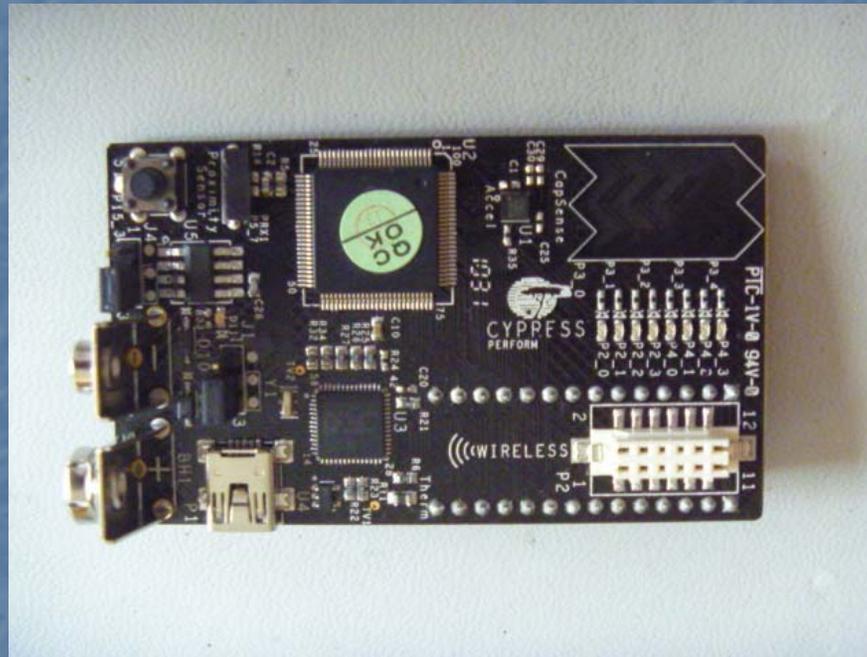


Using the Cypress PSoC Processor



January 15, 2011

Lloyd Moore

President/Owner, CyberData Corporation

Overview

- Programmable Systems On Chip
- Cypress Family
- PSoC 3/5 Architectural Overview
- Configurable Components
- Development Environment
- Cool Peripherals For Robotics
- Best Practices
- Resources

Programmable Systems on Chip

- Single chip contains:
 - Traditional processor
 - Traditional peripherals
 - CPLD/FPGA hardware
 - Analog hardware
 - Programmable analog hardware
- Primary advantages are reduced part count, reduced cost, increased flexibility and increased reliability
- Several now on the market
 - Actel SmartFusion
 - Xilinx
 - Cypress PSoC

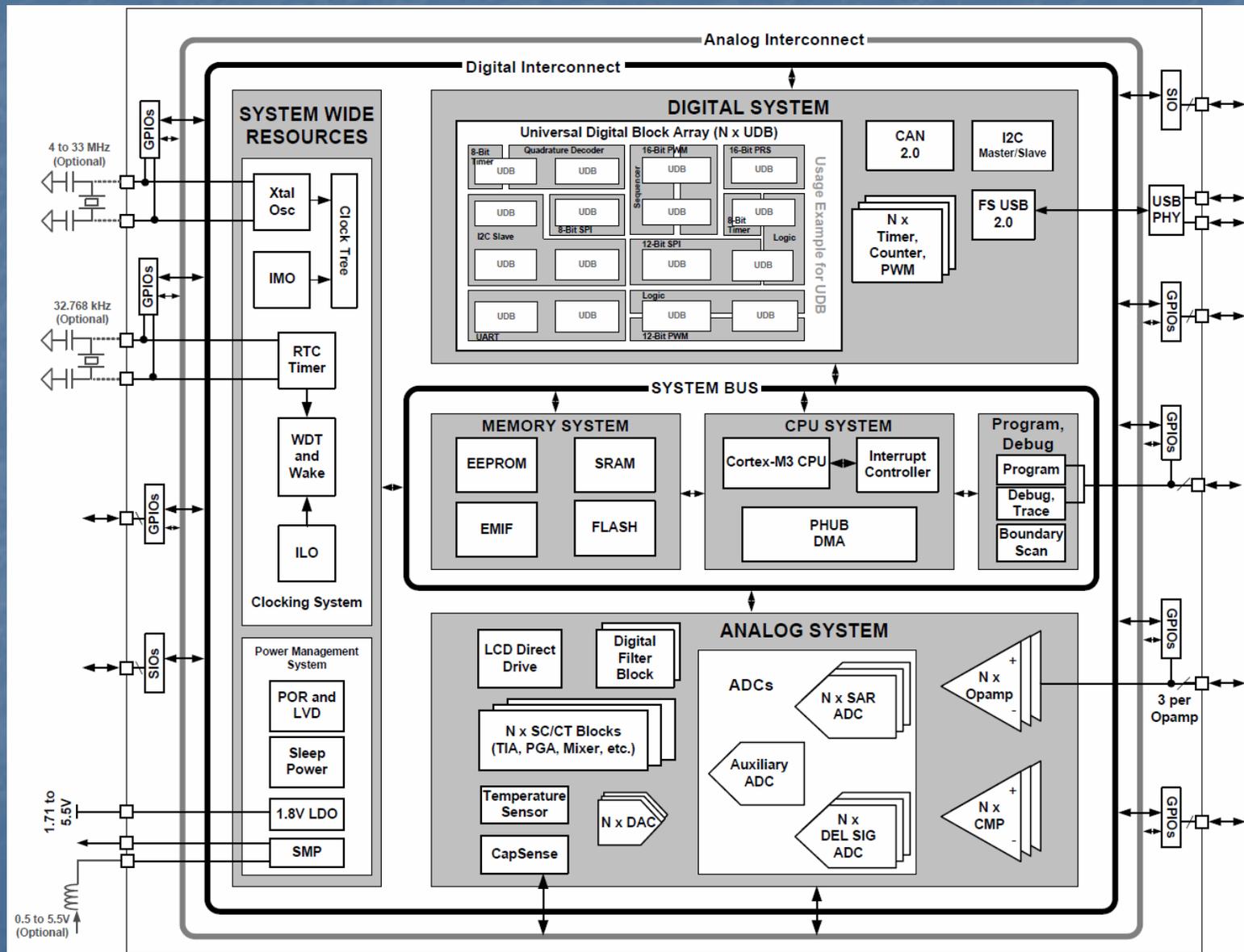
Cypress PSoC Family

- PSoC 1
 - The original design
 - 8 bit M8C core, 4 MIPS
 - Available since 2001
- Power PSoC
 - Simple devices, for LED lighting and motor control
 - Have high current FETs on board (1A range)
- PSoC 3
 - Redesign of development tool chain
 - Redesign of analog blocks
 - Enhanced 8051 core, 33 MIPS
 - Production parts available December 2010
- PSoC 5
 - Same base architecture as PSoC 3 + 2 additional SAR ADCs
 - 32 bit ARM Cortex M3 core, 100 Dhrystone MIPS
 - Sampling now, production parts Q2 2011
- Will be talking **only** about the PSoC 3 & 5 today

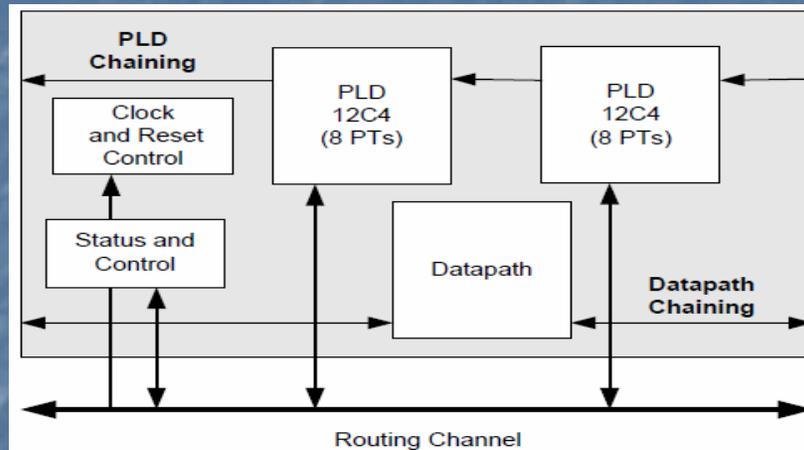
Cypress PSoC Family

- Programmable in 'C' or assembly
 - 'C' is the default / recommended language
- Low cost of entry
 - \$50 to \$250 for development kits
 - Development tools are a free download
 - PSoC Creator
 - PSoC Programmer
 - C – Compiler (Keil for PSoC3, GCC for PSoC5)
- Good balance of processor, analog blocks and digital blocks
- Single chip solution for many designs
- Can start with the PSoC 3 and easily migrate to PSoC 5 when available / needed
 - Plan for this in your design to make it smooth
 - Only a recompile needed in many cases
 - CANNOT migrate to/from PSoC 1

PSoC 3 & 5 Architecture



Universal Digital Blocks

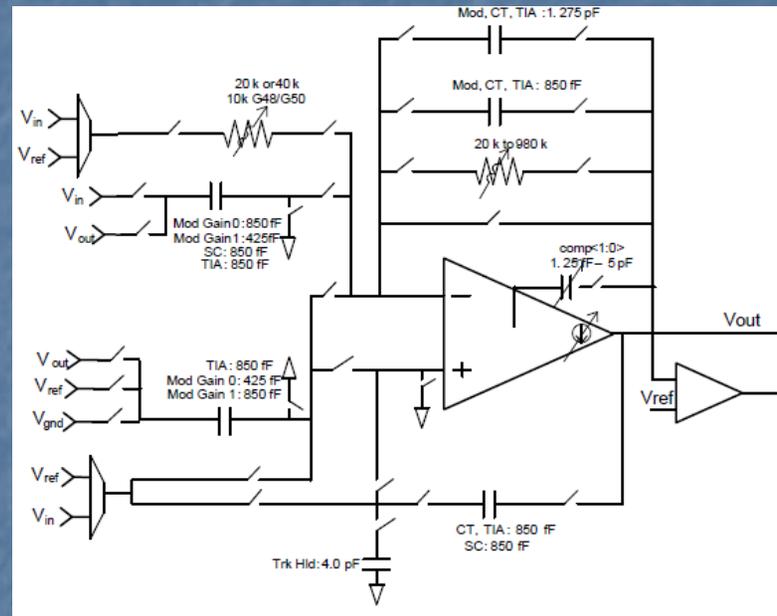


- Each block contains 2 PLD blocks
- The Datapath is a programmable ALU
 - 8 bit single cycle ALU with shift and mask operations
 - 2 Accumulators
 - 2 Data registers
 - 2 FIFO banks (4 bytes deep)
- PLD usage can be decoupled from Datapath
 - Allows for more efficient resource mappings
 - Cross individual block boundaries
- PLD and Datapath blocks can be chained for 16-32 bit operations

UDB Uses

- Counters
- Timers
- PWM
- UART
- Combinational discrete logic
- Pseudo random number generator
- Quadrature Encoder
- CRC
- Many, many others

SC/CT Blocks



- Basically an op-amp core with programmable resistors and capacitors attached
- Used for building
 - More op-amps!, Programmable gain amplifiers
 - Transimpedance amps, Mixers
 - Sample and hold amplifiers

Prebuilt Components

- Hardware components normally combined to make what you need for your application
- Large library of prebuilt components available
- Each component consists of
 - Hardware specification (Graphical or VHDL)
 - Schematic symbol for graphical editor
 - 'C' API
 - Datasheet
 - Custom configuration dialog
- Extensive abilities to also create your own components
 - VHDL or graphical design for hardware description
 - PDF for the datasheet
 - .NET for configuration dialog
 - Complete SDK in the tool chain for building these

Development Environment

- PSoC Creator – Full IDE
 - Pretty similar to Visual Studio or Eclipse
 - Hardware aspects programmed with a schematic capture style tool
 - Integrated programmer / debugger

The screenshot displays the PSoC Creator IDE interface. On the left, the Component Catalog shows various hardware components like ADC, DAC, and CapSense. The central workspace shows a schematic diagram with a 'CapSenseProximity' component and an 'LED Control Block' containing an 'LED Control Bus' and an 'AND' gate. The right pane shows C code for the main function, which configures the CapSense proximity sensor and controls the LEDs based on the sensor's output. The bottom pane shows the output window with compilation and execution logs.

```
mainc
1: Turn on LEDs depending on proximity of the finger to the proximity sensor
.....
2: Array which holds slot signal values: slot signal = rowcount - baseline
.....
extern uint8 CapSenseProximity_CSM_SlotSignal[ (CapSenseProximity_TOTAL_SCANSLOT_COUNT - CapSense
void main()
{
    uint8 Sensitivity = 15; // Constant used to scale down the proximity sensor value to calc
    uint8 LedData = 0; // This value is used to calculate the number of LED's to be turn
    uint8 ProximitySignal = 0; // This variable is used to store proximity sensor value
    uint8 i = 0;

    /* Enable global interrupt */
    CYGlobalIntEnable;
    /* Turn off all LEDs */
    LED_Control_Bus_Write(LedData);
    /* Start and Initialize CapSense */
    CapSenseProximity_Start();
    CapSenseProximity_CSM_InitializeSlotBaseline(CapSenseProximity_CSM_PROX_PROXIMITY);

    while(1)
    {
        /* Scan and update sensor baseline sensor */
        CapSenseProximity_CSM_ScanSlot(CapSenseProximity_CSM_PROX_PROXIMITY);
        CapSenseProximity_CSM_UpdateSlotBaseline(CapSenseProximity_CSM_PROX_PROXIMITY);

        /* Check if sensor is active */
        LedData=0;
        if(CapSenseProximity_CSM_CheckIsSlotActive(CapSenseProximity_CSM_PROX_PROXIMITY))
        {
            /* Read current proximity signal */
            ProximitySignal = (uint8)CapSenseProximity_CSM_SlotSignal[CapSenseProximity_CSM_PR
            /* divide the signal to turn the LEDs on depending upon signal value */
            tempVar = ProximitySignal / Sensitivity;
            for(i = 1; i <= tempVar; i++)
            {
                LedData = LedData << 1;
                LedData = LedData | 1;
            }
        }
        /* Update the LED state*/
    }
}
```

CapSense Module

- Very flexible architecture
 - Individual buttons, matrix of buttons
 - Linear sliders, radial sliders
 - Full touch pads
 - Proximity sensors
 - Works with most materials, not just fingers!
- Manual and automatic tuning
 - GUI tuning application to assist

LCD Module

- Modes:
 - Character
 - Graphic
 - Segment
- Character LCD VERY helpful for debugging even if not needed for application
 - Very simple API, base data types supported
 - Can upgrade default LCD in dev kit
 - Lumex family of character displays, 4/8 bit parallel interface

Boost Converter

- Input from 0.5V to 5.5V
 - Allows for input from a solar cell / energy harvesting
- Requires an external inductor, capacitor and optional diode
- Source up to 30mA with internal diode, 50mA with external diode
- Can also be used to form a regulated power supply not used by the processor

Best Practices

- Develop the hardware layer first
 - Will automatically generate API for you
- Implement timing critical functions in hardware, complex functions in software
- Use provided APIs and macros
 - Allows for component upgrades automatically
 - Allows for conversion of projects between PSoC3 and PSoC5 with minimal changes
 - Also different processors in the same family

More Best Practices

- Reserve and use “debug pins” to access internal signals
- Keep digital and analog signals separate
 - Analog domain clocks available
 - Processor die split, orient this with board layout
- Use dedicated pins for internal op-amps
 - Reduced silicon switch resistance
- Buffer analog signals
 - Most internal sources have very minimal drive capability

And MORE Best Practices

- Watch for hardware race conditions
 - Use a higher frequency clock and “sequencer block” to control complex sequential logic
 - Many resets and presets require a clock cycle to take effect
 - May look like it works, but happening “by chance”
- Read and know the items on the errata sheets!
 - Development tools can also adjust for errata items
- Reserve time for experimentation
 - MANY ways to do things on this processor
 - Power of the processor comes from flexibility
 - Distinction between hardware and software largely gone
 - Combining hardware and software approach often the cleanest and simplest solution

Summary

- The PSoC processor integrates programmable analog and digital with a traditional processing core
- The PSoC can be a one chip solution for many robotics projects
- Develop the hardware configuration first and then develop the software, leaving time to experiment with different configurations

Resources

- PSoC Product Web Site:
 - <http://www.cypress.com/?id=1353>
- PSoC Developer Community:
 - <http://www.psocdeveloper.com/forums/>
- PSoC Training On Demand:
 - <http://www.cypress.com/training>
- PSoC 5 FirstTouch Starter Kit: \$50
 - <http://www.cypress.com/?rID=43674>
- PSoC Full Development Kit (1, 3, 5): \$249
 - <http://www.cypress.com/?rID=37464>
- My Contact Info:
 - Lloyd@CyberData-Robotics.com
 - <http://www.CyberData-Robotics.com>

Questions????

- IDE demo if time
- Will be around a bit after the meeting for individual questions
- Feel free to e-mail me