PSoC Technology

Team 1
Cecilia Acosta
Brett Donlon
Matt Durak
Aaron Thomson
Nathan Ward
• Introduction
• Applications
• Hardware
• Software
  • PSoC Creator Overview
  • Digital Blocks
  • Analog Blocks
• Design in PSoC Creator
• Questions
• 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.
• General applications
  • Sensing (touch, light, proximity, temperature)
  • Motor control
  • Voltage monitoring and sequencing

Applications

Temp. sensor board  Cap-sensor  Brushless DC motor Drive
• 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
Hardware

http://www.cypress.com/?docID=27015
Software: PSoC Creator Overview
• Basic File Operation
• Compiler Options
• Build/Program
• Debug/Step
• Similar to Visual Studio

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

Workspace
• Output of Build/Program
• Notice List includes Compiler/Linker errors
• Breakpoints for debugging
• Variable and memory values, stack
• C Source Files/Header Files
• Schematic Layout
• Pin assignment and configuration
• Wires
• Annotations and Labels
• Components from library
• Generate symbols to add to library

Schematic Layout
• Assign virtual pins to hardware pins
• Configure Clocks
• Configure Interrupts
• Configure DMA
• Other Configuration
- A/D Converter
- D/A Converter
- Op-Amps
- MUXs
- Voltage and Current Sources
• OpAmp or Voltage Follower
• 4 OpAmps in 1

<table>
<thead>
<tr>
<th></th>
<th>Non-inverting input</th>
<th>Inverting input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>opamp_0</td>
<td>P0[2]</td>
<td>P0[3]</td>
<td>P0[1]</td>
</tr>
<tr>
<td>opamp_2</td>
<td>P0[4]</td>
<td>P0[5]</td>
<td>P0[0]</td>
</tr>
</tbody>
</table>
OpAmp Function

- Init
- Enable
- Start

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

- Gain between 1 and 50
- Gain between -1 and -49
Trans-Impedance Amp

- \( \text{Vout} = \text{Vref} - \text{Iin} \times \text{Rfb} \)
Analog MUX

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void AMux_Init(void)</td>
<td>Disconnect all channels</td>
</tr>
<tr>
<td>void AMux_Start(void)</td>
<td>Disconnect all channels</td>
</tr>
<tr>
<td>void AMux_Stop(void)</td>
<td>Disconnect all channels</td>
</tr>
<tr>
<td>void AMux&gt;Select(uint8 chan)</td>
<td>Disconnect all channels, then connect &quot;chan&quot;</td>
</tr>
<tr>
<td>void AMux_Connect(uint8 chan)</td>
<td>Connect &quot;chan&quot; signal, but do not disconnect other channels.</td>
</tr>
<tr>
<td>void AMux_Disconnect(uint8 chan)</td>
<td>Disconnect only &quot;chan&quot; signal</td>
</tr>
<tr>
<td>void AMux_FastSelect(uint8 chan)</td>
<td>Disconnect the last channel that was selected by the AMux_Select() or AMux_FastSelect() function, then connect the new signal &quot;chan&quot;.</td>
</tr>
<tr>
<td>void AMux_DisconnectAll(void)</td>
<td>Disconnect all channels</td>
</tr>
</tbody>
</table>
- Current Source or Sink
- 3 Current Ranges

**Analog Current DAC**
Analog Current DAC

- Start
- Enable
- Init

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

• Four Different Configurations
Communication and Digital Blocks
SPI: Example of Communication Block
Configure 'SPI_Master'

- Name: SPIM_1
- 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 RX FIFO Full
  - Interrupt On RX FIFO Not Empty
  - Interrupt On RX FIFO Overrun
  - Interrupt On Byte/Word Transfer Complete
  - Interrupt On SPI Idle

Buttons: Data Sheet, OK, Apply, Cancel
Function Name: SPI_M_Start

Summary:
Initialize and Enable the SPI Master component.

Parameters:
None.

Return:
None.

Global variables:
SPIM_initVar - used to check initial configuration, modified on first function call.

Theory:
Enable the clock input to enable operation.

Reentrant:
No.

void SPI_M_Start(void)
{
    if(SPIM_initVar == 0u)
    {
        SPIM_Init();
        SPIM_initVar = 1u;
    }
    SPIM_Enable();
}
/* ======== */

/*
* 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 <SPI1.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 */
The main steps for creating embedded design in PSoC Creator are:

1) Configure
2) Develop
3) Debug
4) Reuse
• **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
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
Pin Out for CapSense Slider
<table>
<thead>
<tr>
<th>Alias</th>
<th>Name</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS_slider_e3</td>
<td><code>\CapSenseSlider:sbCSD:cPort\[3]</code></td>
<td>P3[3]</td>
</tr>
<tr>
<td>LS_slider_e2</td>
<td><code>\CapSenseSlider:sbCSD:cPort\[2]</code></td>
<td>P3[2]</td>
</tr>
<tr>
<td>LS_slider_e1</td>
<td><code>\CapSenseSlider:sbCSD:cPort\[1]</code></td>
<td>P3[1]</td>
</tr>
<tr>
<td>LS_slider_e0</td>
<td><code>\CapSenseSlider:sbCSD:cPort\[0]</code></td>
<td>P3[0]</td>
</tr>
<tr>
<td>sCmod</td>
<td><code>\CapSenseSlider:sbCSD:cCmod\</code></td>
<td>P5[4]</td>
</tr>
<tr>
<td>dPins_LSB[3:0]</td>
<td></td>
<td>P2[3:0]</td>
</tr>
<tr>
<td>dPins_MSB[3:0]</td>
<td></td>
<td>P4[3:0]</td>
</tr>
</tbody>
</table>

**Pin Assignment**
<table>
<thead>
<tr>
<th></th>
<th>P3[0]</th>
<th>CapSense slider element 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>P3[1]</td>
<td>CapSense slider element 2</td>
</tr>
<tr>
<td>45</td>
<td>P3[2]</td>
<td>CapSense slider element 3</td>
</tr>
<tr>
<td>46</td>
<td>P3[3]</td>
<td>CapSense slider element 4</td>
</tr>
<tr>
<td>31</td>
<td>P5[4]</td>
<td>CapSense Modulator capacitor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>P2[0]</th>
<th>LED 1 drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>95</td>
<td>P2[1]</td>
<td>LED 2 drive</td>
</tr>
<tr>
<td>96</td>
<td>P2[2]</td>
<td>LED 3 drive</td>
</tr>
<tr>
<td>97</td>
<td>P2[3]</td>
<td>LED 4 drive</td>
</tr>
<tr>
<td>69</td>
<td>P4[0]</td>
<td>LED 5 drive</td>
</tr>
<tr>
<td>70</td>
<td>P4[1]</td>
<td>LED 6 drive</td>
</tr>
<tr>
<td>80</td>
<td>P4[2]</td>
<td>LED 7 drive</td>
</tr>
<tr>
<td>81</td>
<td>P4[3]</td>
<td>LED 8 drive</td>
</tr>
</tbody>
</table>
```
#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_InitializeAllBaselinea();

    while(1)
    {
        /* Scan and update CapSense slider sensor */
        CapSenseSlider_CSD_ScanAllSlots();
        CapSenseSlider_CSHL_UpdateAllBaselinea();

        /* Get Centroid position of the finger on the slider */
        CentroidPosition = (uint8) CapSenseSlider_CSHL_GentroidPos(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 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