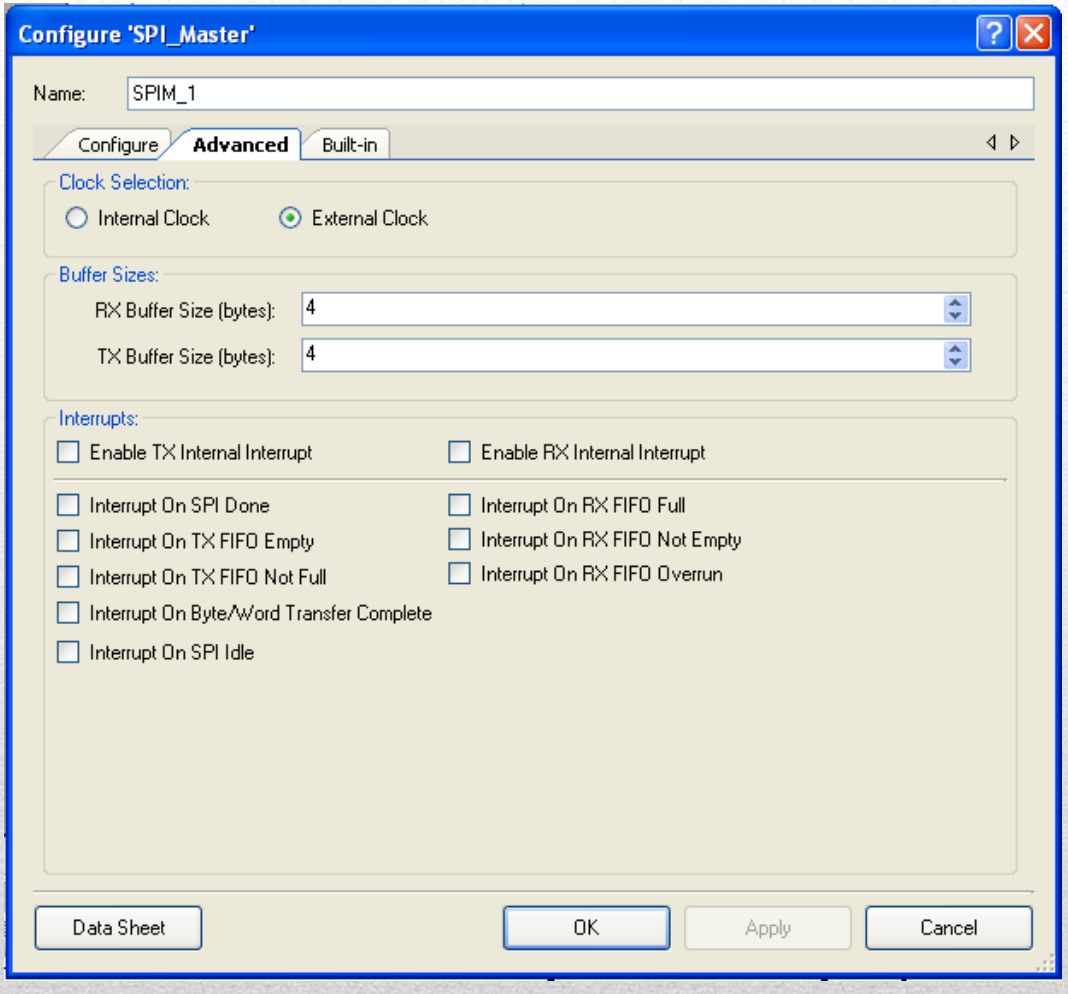
Delta Sigma ADC



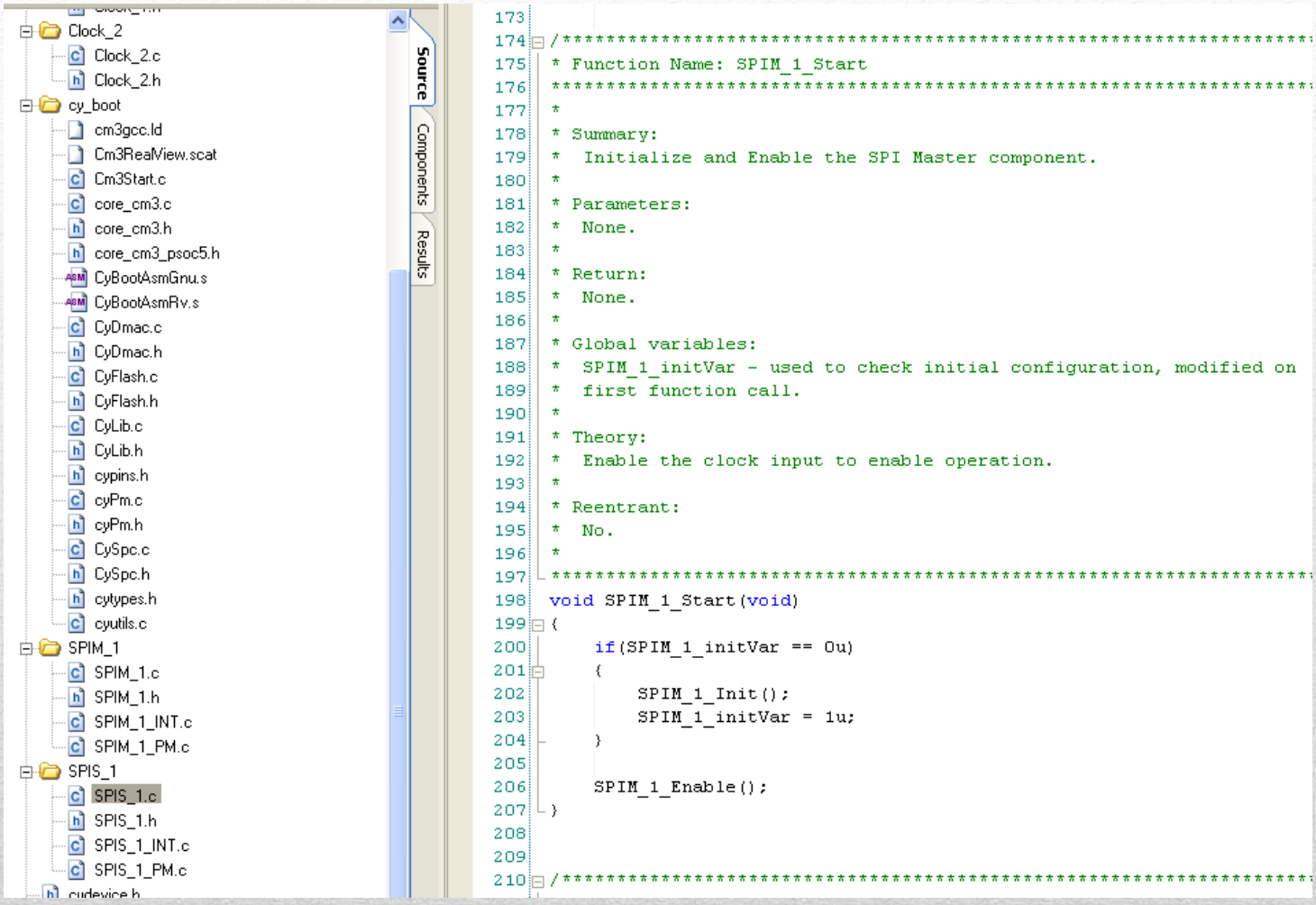
Communication and Digital Blocks

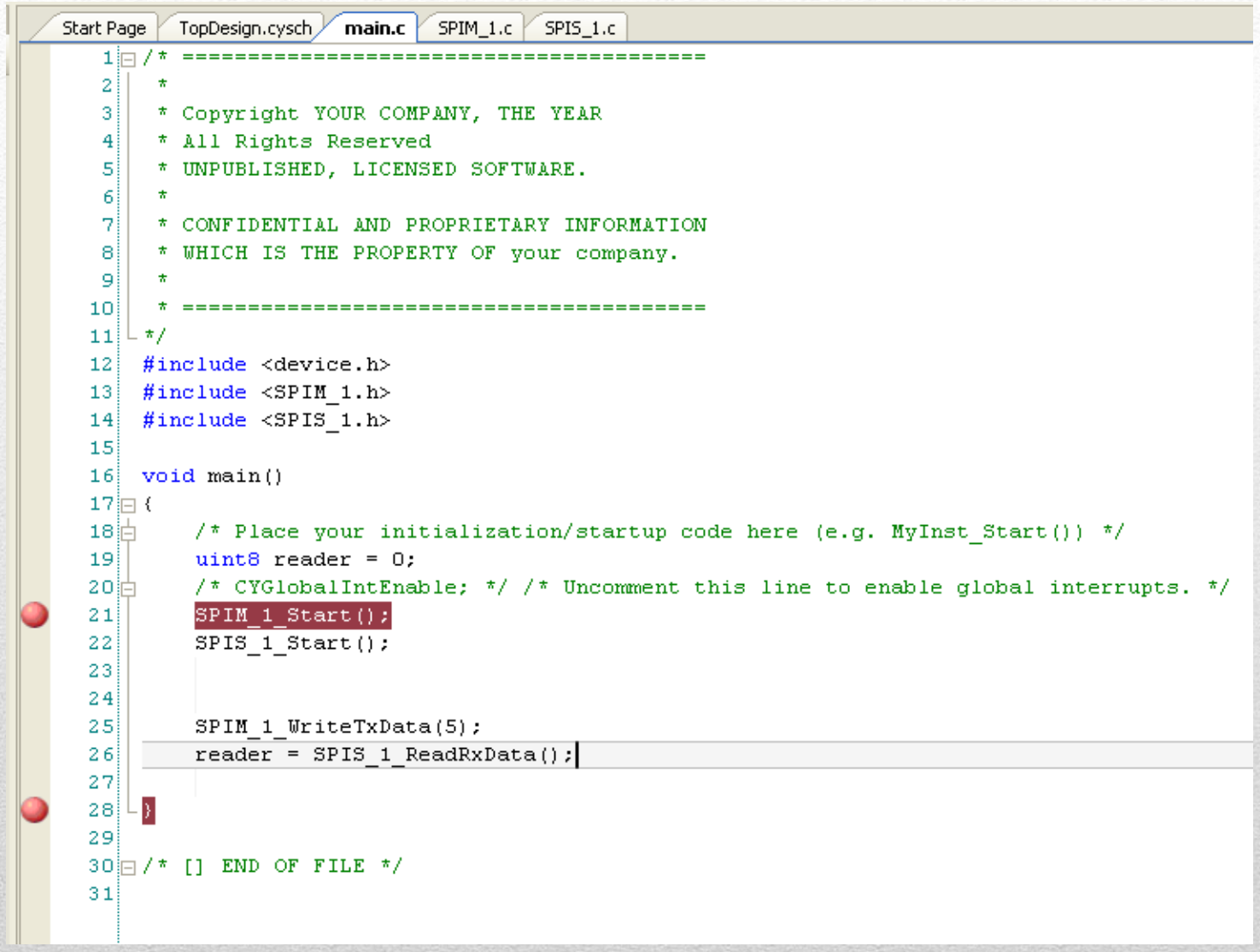


SPI: Example of Communication Block


Configure 'SPI_Master'
Name:
SPIM_1
Configure
Advanced
Built-in
Clock Selection:
Internal Clock
External Clock
Buffer Sizes:
RX Buffer Size (bytes):
4
TX Buffer Size (bytes):
4
Interrupts:
Enable TX Internal Interrupt
Enable RX Internal Interrupt
Interrupt On SPI Done
Interrupt On TX FIFO Empty
Interrupt On TX FIFO Not Full
Interrupt On Byte/Word Transfer Complete
Interrupt On SPI Idle
Interrupt On RX FIFO Full
Interrupt On RX FIFO Not Empty
Interrupt On RX FIFO Overrun
Data Sheet
OK
Apply
Cancel

SPI Advance


Clock_2
Clock_2.c
Clock_2.h
cy_boot
cm3gcc.ld
Cm3RealView.scat
Cm3Start.c
core_cm3.c
core_cm3.h
core_cm3_psoc5.h
CyBootAsmGnu.s
CyBootAsmRv.s
CyDmac.c
CyDmac.h
CyFlash.c
CyFlash.h
CyLib.c
CyLib.h
cypins.h
cyPm.c
cyPm.h
CySpc.c
CySpc.h
cytypes.h
cyutils.c
SPIM_1
SPIM_1.c
SPIM_1.h
SPIM_1_INT.c
SPIM_1_PM.c
SPIS_1
SPIS_1.c
SPIS_1.h
SPIS_1_INT.c
SPIS_1_PM.c
cydevice.h
Source
Components
Results
173
174 /***
175 * Function Name: SPIM_1_Start
176 **
177 *
178 * Summary:
179 * Initialize and Enable the SPI Master component.
180 *
181 * Parameters:
182 * None.
183 *
184 * Return:
185 * None.
186 *
187 * Global variables:
188 * SPIM_1_initVar - used to check initial configuration, modified on
189 * first function call.
190 *
191 * Theory:
192 * Enable the clock input to enable operation.
193 *
194 * Reentrant:
195 * No.
196 *
197 ***
198 void SPIM_1_Start(void)
199 {
200 if(SPIM_1_initVar == 0u)
201 {
202 SPIM_1_Init();
203 SPIM_1_initVar = 1u;
204 }
205
206 SPIM_1_Enable();
207 }
208
209
210 /***


Start Page
TopDesign.cysch
main.c
SPIM_1.c
SPIS_1.c
/* ==
*
* Copyright YOUR COMPANY, THE YEAR
* All Rights Reserved
* UNPUBLISHED, LICENSED SOFTWARE.
*
* CONFIDENTIAL AND PROPRIETARY INFORMATION
* WHICH IS THE PROPERTY OF your company.
*
* ==
*/
#include <device.h>
#include <SPIM_1.h>
#include <SPIS_1.h>
void main()
{
/* Place your initialization/startup code here (e.g. MyInst_Start()) */
uint8 reader = 0;
/* CYGlobalIntEnable; */ /* Uncomment this line to enable global interrupts. */
SPIM_1_Start();
SPIS_1_Start();
SPIM_1_WriteTxData(5);
reader = SPIS_1_ReadRxData();
}
/* [] END OF FILE */

Design in PSoC Creator

The main steps for creating embedded design in PSoC Creator are:

1) Configure
2) Develop
3) Debug
4) Reuse

Design in PSoC Creator

- **CONFIGURE –** Choose the on-chip peripherals, drag onto schematic, set the parameters (e.g. duty cycle of PWM, power and gain of amplifier). Datasheets available for components.

- **DEVELOP –** C based development flow with automatically generated software APIs. Consistently named, reduce coding errors, and ensure correct interaction with peripheral.

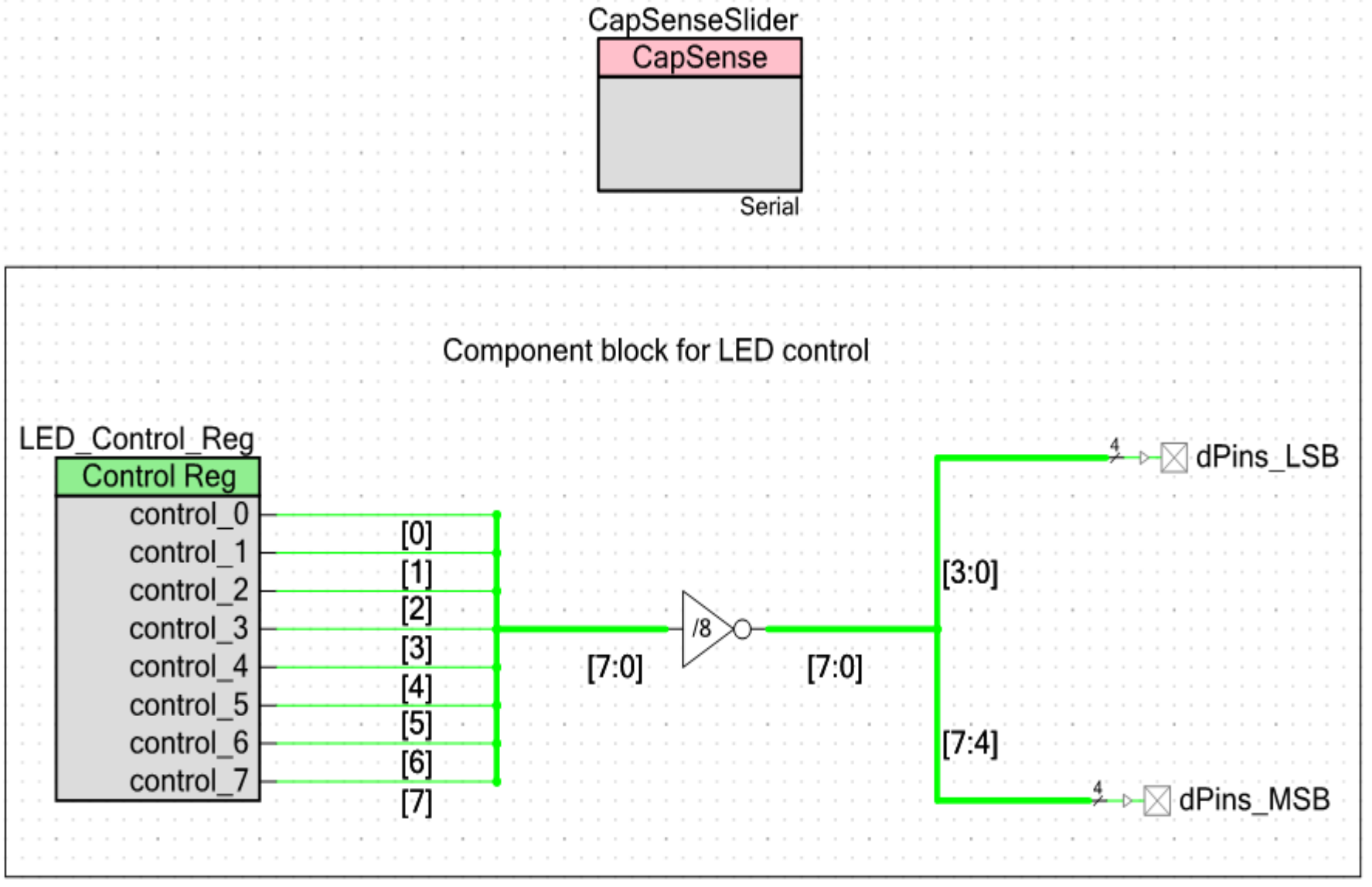
Design in PSoC Creator

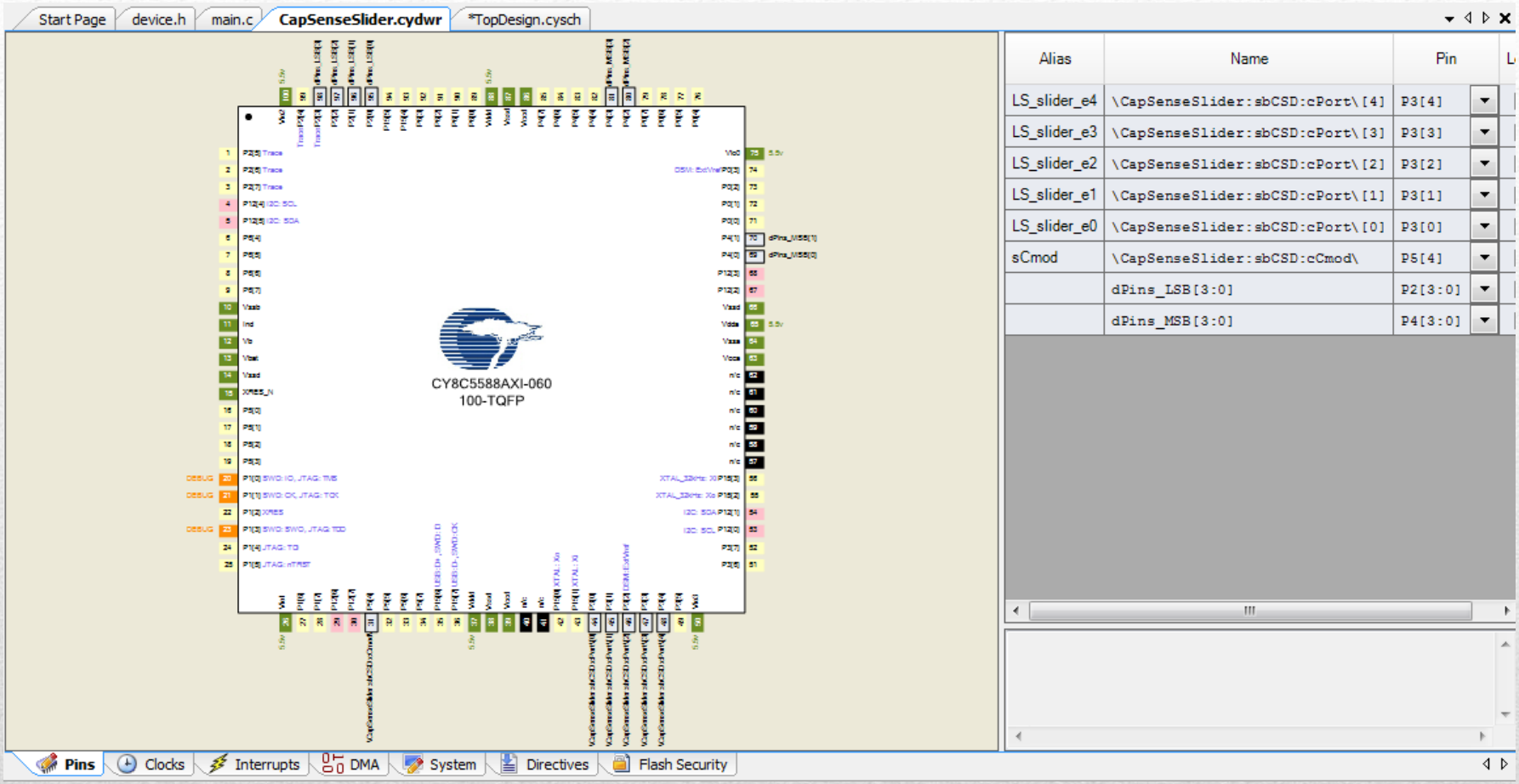
- **DEBUG –** Has, in addition to features of a standard debugger, a peripheral debug window with status of internals of the on-chip components. C, disassembly windows, registers, memory, call stack windows included as well. MiniProg3 provides host-to-device connectivity, which connects PC's USB port to device JTAG interface.

- **REUSE –** Working design can be made into reusable component. A symbol is generated for the design. Once the component is saved into a library, it can be reused.

- Detects position of finger on CapSense slider of PSoC 5 First Touch kit board and indicates position using LEDs
- Bank of capacitive sensors form a slider
- Detects presence of finger by a change in capacitive value
- CapSense provides APIs that report the relative position of the finger on the slider
- Firmware lights the corresponding LEDs

Example: CapSense Slider


CapSenseSlider
CapSense
Serial
Component block for LED control
LED_Control_Reg
Control Reg
control_0
control_1
control_2
control_3
control_4
control_5
control_6
control_7
[0]
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[7:0]
/8
[7:0]
[3:0]
[7:4]
4
dPins_LSB
4
dPins_MSB



Pin Out for CapSense Slider

Alias	Name	Pin
LS_slider_e4	\CapSenseSlider:sbCSD:cPort\[4]	P3[4]
LS_slider_e3	\CapSenseSlider:sbCSD:cPort\[3]	P3[3]
LS_slider_e2	\CapSenseSlider:sbCSD:cPort\[2]	P3[2]
LS_slider_e1	\CapSenseSlider:sbCSD:cPort\[1]	P3[1]
LS_slider_e0	\CapSenseSlider:sbCSD:cPort\[0]	P3[0]
sCmod	\CapSenseSlider:sbCSD:cCmod\	P5[4]
	dPins_LSB[3:0]	P2[3:0]
	dPins_MSB[3:0]	P4[3:0]

Pin Assignment

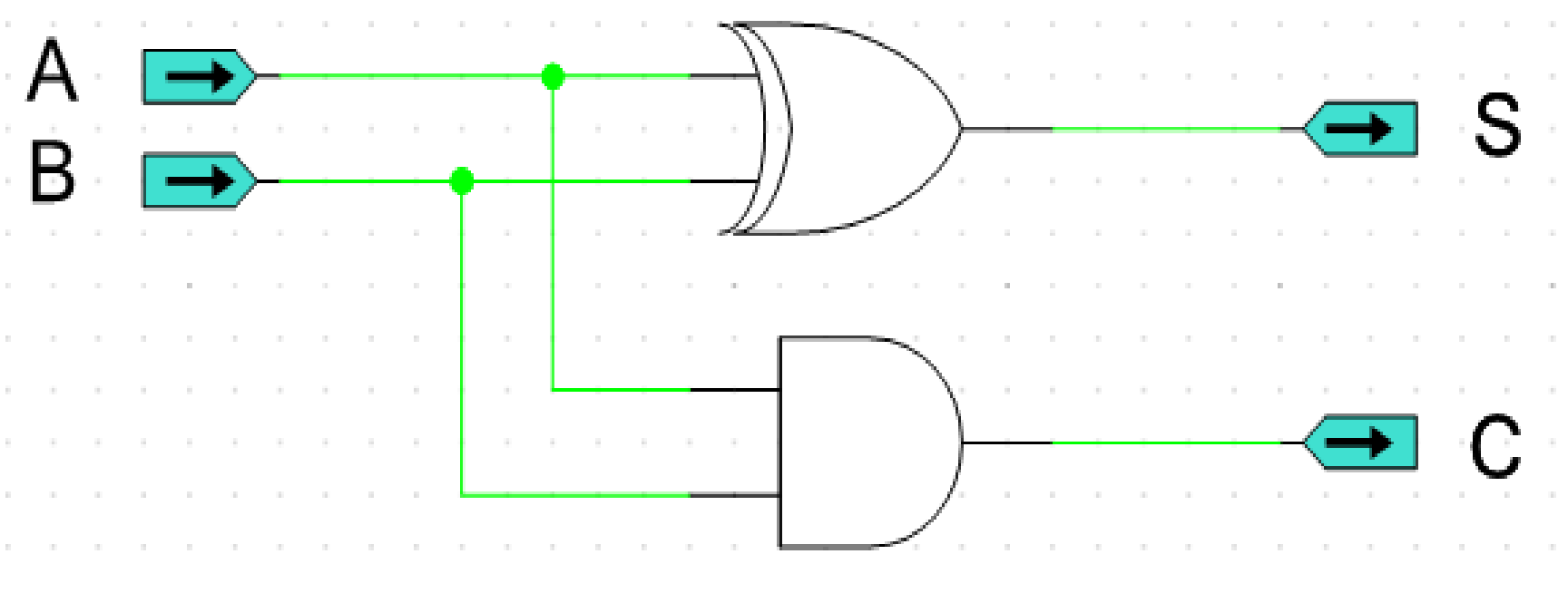
44	P3[0]	CapSense slider element 1
45	P3[1]	CapSense slider element 2
46	P3[2]	CapSense slider element 3
47	P3[3]	CapSense slider element 4
48	P3[4]	CapSense slider element 5
31	P5[4]	CapSense Modulator capacitor

95	P2[0]	LED 1 drive
96	P2[1]	LED 2 drive
97	P2[2]	LED 3 drive
98	P2[3]	LED 4 drive
69	P4[0]	LED 5 drive
70	P4[1]	LED 6 drive
80	P4[2]	LED 7 drive
81	P4[3]	LED 8 drive

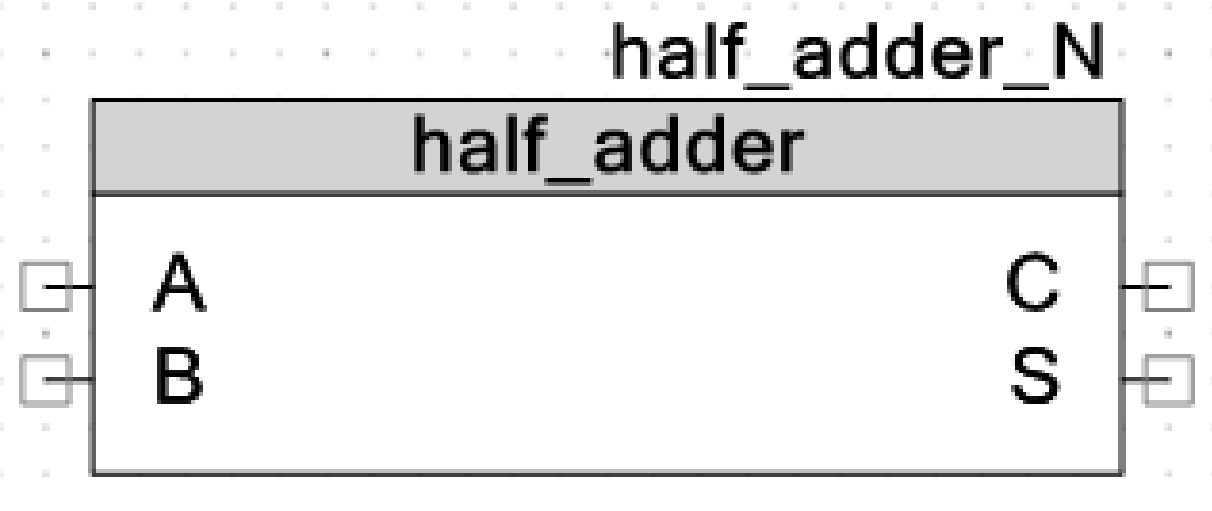
```c
#include <device.h>

#define NUM_LED (3)      // Constant to convert the Centroid position to a range of 0x01 - 0x08

void main()
{
    uint8 CentroidPosition=0xFF;

    uint8 LedData=0;

    /* Enable global interrupt */
    CYGlobalIntEnable;

    /* Turn off all LEDs on power on*/
    LED_Control_Reg_Write(LedData);

    /* Start and initialize CapSense */
    CapSenseSlider_Start();
    CapSenseSlider_CSHL_InitializeAllBaselines();

    while(1)
    {
        /* Scan and update Capsense slider sensor */
        CapSenseSlider_CSD_ScanAllSlots();
        CapSenseSlider_CSHL_UpdateAllBaselines();

        /* Get Centroid position of the finger on the slider */
        CentroidPosition = (uint8)CapSenseSlider_CSHL_GetCentroidPos(CapSenseSlider_CSHL_LS_SLIDER);

        /* If a finger is detected on the slider then turn on the associated LED*/
        LedData = 0;
        if(CentroidPosition != 0xFF)
        {
            /* Find the finger position on slider based on 8 LEDs of the total resolution of 64 counts */
            LedData = 1 << (CentroidPosition >> NUM_LED);
        }

        /* Write to the LED control register and update LED status*/
        LED_Control_Reg_Write(LedData);
    }
}
```

Reuse

- Reuse pertains to creation of components and placement of these components in a Library
- Working designs can be grouped as a component for reuse in later projects
- Symbol representation replaces full schematic representation
- Saves time and physical space thereby reducing overall cost
- Eg. A full PCB layout could potentially be saved as a component in PSoC Creator



Example: Half Adder



- Half adder schematic is now represented as a symbol
- Can be reused without the need to repeat schematic layout

Half Adder Symbol

Questions