Universidad Autónoma de Madrid
Escuela politécnica superior

Proyecto fin de carrera

ANÁLISIS DE LA TECNOLOGÍA PSOC APLICADA A REDES DE SENSORES

Ingeniería de Telecomunicación

Pablo Bacho Manzorro
Febrero, 2013

ANÁLISIS DE LA TECNOLOGÍA PSOC APLICADA A REDES DE SENSORES

AUTOR: Pablo Bacho Manzorro
TUTOR: Gustavo Sutter

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Febrero, 2013

Abstract

Abstract

Sensor networks are all over the world, technology is getting better and cheaper everyday and their usefulness is indisputable. However and despite the prices drop, industrial targeting of these networks makes final systems hardly affordable for budget-tightened users that otherwise could benefit from their features.

This work aims to introduce an affordable and flexible sensor network solution, leveraging open-source and other low-cost widely available technologies, to help those who could use a sensor network but for the required investment by commercial systems.

PSoC is a programmable embedded System-on-a-Chip manufactured by Cypress Semiconductor, integrating configurable analog and digital peripheral functions, memory and a microcontroller on a single chip. Because of its rich analog and digital capabilities, it has become the cornerstone of the Wireless Sensor Network proposed on this document. Around PSoC, other technologies are deployed, such as Zabbix and Raspberry Pi, to build a widely accessible and friendly system.

Key words

Wireless Sensor Network, WSN, Programmable System-on-Chip, PSoC, programmable logic, programmable analog, signal conditioning, sensing, XBee, Raspberry Pi, Zabbix

Resumen

Las redes de sensores se encuentran por todo el mundo, los grandes avances en su tecnología y el descenso coste de la misma es indiscutible. A pesar del descenso de precios, la orientación industrial de estas redes aumentan su valor en el mercado, haciéndolas inaccesibles para usuarios con presupuestos más limitados, que de otra forma podrían beneficiarse de sus ventajas.

Este trabajo presenta una solucin asequible y flexible para desplegar redes de sensores, aprovechando una serie de tecnologías libres o de bajo coste y ampliamente disponibles, con el fin de ayudar a aquellos que puedan hacer uso de una red de sensores avanzada, pero sin hacer frente a la inversión requerida por los sistemas comerciales.

PSoC es un sistema embebido desarrollado por Cypress Semiconductor que integra funciones programables analgicas y digitales, memoria y un microcontrolador en un único circuito integrado. Dadas sus altas prestaciones analógicas y digitales, se ha convertido en la piedra angular de la red de sensores propuesta en este documento. Alrededor de PSoC, otras tecnologías entran en juego, tales como Zabbix y Raspberry Pi, para construir un sistema amigable y ampliamente accesible.

Palabras Clave

Red de sensores, WSN, Programmable System-on-Chip, PSoC, lógica programable, analógica programable, acondicionamiento de señal, XBee, Raspberry Pi, Zabbix

Contents

List of Figures

List of Tables

1 Introduction

1.1 What is a sensor network

A sensor network consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and to cooperatively pass their data through the network to a main location.

Sensor networks are usually wireless -therefore known as Wireless Sensor Networks (WSNs)-, providing easy, fast and cost effective deployment. Their battery life extends within months to a few years, and their communication range from a few meters to several kilometers, or even worldwide using the existing telecommunications infrastructure.

1.2 Applications

Sensor networks have endless applications, hereby are some of them to help making an idea of their usefulness.

Area monitoring
A number of nodes are deployed in the monitored area, triggering alarms whenever an intrusion or anomaly is detected in the perimeter. They can be used to guard critical infrastructure such as gas and oil pipelines, as well as non-critical facilities given the low cost of WSNs.

Environmental sensing
Nodes deployed in different locations measure air, soil and water quality parameters to control the air polution, soil moisture, water quality, etc. These sensor networks are used, for example, in waste-water plants where electricity and broadband communication networks are not always accessible. They can be used for early detection of forest fires, floods or landslides. Or can even be deployed off-shore to monitor maritime traffic and help protect the sea wildlife [28].

Industrial automation
Sensor networks are used for industrial processes automation and machine health monitoring. The latter subject is of special interest, motes can be placed in dangerous areas for workers as well as, since they are wireless, on moving parts.

Agricultural monitoring
Wireless sensor networks are deploy in field crops and greenhouses, optimizing the management of resources. They can automate irrigation streamlining water use, preempt frosts and detect diseases early.

Structural monitoring
Motes placed on mechanical stress points of a building, bridges or railroads can monitor vibration, stretching, cracks, etc. without the need of halting the activity and being more accurate than a visual inspection.

1.3 State of the art

There are a high number of sensor networks solutions in the market. In this section, some of the leading solutions that allows the user to customize in anyway the system deployed are commented.

1.3.1 Community and non-for-profit solutions

Community solutions are leaded by Arduino. Most of these sensor networks have Arduino-compatible motes, sharing tools, libraries and hardware.

Arduino

Arduino is an open-source electronics prototyping platform. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing).

The boards can be built by hand or purchased preassembled. The software can be downloaded for free. The hardware reference designs (CAD files) are available under an open-source license.

Though it is a general purpose prototyping platform and it has no native radio capabilities, Arduino has become the base of many WSNs developed by amateurs and non-profit organizations. Most of these projects use XBee radios, or any other radio platform that can be controlled via UART port.

It is offered in several versions with diferent shapes and MCUs, the core Arduino is offered by the genuine manufacturer at EUR 20.

panStamp

panStamps are small wireless boards specially designed to fit in low-power applications, with the philosophy of *simple to program and simple to work with.*

There is a catalog of boards containing sensors and actuators to plug a panStamp. Wireless communications is performed using a protocol called SWAP over 868/915 MHz.

panStamps are Arduino clones so they use the same IDE and libraries. They provide the full solution with Lagarto server, a custom software for operating the network.

1.3.2 Commercial solutions

Commercial solutions are often powered by Texas Instruments MSP430 and Atmel ATmega MCUs.

WaspMote by Libelium

WaspMote is an Arduino-based sensor node manufactured by Libelium, a Spanish company based in Zarazoza.

The core of the WaspMote is an ATmega1281, it has one connector for a *sensor shield* which is an external board with the sensors themselves, and one connector XBee-compatible for communications. It features some on-board peripherals such as a Real Time Clock (RTC) and an accelerometer.

Firmware is developed using a proprietary Integrated Development Environment (IDE) based on Arduino IDE and sensors monitoring and control is performed also via proprietary software, both developed by the manufacturer.

It has an open Application Programming Interface (API), but the hardware itself and the monitoring software are closed-source.

WaspMote is offered in different radio configurations starting at EUR 130 [30].

Tinynode

Tinynode is a Swiss-based company developing wireless sensor networks for vehicle detection.

The company also offer their platform, Tinynode 584, which features a Texas Instruments MSP430 MCU as the core running the TinyOS Real-Time Operative System (RTOS) and a Semtech radio transceiver for communications. The aim of this platform is to provide easy wireless communication capability to sensors, actuator and controllers.

It makes use of TinyOS Oscilloscope software to display the data.

The board itself is proprietary and closed source. It has a retail price of EUR 109.

nCore by Nebusens

This company based in Spain offers some devices to build a sensor network from specific small ID tags to customizable nodes.

The network is ready to run out-of-the-box, the nodes are already programmed with a specific firmware, flexible enough to read external sensors with no embedded programming. All the development has to be done in the control center side, where a PC sends the configuration parameters to the nodes and requests a capture or actuation.

The firmware, API and hardware designs are closed-source. Retail price is not disclosed.

1.3.3 Trends

After reviewing some of the systems used in sensor networks, some conclusions can be reached:

- Arduino is the leading platform in open-source solutions.
- For commercial systems, most common MCUs are the MSP430 and ATmega families.
- Flexibility of the nodes is constrained to the capabilities of these MCUs, which have scarce resources and low computing power.
- Custom software is the common approach to interact with the systems.
- Communication is carried via standard protocols and proprietary stacks indifferently.

1.4 Objectives and approach

This work aims to introduce a powerful and flexible, and yet affordable sensor network solution, leveraging open-source and other low-cost widely available technologies.

To reach the widest spectrum of users, the solution featured on this work have been designed according to the following criteria:

Innovative
Not following the "classic" (as classic as they can be) trends in WSNs.

Low cost
Accessible to everyone, from corporations to single hobbyists.

Multi-purpose
There is not working environment or application in mind. Motes can be deployed either fixed or mobile, buried or off-shore... And as many different kinds of sensors as possible must be integrable in a mote.

Standalone
A complete solution regards nodes, gateways, data storing and a friendly user interface.

Open
Open-source and popular components ensure ease of access to them, liability in the long term, plenty of documentation and help from the community.

Easy to deploy
No deep understanding of the inner workings of the system must be required to deploy and run a network.

Extendable
For the special case that needs more than the common ready-to-deploy features.

1.5 Structure of the document

The present document is organized as follows:

1. Introduction
   This is the introduction of the document, which the reader have just read.
2. An approach to a flexible and easily customizable sensor network
   The system under study is introduced with a brief introduction of the technologies involved.
3. Introduction to PSoC
   An extensive introduction to PSoC, the core of the system, with deep insight in its inner technical details.
4. PSoC and sensors
   Several sensors are arranged by their working principle, solving their signal conditioning problem using PSoC capabilities.
5. PSoC communication capabilities
   Introduction of some of the communication features of PSoC. Introduction by an example how these can be expanded.
6. Power management on PSoC
   An overview of PSoC power system distribution, and low-power modes of operation.
7. Node design considerations
   Some guidelines regarding PSoC internals are proposed for the construction of a node. Deep information on physical organization of PSoC.
8. Conclusions and future work
   Conclusions and considerations are stated on this chapter.

After the study, a prototype of the system has been deployed successfully. Details on this prototype can be found in Appendix B.

2

An approach to a flexible and easily customizable sensor network

2.1 Introduction

This chapter proposes a WSN solution ready to work out-of-the-box, regarding sensing motes, gateway and management software.

2.2 Objectives

The solution introduced in this chapter has been design under the mission of creating a system *easy to acquire and deploy*.

That said, the user must be able to acquire the devices that fulfill his or her needs and deploy the network with no specific technical knowledge. In the other hand, this statement will not limit the possibility of advanced deployments.

To achieve this, the sensor nodes have to be available with different predefined configurations to satisfy the different needs that may arise, designing a flexible mote that boosts possibilities lowering the investment.

2.3 Design challenges

2.3.1 Technical design challenges

Low power consumption
Nodes are usually battery-powered, thus low power drain from batteries and energy harvesting modules is an essential feature for a wireless mote.

Dynamic network topology
This allows nodes to be placed in more complex network architectures covering wider areas with a single network.

Node failure tolerance
An extension of the dynamic network topology is the capability of re-routing packets in case of a node starts dropping connections.

Scalability and heterogeneity of nodes
Covering a wide spectrum of applications requires a high collection of different nodes with different sensors and communication interfaces. Seamless integration of new sensors.

Unattended operation
The sensor network has to operate 24/7 with no human intervention at all.

2.3.2 Strategic design challenges

R&D investment
Research and Development has to be reduced compared to other solutions, to lower the final price of the system.

Manufacturing cost
The lower the manufacturing cost, the lower the final price of each device of a network.

Time-to-market
World is moving fast and short time-to-market is essential developing new solutions.

Ease of support/upgrade
The system has to provide easy maintenance and extensibility during the product life-cycle.

2.4 Network overview

Figure 2.1 illustrates an example configuration of a sensor network. The devices involved in the network are briefly introduced subsequently.

2.4.1 Nodes

They carry two main tasks: *sensing* and *message forwarding*.

Not every node carries these two tasks together, message forwarding usually requires the node to be active all time given the random reception of messages from other nodes and this is power-consuming. So the common approach is having sensing nodes and routing nodes independently, though routing and sensing capabilities could be implemented in a single node if desired.

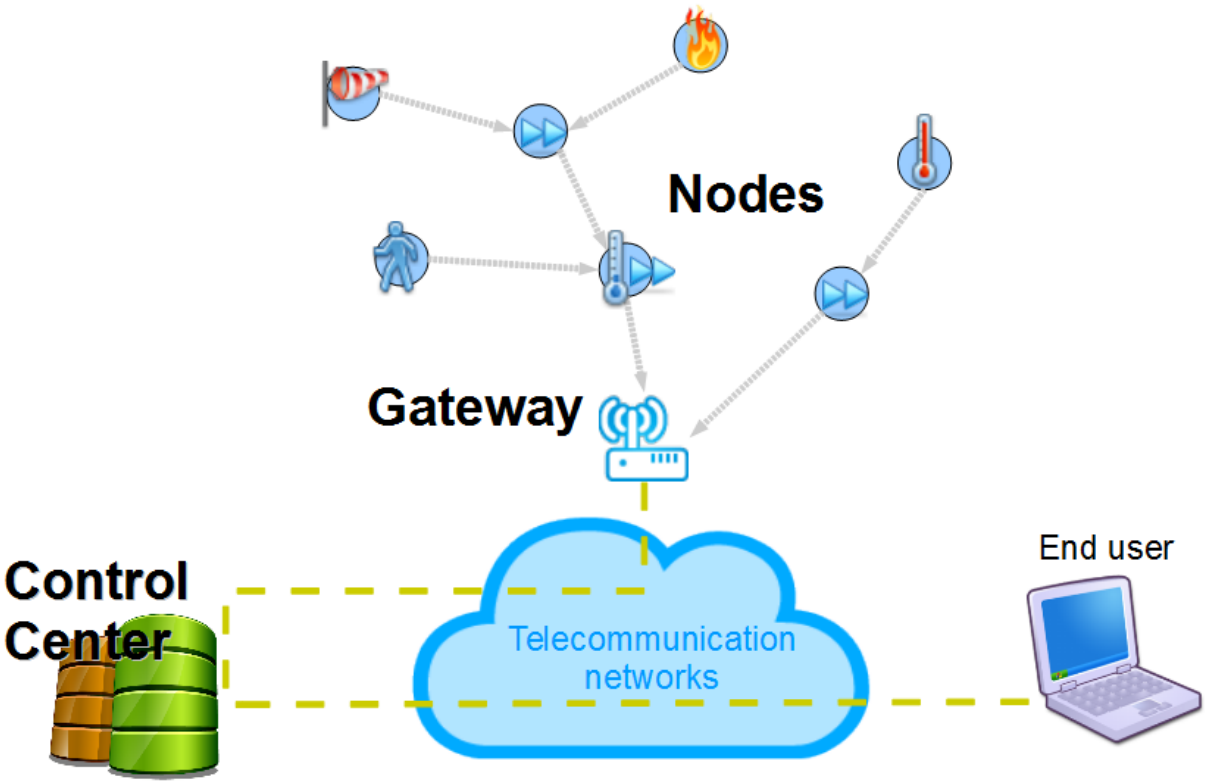


Figure 2.1: WSN typical structure

2.4.2 Gateway

A gateway gathers data from the network and forwards it to the control center.

It is also responsible for bridging, if needed, the communication network used by the nodes (ZigBee, BlueTooth...) with common telecommunications infrastructure used by the control center (Ethernet, WiFi...).

2.4.3 Control center

It is responsible for storing, processing and displaying in a human readable and friendly way the data collected from the sensor network, as well configuring its parameters and issuing commands to it.

2.5 Zabbix at the control center

2.5.1 What is Zabbix?

Zabbix is an enterprise-class open source distributed monitoring solution [34].

It is deployed worldwide in small and big organizations, and corporations, monitoring hundreds of parameters across thousands of devices.

Zabbix offers monitoring, alerting, visualization, and proactive response features.

2.5.2 How is it used at the control center

Zabbix monitors computers' parameters such as CPU temperature, hard-disk space usage, whether or not services are running, etc. This can be accomplished via Simple Network Management Protocol (SNMP) or Intelligent Platform Management Interface (IPMI) protocols for network electronics and a software agent for servers and computers.

In a sensor network, the agent behavior is emulated and these parameters just have to be switched to temperature, humidity, PIR... and it will interpret and handle them exactly the same way it does with computers.

2.5.3 Advantages using Zabbix

Technical advantages

1. Being open-source it can be studied, modified and adapted to suit the particular needs of a sensor network.
2. Centralized monitoring system allows to store all information (configuration and performance data) in relational database for further easier processing and re-use of data
3. Built-in rich visualization capabilities including numeric data, history, graphics, up-to-date slides...
4. Zabbix runs on any computer and server architectures able to run a Linux distribution and some required dependencies.

Strategic advantages

Using Zabbix at the control center leverages economies of scale, resulting in an outstanding software with no development nor acquisition cost, and very little cost of deployment and ownership.

1. Full featured enterprise-class monitoring system ready to work out-of-the-box.
2. GPL license and outstanding support from the community ensures little cost of deployment and ownership.
3. It is enterprise oriented with all the benefits that implies: robustness, reliability, support...
4. Being a professional IT security platform, it is constantly being audited by security professionals.
5. Zabbix company is extremely active on its development and support, providing frequent updates.

2.5.4 Withdraws using Zabbix

The main withdraw using Zabbix is that it is meant for computer networks, this has some implications:

1. It is feature-bloated for sensor networks.
2. The development of new features is computer networks oriented and may not fill sensor networks needs.

Thus, specific sensor networks needs have to be integrated into Zabbix. However, similarities between computer and sensor networks are so many that Zabbix comes right out-of-the-box with everything it is needed to operate a sensor network. Refer to Appendix B for further details.

2.5.5 Alternative technologies

All WSNs manufacturers seem to have implemented their own monitoring software. So there are few alternatives:

Cosm

Cosm is a platform with an open API for sensor data storage and visualization. It runs exclusively on Cosm's servers and data is either shared publicly with its free license or managed privately with a commercial license.

Cacti

Cacti is an open-source tool for storing data and creating graphics upon it. Very good at it, but lacks other functionality.

Other computer network monitoring solutions

Such as *Pandora FMS* or *Nagios* are equally valid for the tasks as Zabbix is. While *Pandora FMS* beats on front-end friendliness, it lacks somewhat on functionality and its community support is not as active as Zabbix's. *Nagios* is a full-featured reputable monitoring solution but, as it happens with Pandora FMS, only a lighter version is open-source, having somewhat restricted the availability of new features.

Full-custom software

The development of a full-custom software with the same functionality of Zabbix would be long in time and hence in economic resources, preventing the project to achieve its goals.

Zabbix is absolutely valid for common operation with sensor networks and it is highly extensible. Developing complementary tools for the special case is easier than a full-custom software with all the functionality, stability and safety that Zabbix provides.

2.6 Raspberry Pi as the gateway

2.6.1 What is Raspberry Pi

The Raspberry Pi is a credit-card-sized single-board computer developed in the United Kingdom by the Raspberry Pi Foundation with the intention of stimulating the teaching of basic computer science in schools [31].

The Foundation offers two versions, priced at US$ 25 and US$ 35:

	Model A	Model B
SoC	Broadcom BCM2835	
CPU	$700MHz$ ARM1176JZF-S core (ARM11 family)	
GPU	Broadcom VideoCore IV	
Memory (SDRAM)	$256MB$ (shared with GPU)	$512MB$ (shared with GPU)
USB 2.0 ports	1	(2 via integrated USB hub)
Video outputs	Composite RCA, HDMI, DSI	
Audio outputs	$3.5mm$ jack, HDMI	
Onboard storage	SD / MMC / SDIO card slot	
Onboard network	None	10/100 Ethernet (RJ45) via USB hub
Low-level peripherals	8 GPIOs, I^2C, SPI with 2 SS	
Power ratings	$300mA$ $(1.5W)$	$700mA$ $(3.5W)$
Power source	$5V$ via MicroUSB or GPIO header	

Table 2.1: Raspberry Pi models A and B features

The Foundation provides Debian and Arch Linux ARM distributions for download.

2.6.2 How is it used as the gateway

The Raspberry Pi runs three pieces of software that working together enable the transmission of data:

A service called *Zabbix Proxy* which handles the data forwarding to the Zabbix server.

A sensor interface controller that drives the communication interface exposed to the sensor network.

A data translator from sensor network frames to Zabbix Proxy's.

2.6.3 Advantages using Raspberry Pi

It is a low-cost computer which suits absolutely a low-cost sensor network.

It is a Linux computer running actual Linux programs. Developers find in Raspberry Pi a full-featured, familiar and friendly environment to develop for.

Extensible through external hardware connected via USB, UART, I^2C, SPI or General Purpose Input/Output (GPIO).

Great support from the community means shorter R&D time and an active development of new features and hardware parts available for the device.

It is a successful device meaning that it will be available long-term as well as supported by the manufacturer.

2.6.4 Withdraws using Raspberry Pi

Raspberry Pi foundation holds the intellectual property and the design files are undisclosed, limiting the adaptability of the device to custom needs.

Ethernet is the only communications standard supported on-board and a modem or other network equipment is needed to communicate the Raspberry Pi with a remote Zabbix server.

2.6.5 Alternative technologies

OpenWRT

OpenWrt is a Linux distribution primarily targeted at routing on embedded devices. It can be run on customer-premises routers and residential gateways, among other small computers.

OpenWrt is free and open-source software licensed under the GPL. It supports hardware from a list of 46 manufacturers, plus many others being worked on. Many of those manufacturers offer their products as Original Equipment Manufacturer (OEM). This include support for PCIe (Wifi and BlueTooth), USB host (3G modems) and slave, Ethernet, DSL, serial communications (UART, SPI..) and GPIOs.

The technical advantages using OpenWRT are similar to those using the Raspberry Pi board, with the addition that OpenWRT runs on actual routers, not needing a second network device for remote communications.

However, the big drawback using OpenWRT is not technical but strategic. It runs in so many hardware platforms that its development and support is somewhat restricted. It has many bugs, there is very little technical information about the hardware itself (since they are commercial products) and availability of software for some architectures is scarce. In comparison with Raspberry Pi, development for OpenWRT is a complex and time-consuming task.

Raspberry Pi has been such a successful device that there is plenty of software already adapted to it and much more being worked on.

Full custom solution

The Research and Development required to develop a full-custom device for this purpose would drive this project off from its goal of an affordable solution.

Both options considered are suitable for the purpose, with their advantages and disadvantages. Raspberry Pi has been chosen for its reduced cost and high popularity, but OpenWRT can be kept as a valid alternative.

2.7 Flexible architecture of the node

2.7.1 Heterogeneity of nodes

One of the goals mentioned on this document is to achieve the widest heterogeneity of nodes possible to suit multiple purposes with the same system.

To understand the approach taken on this proposal, four functional parts of the node have to be differentiated:

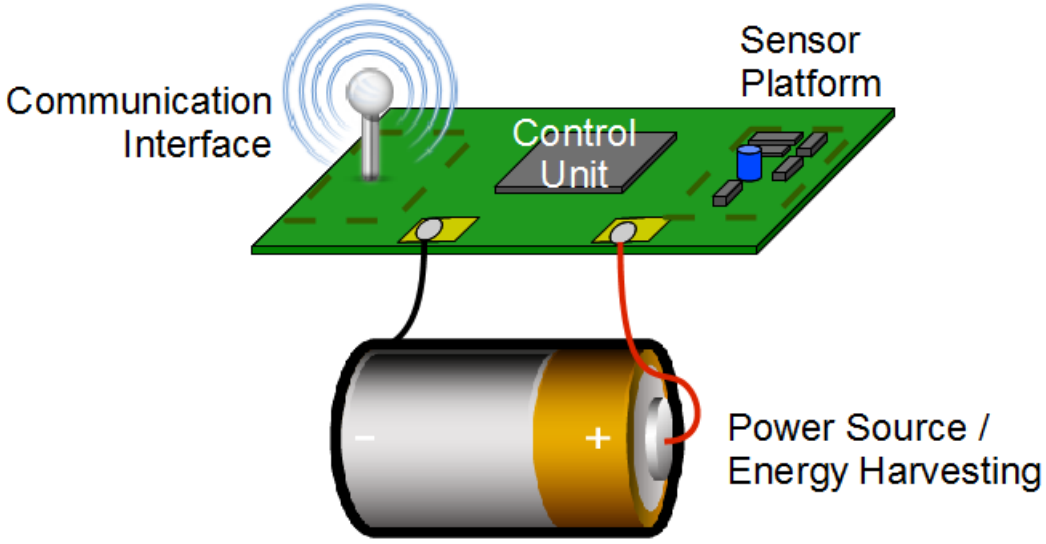


Figure 2.2: Node functional parts

Given this approach, the number of nodes to design and produce is:

- CU as the control unit.
- nSP as the number of sensing platforms.
- nCI as the number of communication interfaces.
- nPM as the number of power modules supported.

$$Number\ of\ nodes = 1\ CU * nSP * nCI * nPM$$

The number of different nodes to design, verify and produce increase geometrically as different communication interfaces or sensor platforms are available. This is one of the reasons why commercial systems stick to a single communication interface (as it is ZigBee) and narrow power source options (like the differential ± connector), trying to maximize just the number of supported sensors.

2.7.2 Breaking down the problem

Since the objective of this project is to build a flexible and easily customizable sensor network, it seems reasonable to break the main areas of the node down to physically independent but interconnected devices.

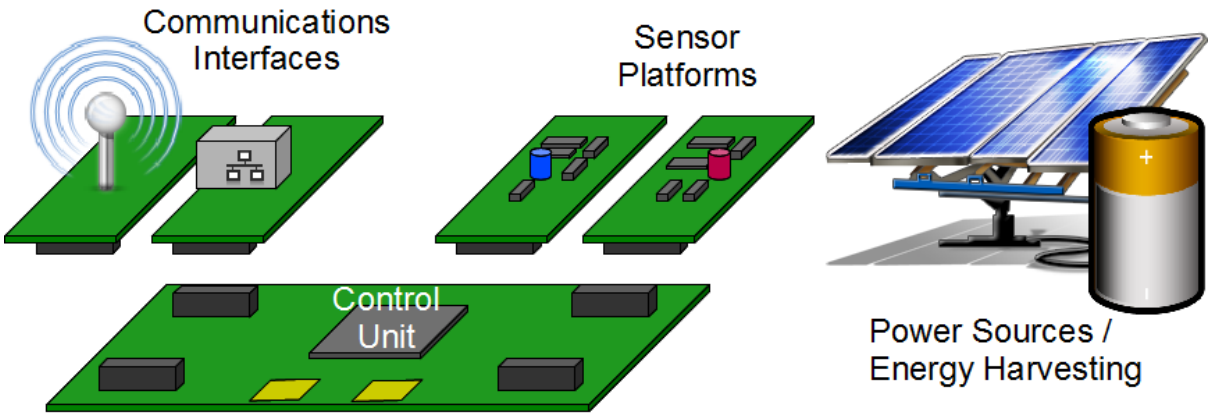


Figure 2.3: Independent node parts

With this approach:

$$Number\ of\ available\ nodes = 1\ CU * nSP * nCI * nPM$$

There is the same number of available nodes than with a singleton system, keeping the diversity of options and increasing them as communication interfaces and sensing platforms are designed, but:

$$Number\ of\ parts\ to\ design = 1\ CU + nSP + nCI + nPM$$

The number of parts to design, verify and produce to maintain the same heterogeneity of nodes have drop substantially.

Benefits

Shorter R&D
The required R&D for each part is amazingly simpler than redesigning the stand-alone board. New designs require much less effort regarding time and human resources. Design & Reuse philosophy.

Each part is produced in bigger bulk orders
There are less different designs to produce, so orders get better bulk rates.

Easy to support
Smaller system to deal with when releasing updates or fixes.

Technical challenges

Extreme hardware flexibility needed from the *Control Unit*:

- What peripherals will it need? How many of them?
- What kind of sensors will it have connected? Are they analog or digital?
- How to divide and route the I/Os?

Next chapter of this document will introduce a technology that addresses all this questions and allows the design of a node with this high degree of flexibility.

3

Introduction to PSoC

3.1 What is PSoC

PSoC is a programmable embedded System-on-a-Chip manufactured by Cypress Semiconductor, integrating configurable analog and digital peripheral functions, memory and a microcontroller on a single chip.

With PSoC the functions of a microcontroller, a Complex Programmable Logic Device (CPLD) and high-performance analog can be integrated with unmatched flexibility. This saves cost, board space, power and development time.

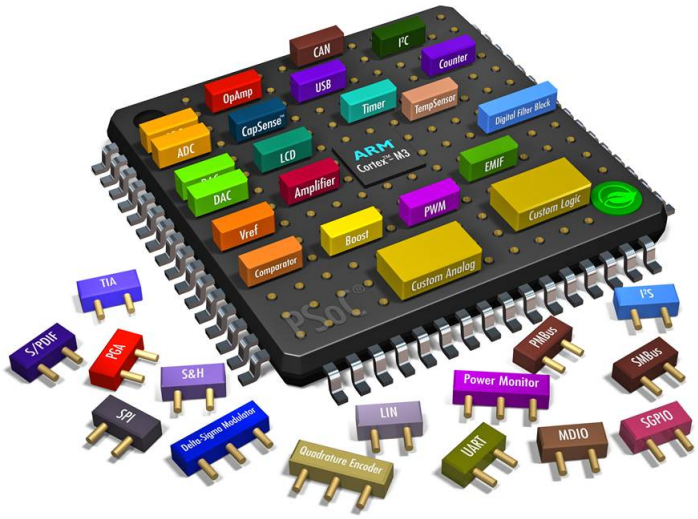


Figure 3.1: PSoC Commercial Picture

PSoC is presented in three different *flavours*: PSoC1, PSoC3 and PSoC5LP.

PSoC1 is the oldest version of the PSoC family, and somewhat limited in routability, analog capabilities and digital functions making it unsuitable for the purpose of this project. Thus, hereinafter only PSoC3 and PSoC5LP (that replaces the former PSoC5) characteristics will be referred, which are newer and designed under a radically different architecture.

3.1.1 PSoC3 and PSoC5LP architecture

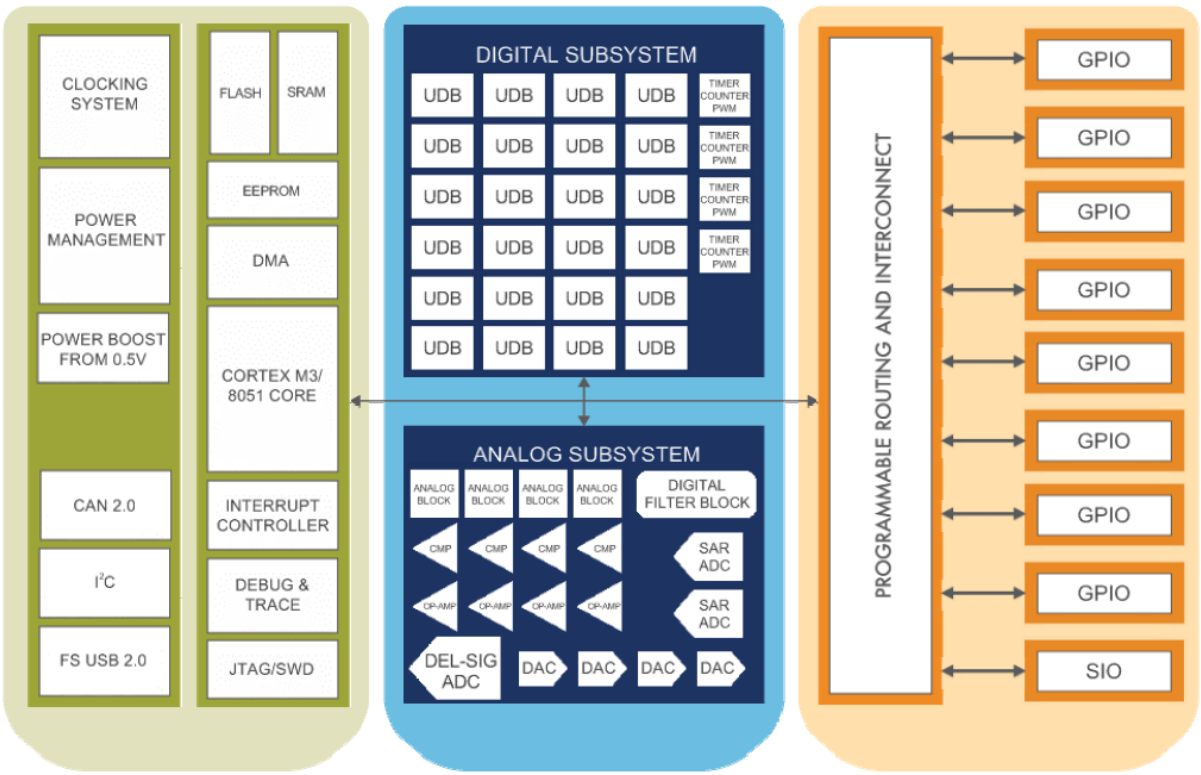


Figure 3.2: PSoC3 and PSoC5LP platform architecture

3.1.2 CPU characteristics

	PSoC3	PSoC5LP
CPU	8-bit 8051 CPU	32-bit ARM Cortex-M3
Cache		1KB cache
Speed	Up to $67MHz$, $33MIPS$	Up to $67MHz$, $84DMIPS$
Flash	$8KB$ to $64KB$	$32KB$ to $256KB$
SRAM	$2KB$ to $8KB$	$8KB$ to $64KB$
EEPROM		$2KB$
Operation	$0.5V$ to $5.5V$	$0.5V$ to $5.5V$

Table 3.1: PSoC3 and PSoC5LP core characteristics

3.1.3 Clocking System

The clocking system provides clocks for the entire device. It allows the user to trade off current, frequency, and accuracy. A wide range of frequencies can be generated, using multiple sources of clock inputs combined with the ability to set divide values.

The components of the clocking system block diagram are defined as follows:

1. Internal Main Oscillator (IMO)
2. Internal Low-Speed Oscillator (ILO)
3. A 4 to 25 MHz External Crystal Oscillator (MHzECO)
4. A 32 kHz External Crystal Oscillator (kHzECO)
5. Digital Signal Interconnect (DSI) signal, which can be derived from the clocks developed in Universal Digital Blocks (UDBs) or off-chip clocks routed through pins
6. A Phase-Locked Loop (PLL) to boost the clock frequency of some select internal and external sources

Figure 3.3 illustrates the configuration dialog with all the connection possibilities of these clocks, including sources and outputs, with an example of configuration.

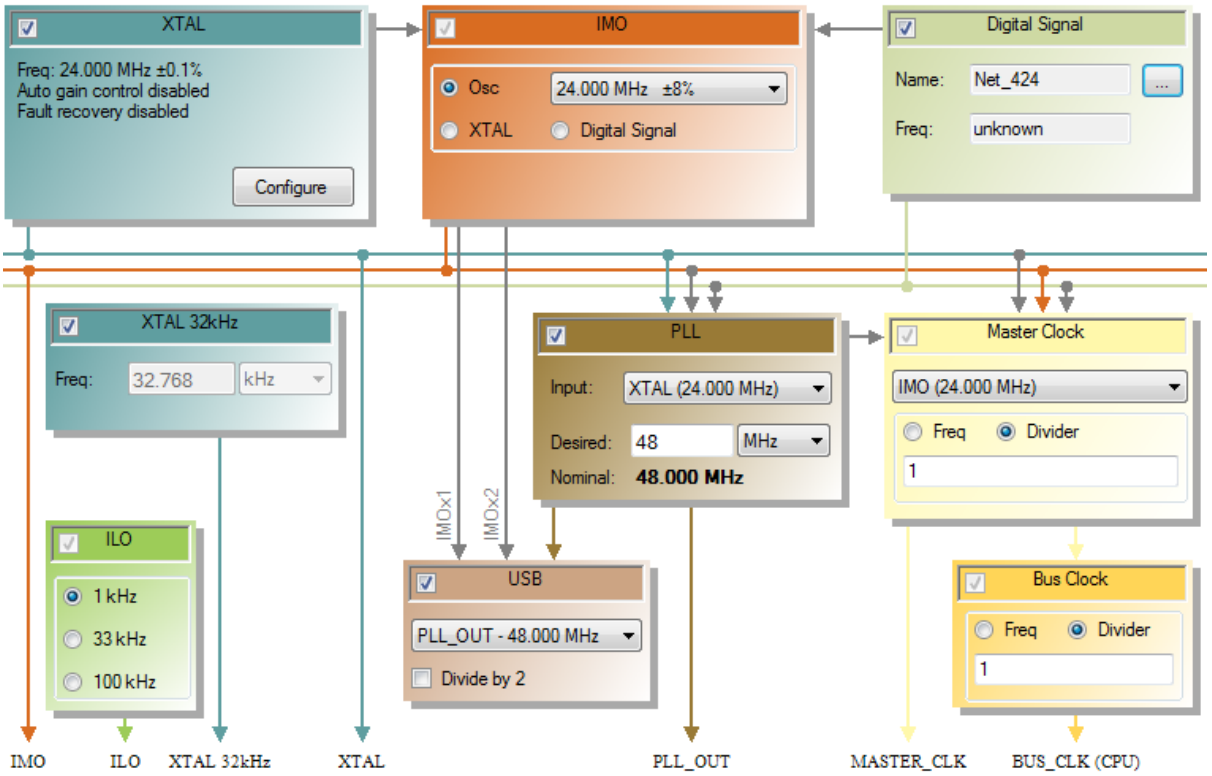


Figure 3.3: Clocking System configuration dialog

Table 3.2 summarizes the available clocks and their respective frequency ratings.

Source	F_{min}	F_{max}
IMO	3 MHz	62.6 MHz
ILO	1 kHz	100 kHz
MHzECO	4 MHz	25 MHz
kHzECO	32.768 kHz	
PLL	24 MHz	67 MHz

Table 3.2: Oscillator summary

3.1.4 Memory resources

SRAM

- Organized in blocks of $4KB$. Up to 16 blocks depending on family.
- 8-, 16- or 32-bit accesses. In PSoC3 devices the CPU has 8-bit direct access to SRAM.
- Zero wait state accesses.
- Arbitration of SRAM accesses by the CPU and the DMA controller.
- Different blocks can be accessed simultaneously by the CPU and the DMA controller.
- In PSoC5LP, code can be executed out of portions of SRAM.

Flash Program Memory

- Organized in rows, where each row contains 256 data bytes plus 32 bytes for either Error Correcting Code (ECC) or data storage.
- Organized as:

  PSoC3 One block of 64, 128n or 256 rows.

  PSoC5LP One block of 128 or 256 rows, or as multiple blocks of 256 rows each.
- Stores CPU program and bulk or nonvolatile data.
- Access:

  PSoC3 8-bit direct access.

  PSoC5LP 8-, 16-, or 32- bit read accesses.
- Programmable with a simple command /status register interface.
- Four levels of protection:

  1. Unprotected.
  2. Read Protect from external devices.
  3. Disable External Write.
  4. Disable Internal Write.

EEPROM

- Organized in rows, each row contains 16 bytes.
- Organized as one block of 32, 64, or 128 rows, depending on the device.
- Stores non-volatile data.
- Write and erase using SPC commands.
- Byte read access by CPU or DMA using the PHUB.
- Programmable with a simple command/status register interface.

External Memory Interface

PSoC provides an External Memory Interface (EMIF) for connecting to external memory devices and peripherals.

- Supports four types of external memory:
  1. Synchronous SRAM.
  2. Asynchronous SRAM.
  3. Cellular RAM/PSRAM.
  4. NOR flash.

  Up to 24 address bits. Memory can be 8 or 16 bits wide.

3.1.5 Any pin, any purpose

This is one of the mottoes under which PSoC is commercialized.

On PSoC *almost* any internal signal may be routed to *almost* any pin. This capability of its Input/Output (I/O) greatly simplifies circuit design and board layout. All I/O pins are available for use as digital inputs and outputs for both the CPU and digital peripherals. In addition, all I/O pins can generate an interrupt.

What this means for a real world application is that, unlike classic MCUs, there are no pins pre-assigned for SPI, UART or ADC functions. The user is free to choose the I/Os he or she wishes to connect those functions to.

The *almost* part is because there are some pins that are intrinsically associated to some functions that need low path resistance and capacitance, such as USB or OpAmps ports. When these functions are not enabled, their pins may be used for any other purpose though.

The I/O supports 8 different drive modes illustrated on figure 3.4.

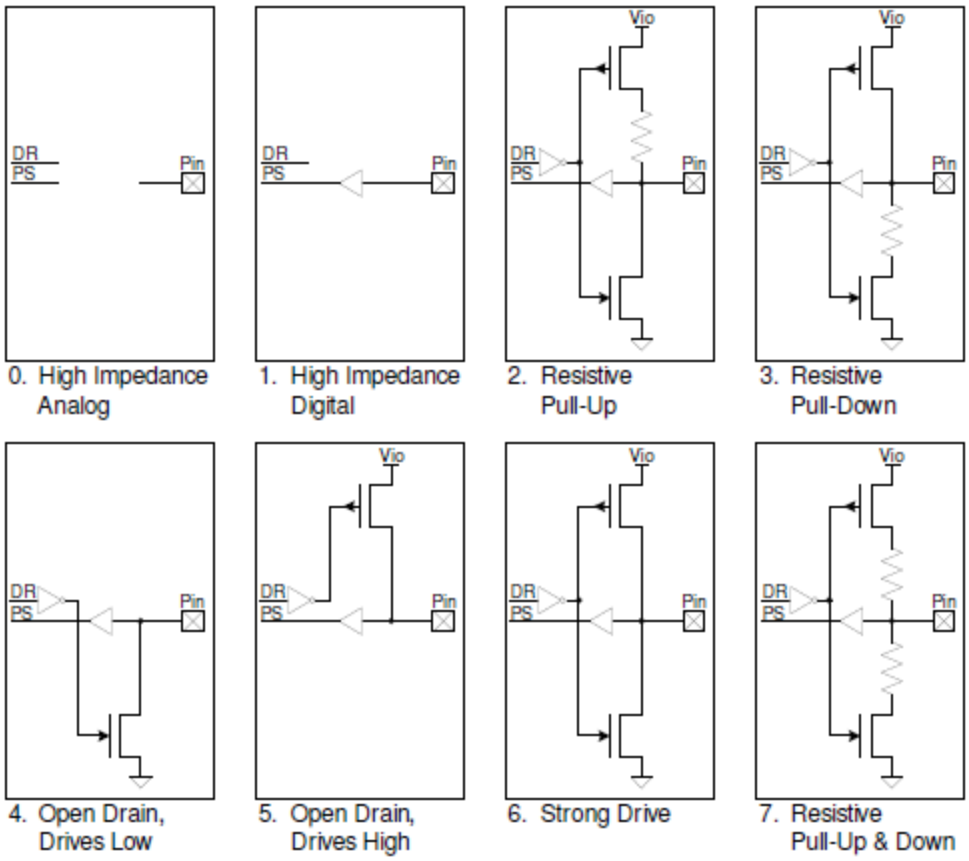


Figure 3.4: I/O Drive Modes

There are two types of I/Os: General Purpose Input/Output (GPIO) and Special Input/Output (SIO). Both GPIO and SIO provide similar digital functionality. The primary differences are their analog capability and drive strength.

The PSoC I/O system has these features, depending on the pin type:

- Supported by both GPIO and SIO pins:
  - User programmable I/O state and drive mode on device reset.
  - Flexible drive modes.
  - Support level and edge interrupts on pin basis.
  - Slew rate control.
  - Supports CMOS and low voltage TTL input thresholds.
  - Separate port read and write registers.
  - Separate I/O supplies and voltages for up to four groups of I/O.
- Provided only on the GPIO pins:
  - Supports LCD drive.
  - Supports CapSense.
  - Supports JTAG interface.

  - Analog input and output capability.
  - $8mA$ sink and $4mA$ source current.
  - Ports can be configured to support EMIF address and data.

- Provided only on SIO pins:
  - Hot swap capability ($5V$ tolerance at any operating `Vdd`).
  - Single enable and differential input with programmable threshold.
  - Regulated output voltage level option.
  - Overvoltage tolerance up to $5.5V$.
  - Higher drive strength than GPIO: $25mA$ sink and $4mA$ source current.

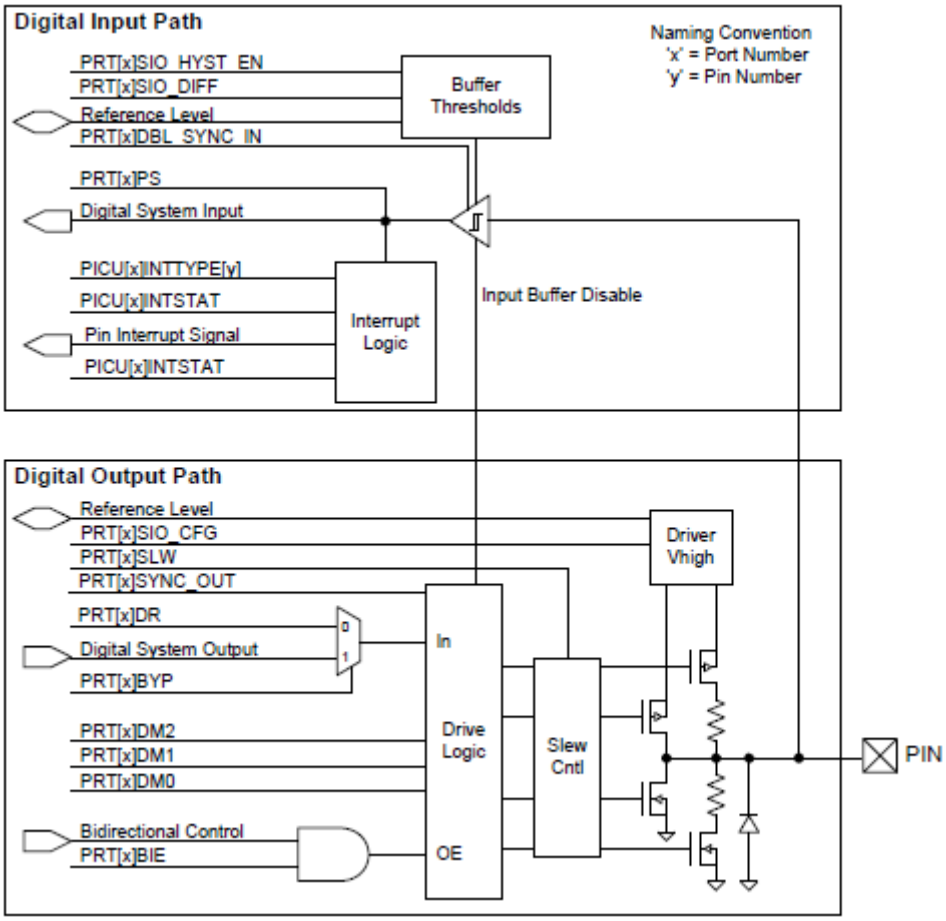


Figure 3.5: SIO block diagram

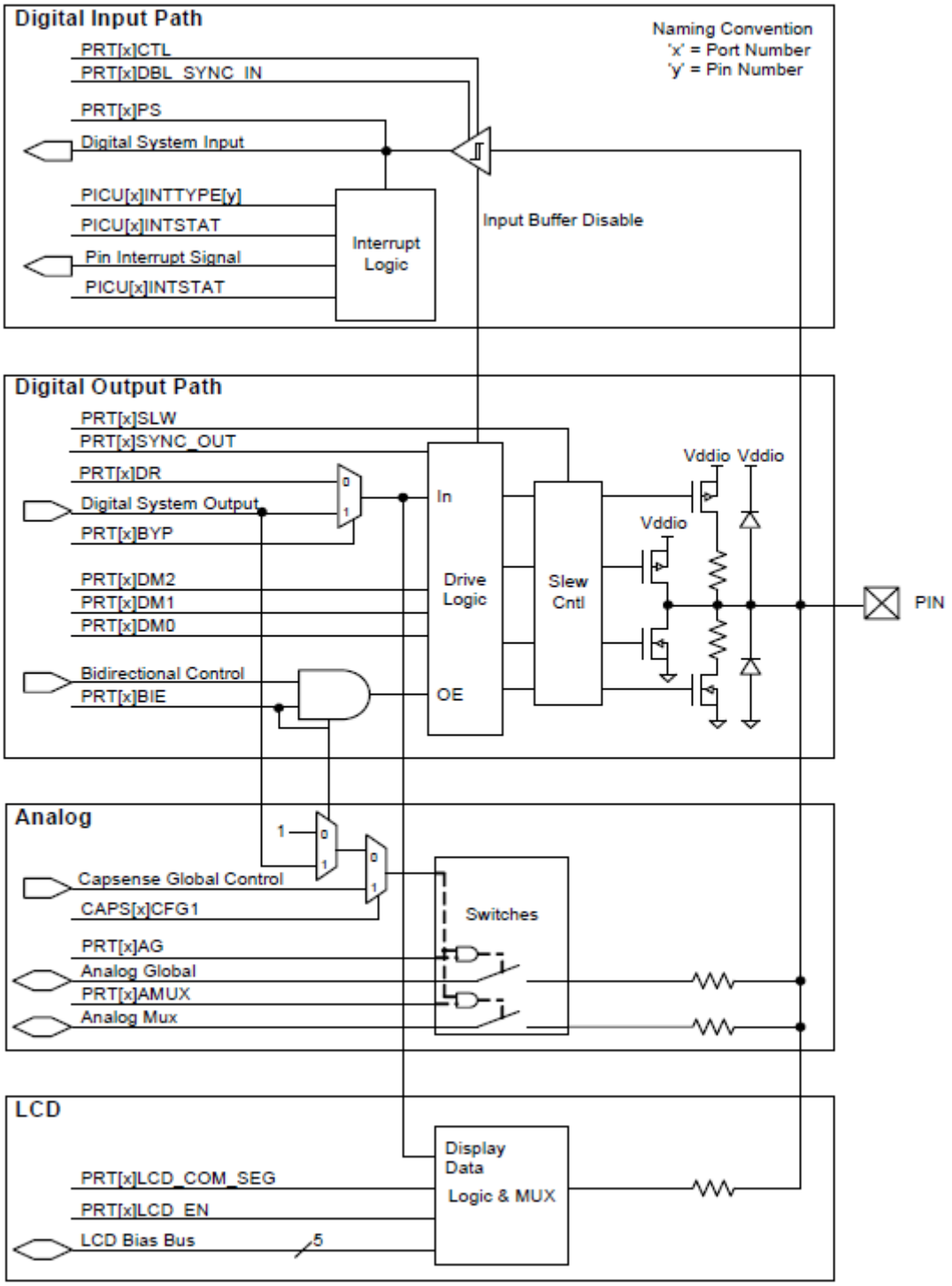


Figure 3.6: GPIO block diagram

3.2 Analog Subsystem

3.2.1 Analog to Digital Converters

Delta-Sigma ADC

Both PSoC3 and PSoC5 feature a Delta-Sigma Analog to Digital Converter [19] that provides a low-power, low-noise front end for precision measurement applications. It can be used in a wide range of applications, depending on resolution, sample rate, and operating mode. It can produce 16-bit audio; high speed and low resolution for communications processing; and high-precision 20-bit low-speed conversions for sensors such as strain gauges, thermocouples, and other high-precision sensors.

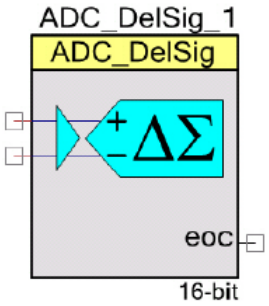


Figure 3.7: Delta-Sigma ADC component symbol

Delta-sigma converters use oversampling to spread the quantization noise across a wider frequency spectrum. This noise is shaped to move most of it outside the input signal's bandwidth. An internal low-pass filter is used to filter out the noise outside the desired input signal bandwidth. This makes delta-sigma converters good for both high-speed medium-resolution (8 to 16 bits) applications, and low-speed high-resolution (16 to 20 bits) applications. The sample rate can be adjusted between 10 and 384000 samples per second, depending on mode and resolution. Choices of conversion modes simplify interfacing to single streaming signals such as audio, or multiplexing between multiple signal sources.

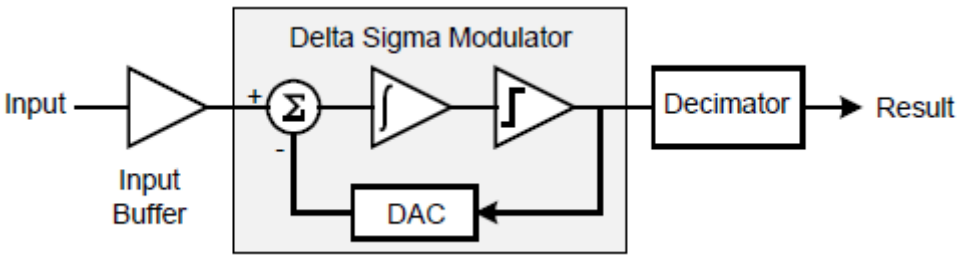


Figure 3.8: Delta-Sigma ADC block diagram

Features

- Selectable resolutions, 8 to 20 bits (device dependent)
- Eleven input ranges for each resolution

- Sample rate 10 sps to 384 ksps
- Operational modes:
  - Single sample
  - Multi-sample
  - Continuous mode
  - Multi-sample (Turbo)
- High input impedance input buffer
- Selectable input buffer gain (1, 2, 4, 8) or input buffer bypass
- Multiple internal or external reference options
- Automatic power configuration
- Up to four run-time Analog to Digital Converter (ADC) configurations

ADC SAR

The ADC Successive Approximation Register (SAR) component [1] provides high-speed (maximum 1-Msps sampling), medium-resolution (12 bits maximum), analog-to-digital conversion.

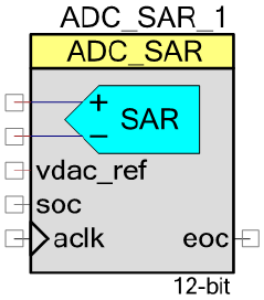


Figure 3.9: ADC SAR component symbol

An input analog signal is sampled and compared with the output of a Digital to Analog Converter (DAC) using a binary search algorithm to determine the conversion bits in succession from MSB to LSB.

Features

- 12-bit resolution at up to 1 Msps maximum
- Four power modes
- Selectable resolution and sample rate
- Single-ended or differential input

This component is exclusive of PSoC5LP devices.

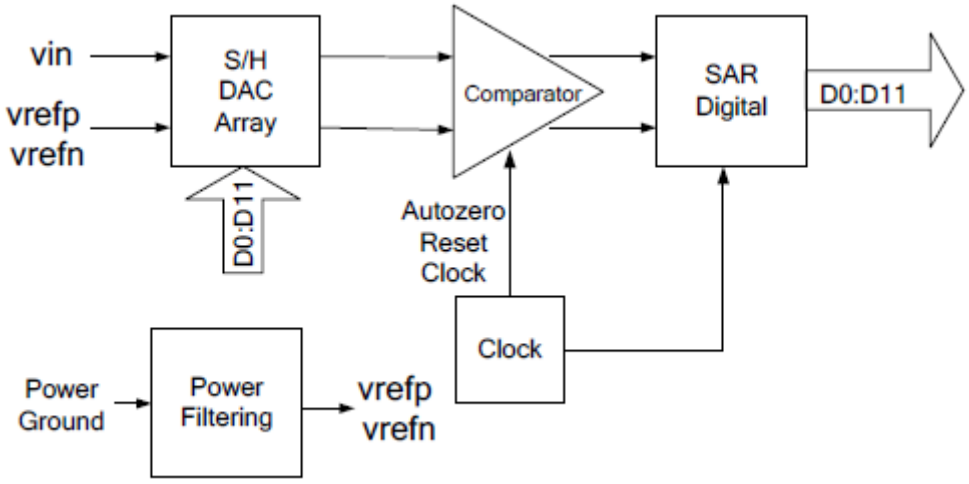


Figure 3.10: ADC SAR block diagram

3.2.2 Digital to Analog Converters

8-bit Current DAC

The IDAC8 component [3] is an 8-bit current output DAC. The output can source or sink current in three ranges. It can be controlled by hardware, software, or by a combination of both hardware and software.

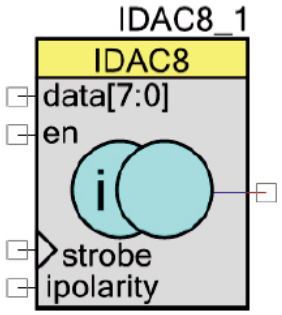


Figure 3.11: IDAC8 component symbol

IDAC8 functionality is implemented using the PSoC VIDAC block. This block is an 8 bit digital analog converter capable of either voltage or current output. The output from the IDAC8 is single-ended. The functional block diagram is shown in figure 3.12.

Features

- Three ranges: 2040A, 255μA, and 31.875μA
- Current sink or source selectable
- Software- or clock-driven output strobe
- Data source may be CPU, DMA, or digital components

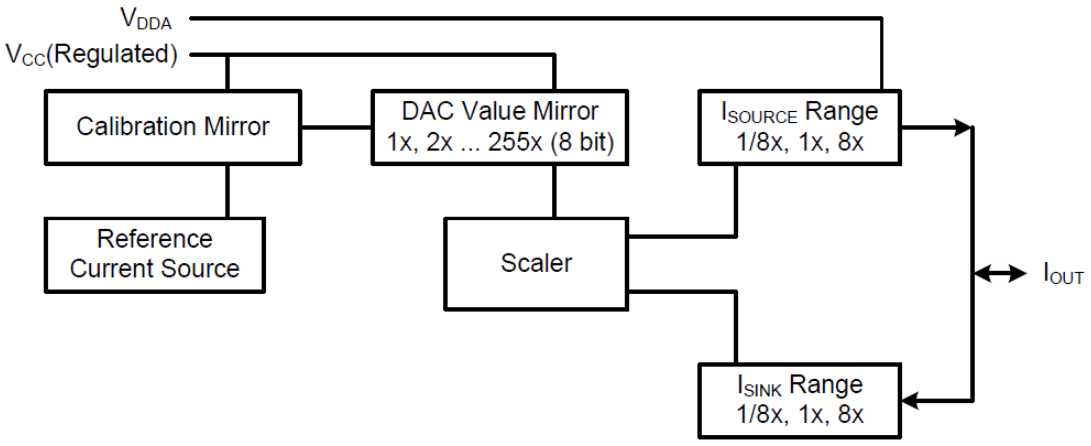


Figure 3.12: IDAC8 block diagram

8-bit Voltage DAC

The VDAC8 component [10] is an 8-bit voltage output DAC. The output range can be from 0 to $1.020V$ ($4mV/bit$) or from 0 to $4.08V$ ($16mV/bit$). The VDAC8 can be controlled by hardware, software, or a combination of both hardware and software.

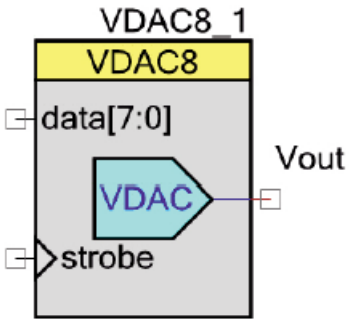


Figure 3.13: VDAC8 component symbol

Figure 3.14 shows the block diagram for the VDAC8 component.

Features

- Voltage output ranges: $1.020V$ and $4.080V$ full scale
- Software- or clock-driven output strobe
- Data source can be CPU, DMA, or digital components

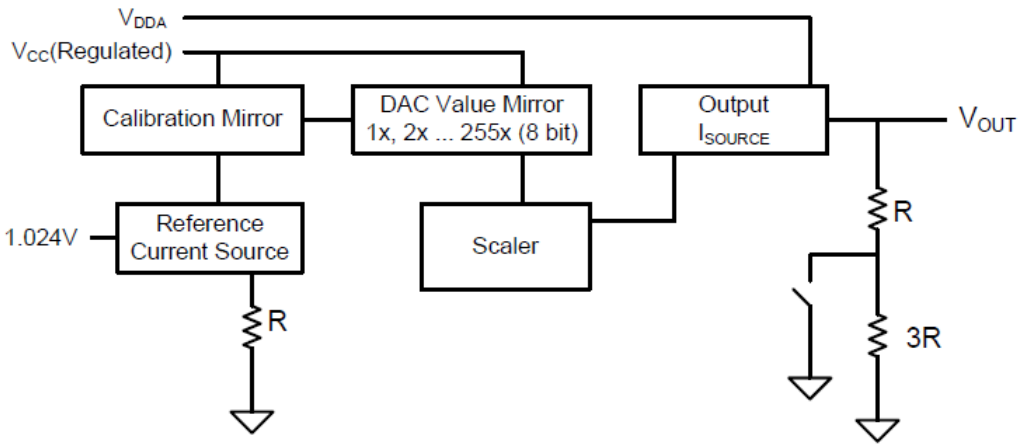


Figure 3.14: VDAC8 block diagram

3.2.3 Operational Amplifiers

The OpAmp component [4] provides a low-voltage, low-power operational amplifier and may be internally connected as a voltage follower. The inputs and output may be connected to internal routing nodes, directly to pins, or a combination of internal and external signals. The OpAmp is suitable for interfacing with high-impedance sensors, buffering the output of voltage DACs, driving up to $25mA$; and building active filters in any standard topology.

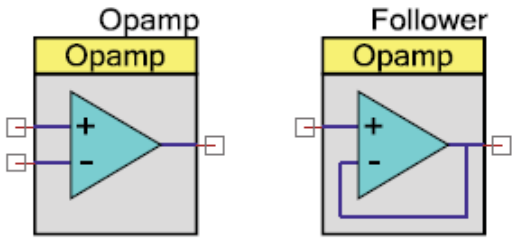


Figure 3.15: OpAmp component symbol

Features

- Follower or generic Operational Amplifier configuration
- Unity gain bandwidth $> 3.0MHz$
- Input offset voltage $2.0mV$ max
- Rail-to-rail inputs and output
- Output direct low resistance connection to pin
- $25mA$ output current
- Programmable power and bandwidth
- Internal connection for follower (saves pin)

There are 4 OpAmps on a PSoC. Each OpAmp is directly connected to specific GPIOs to lower the route resistance and capacitance besides saving routing resources. The user may use though the analog global routing buses at his or her discretion, and the specific GPIO pins may be used for any other purpose when the connected OpAmp is not enabled.

3.2.4 Switched Capacitors - Continuous Time block

The Switched Capacitors (SC) Continuous Time (CT) block [21] [23] is a general purpose block constructed of a rail-to-rail amplifier with arrays of switches, capacitors, and resistors. Register configurations select the block functional topology, power level, and bandwidth.

Features

- Multiple configurations:
  - Naked OpAmp
  - Continuous Time Unity Gain Buffer
  - Track and Hold Amplifier
  - Continuous Time Programmable Gain Amplifier
  - Continuous Time Trans Impedance Amplifier
  - Continuous Time Mixer
  - Sampled Mixer (Non-Return to Zero (NRZ) - Sample and Hold (S/H))
  - Delta-Sigma Modulator
- Routability to GPIO
- Routable reference selection
- Programmable power and bandwidth
- Sample and hold configuration

Block Diagram

Figure 3.16 illustrates the block diagram of the SC/CT block. In the sections below the *Programmable Gain Amplifier (PGA)* and the *Trans-Impedance Amplifier (TIA)* configurations, which might be some of the most useful for this project, are explained in deeper detail.

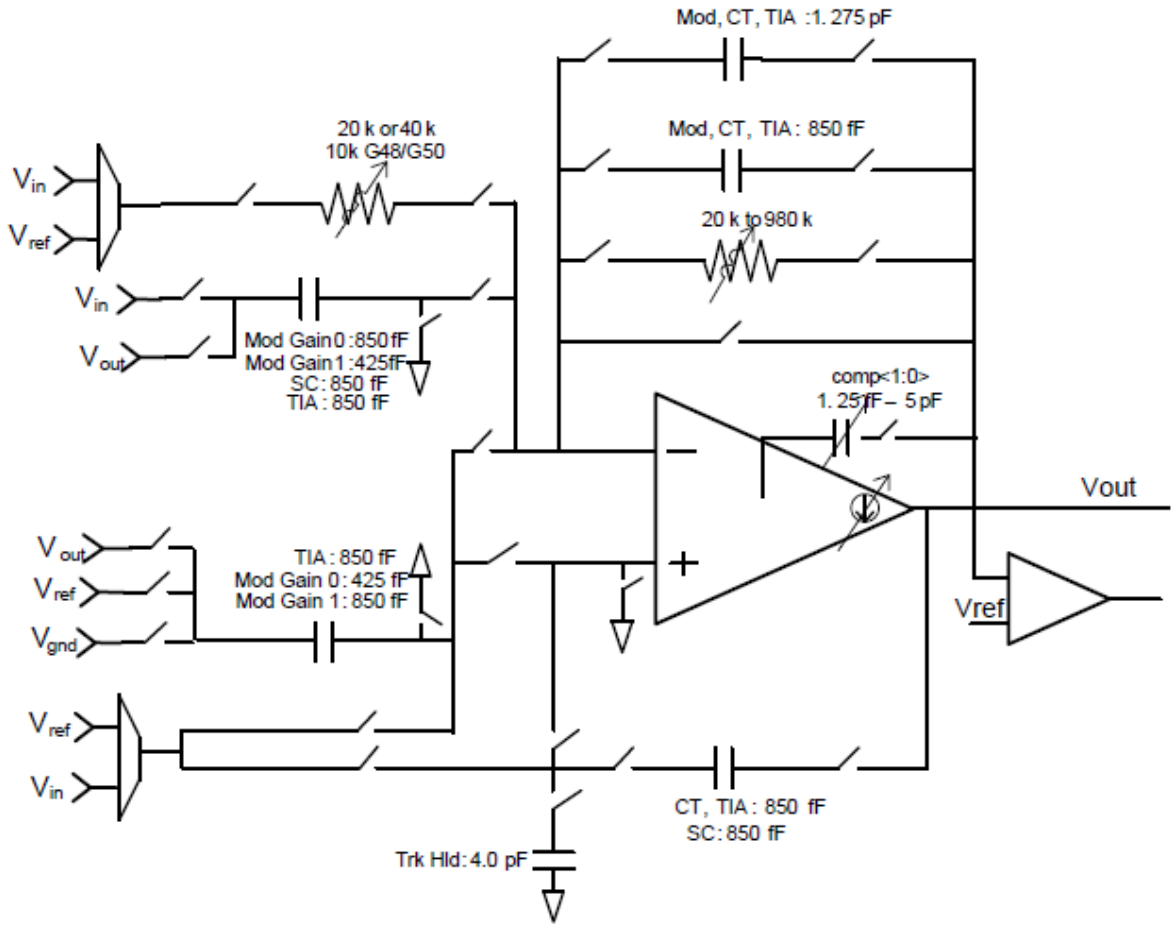


Figure 3.16: SC/CT block diagram

Programmable Gain Amplifiers

The PGA implements an OpAmp-based, non-inverting amplifier with user-programmable gain [5]. This amplifier has high input impedance, wide bandwidth and selectable input voltage reference.

The gain can be between 1 ($0dB$) and 50 ($+34dB$). This setting can be selected using its configuration window on the IDE or changed at run time using the provided API. The maximum bandwidth is limited by the gain-bandwidth product of the OpAmp and is reduced as the gain is increased.

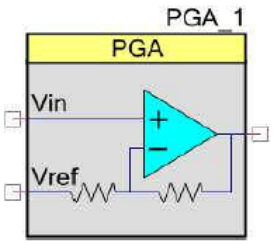


Figure 3.17: PGA component symbol

The PGA is used when an input signal has insufficient amplitude. A PGA can be put in front of a comparator, ADC, or mixer to increase the amplitude of the signal to these components. It can be used as a unity gain amplifier to buffer the inputs of lower impedance blocks, including Mixers or inverting PGAs. A unity gain PGA can also be used to buffer the output of a VDAC or reference.

Features

- Gain steps from 1 to 50
- High input impedance
- Selectable input reference
- Adjustable power settings

Trans-Impedance Amplifier

The TIA component provides an OpAmp-based current-to-voltage conversion amplifier with resistive gain and user-selected bandwidth [8].

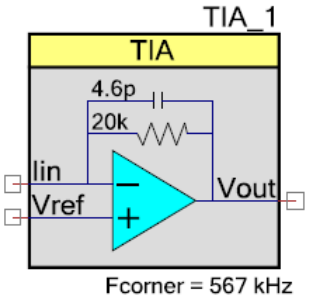


Figure 3.18: TIA component symbol

The TIA is used to convert an external current to a voltage. Typical applications include the measurement of sensors with current outputs such as photodiodes. The conversion gain of the TIA is expressed in ohms, with the available range between $20k\Omega$ and $1.0M\Omega$.

Current output sensors, such as photodiodes, often have substantial output capacitance. This requires shunt feedback capacitance in the TIA to guarantee stability. The TIA has a programmable feedback capacitor to meet this need and provide bandwidth limiting to reduce broadband noise.

Features

- Selectable conversion gain
- Selectable corner frequency
- Compensation for capacitive input sources
- Adjustable power settings
- Selectable input reference voltage

3.2.5 Comparators

The Comparator component [17] provides a hardware solution to compare two analog input voltages. The output can be sampled in software or digitally routed to another component. Three speed levels are provided to allow optimization for speed or power consumption.

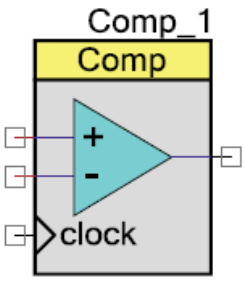


Figure 3.19: Comparator component symbol

The Comparator can provide a fast comparison between two voltages, compared to using an ADC. Although an ADC can be used with software to compare multiple voltage levels, applications requiring fast response or little software intervention are good candidates for this comparator. Some example applications include power supplies or simple translation from an analog level to a digital signal.

A common configuration is to create an adjustable comparator by connecting a voltage DAC to the negative input terminal as shown in figure 3.20.

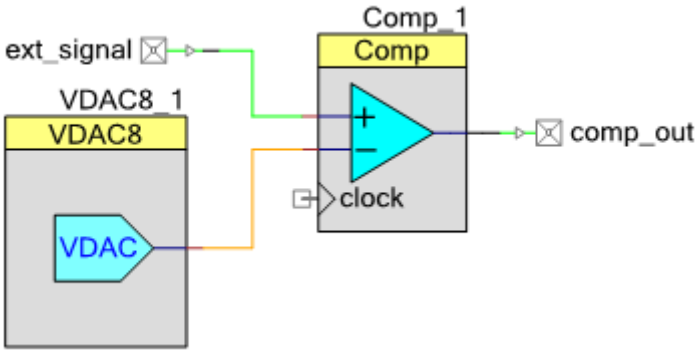


Figure 3.20: Comparator using analog threshold

3.2.6 Digital Filter Block

Some PSoC devices have a dedicated hardware Digital Filter Block (DFB) used to filter applications [21] [23]. The heart of DFB is a Multiply and Accumulate (MAC), which can do 24 bit * 24 bit multiply and 48 bit accumulate in one system clock cycle. In addition, there are data RAMs to store data and coefficients of digital filters.

Features

- Two 24-bit wide streaming data channels

- Two sets of data RAMs each that can store 128 words of 24-bit width each
- One interrupt and two DMA request channels
- Three Semaphore bits to interact with system software
- Data alignment and coherency protection support options for input and output samples

Description

The digital filter block (DFB) is a 24-bit fixed point, programmable limited scope Digital Signal Processing (DSP) engine. It is made up of four primary subfunctions as shown in the DFB block diagram in figure 3.21.

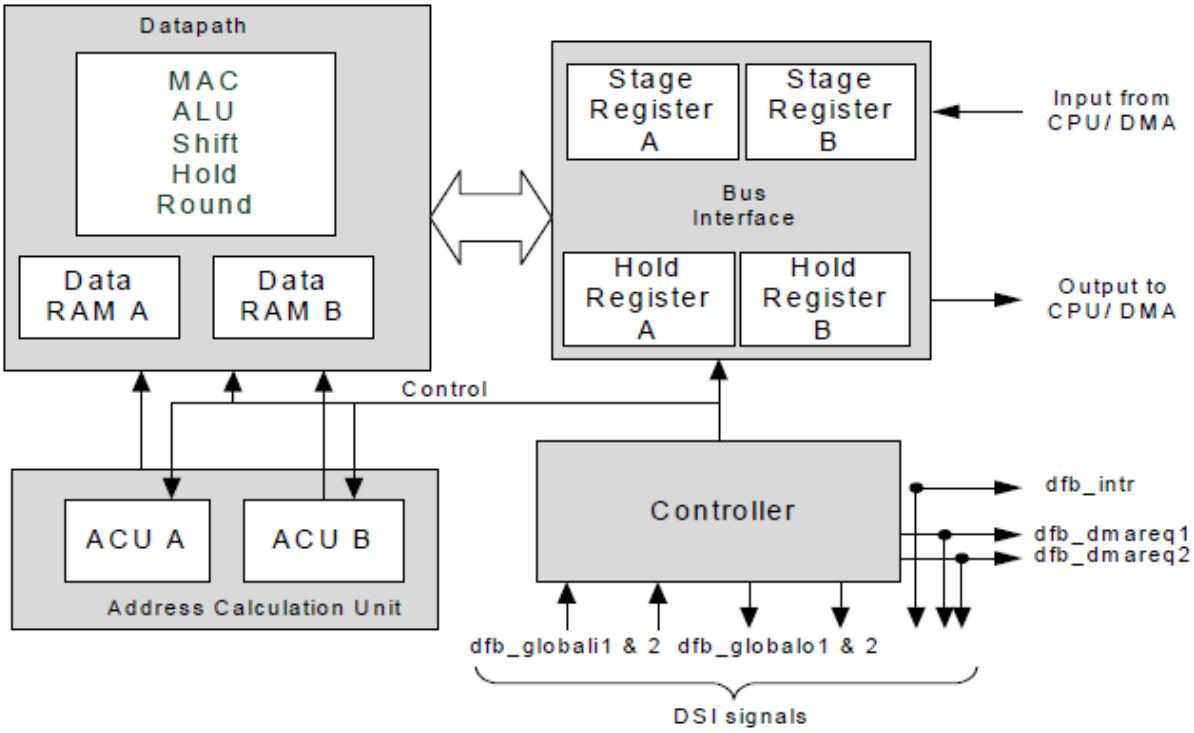


Figure 3.21: Digital Filter Block diagram

The Controller consists of a small amount of digital logic and memories. The memories in the controller are filled with assembled code that make up the data transform function the DFB is intended to perform.

The Datapath subblock is a 24-bit fixed point, numerical processor containing a Multiply and Accumulator (MAC), a multifunction Arithmetic Logic Unit (Arithmetic Logic Unit (ALU)), sample and coefficient and data RAM. The datapath block is the calculation unit inside the DFB.

The Address Calculation Units (ACUs) control the addressing of the two data RAMs in the datapath block. There are two (identical) ACUs, one for each RAM.

Bus Interface: These three subfunctions make up the core of the DFB block and are wrapped with a 32-bit DMA-capable AHB-Lite Bus Interface with Control/Status registers.

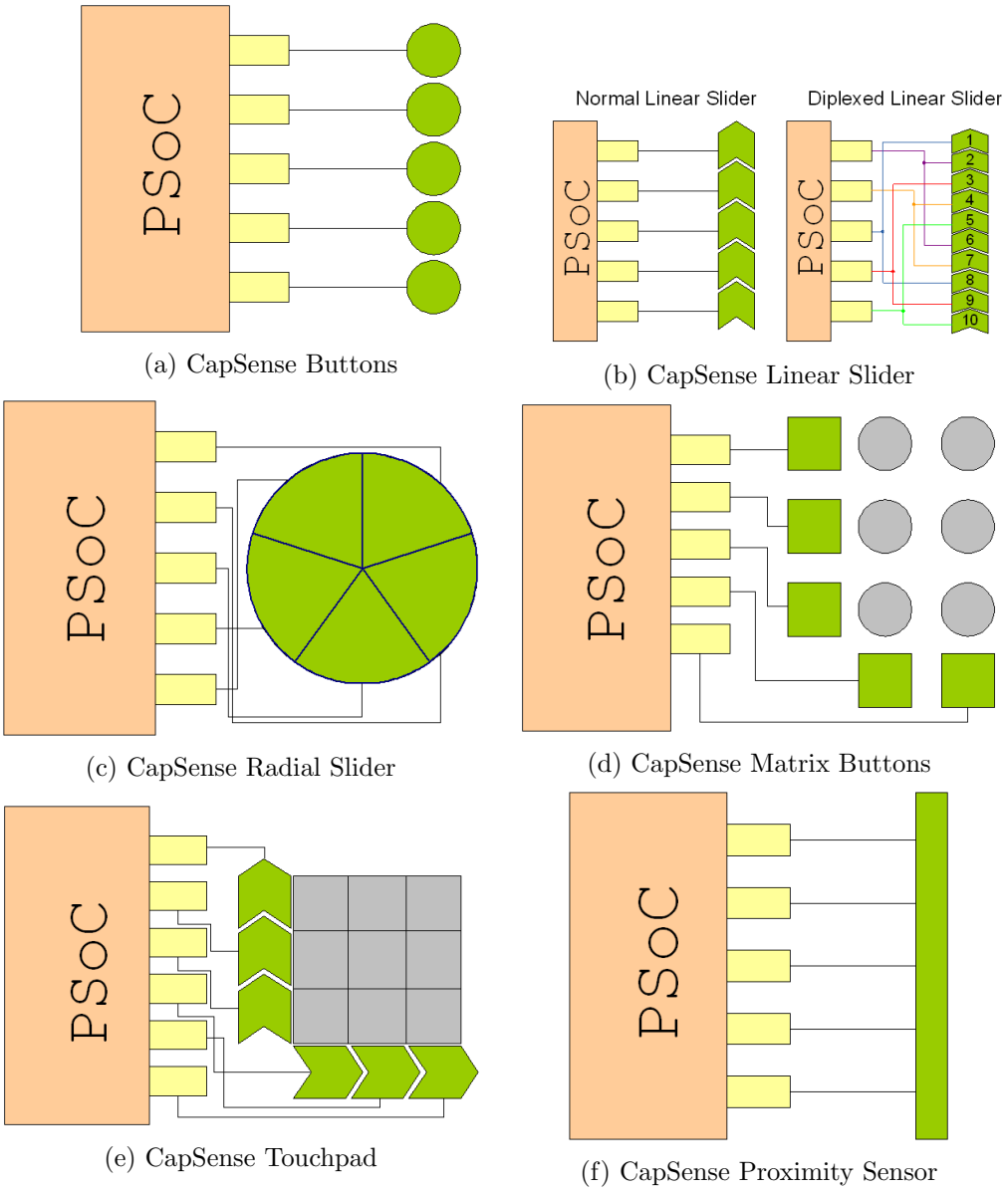
3.2.7 Capacitive Sensing

CapSense is the trademark under which Cypress sell its capacitive sensing technology. It uses a Delta-Sigma Modulator to measure capacitance in applications such as touch sense buttons, sliders, touchpad, and proximity detection (figure 3.22).

The capacitance metering is based on a *relaxation oscillator*. The capacitance to be sensed forms a portion of the oscillator's RC circuit. Basically the technique works by charging the unknown capacitance with a known current.

The equation of state for a capacitor is $i = C\frac{dv}{dt}$. This means that the capacitance equals the current divided by the rate of change of voltage across the capacitor.

The capacitance can be calculated by measuring the charging time required to reach the threshold voltage (of the relaxation oscillator), or equivalently, by measuring the oscillator's frequency. Both of these are proportional to the RC time constant of the oscillator circuit.



(a) CapSense Buttons

(b) CapSense Linear Slider

(c) CapSense Radial Slider

(d) CapSense Matrix Buttons

(e) CapSense Touchpad

(f) CapSense Proximity Sensor

Figure 3.22: CapSense inputs

3.3 Digital Subsystem

3.3.1 Fixed function blocks

Timer, Counter and PWM

Timer blocks in PSoC devices are 8/16 bits and configurable to act as Timer, Counter, or Pulse Width Modulation (PWM) blocks that play important roles in embedded systems. PSoC devices give a maximum of four instances of the block. If additional blocks are required, they can be configured in the UDBs (section 3.3.2) using PSoC Creator (section 3.4). Timer blocks have various clock sources and are connected to the GPIO though the Digital Signal Interconnect (DSI) (section 3.3.3).

Timewheel

Central Timewheel

The Central Timewheel (CTW) is a 1-kHz, free-running, 13-bit counter clocked by the ILO. The CTW is always available, except in hibernate mode.

The main functions of the CTW are:

- Buzzing during sleep mode.
- Waking up the device from a low-power mode.
- Watchdog Timer (WDT).
- General timing purposes.

Fast Timewheel

The Fast Timewheel (FTW) is a 100-kHz, 5-bit counter clocked by the ILO, which can also be used to wake the system. The FTW settings are programmable; the counter automatically resets when the terminal count is reached. It enables flexible, periodic wakeups of the CPU at a higher rate than the rate allowed using the CTW.

Watchdog Timer

The Watchdog Timer (WDT) circuit automatically reboots the system in the event of an unexpected execution path. This timer must be serviced periodically. If not, the CPU resets after a specified period of time.

The WDT has the following features:

- Protection settings to prevent accidental corruption of the WDT
- Optionally-protected servicing (feeding) of the WDT
- A configurable low-power mode to reduce servicing requirements during sleep mode
- A status bit for the watchdog event that shows the status even after a watchdog reset

3.3.2 UDB blocks

Universal Digital Blocks (UDBs) allow for custom implementation of logic peripherals capable of communicating with the CPU [21] [23]. Circuits can be described either using Verilog or schematics capture.

Many users of PSoC may have experience describing hardware for CPLDs and Field-Programmable Gate Arrays (FPGAs). PSoC supports Verilog for Programmable Logic Devices (PLDs), however, these PLDs are far less powerful than full-featured CPLDs or FPGAs, and most designs are too large to be implemented directly on PLDs [16] .

Each UDB block provides a datapath that, working together with PLDs, removes this limitation by performing calculations, comparisons, and other data management tasks.

Features

- For optimal flexibility, each UDB contains several components:
  - ALU-based 8-bit Datapath (DP) with an 8-word instruction store and multiple registers and FIFOs
  - Two PLDs, each with 12 inputs, eight product terms and four macrocell outputs
  - Control and status modules
  - Clock and reset modules
- PSoC chips contain an array of up to 24 UDBs
- Flexible routing through the UDB array
- Portions of UDBs can be shared or chained to enable larger functions
- Flexible implementations of multiple digital functions, including timers, counters, PWM (with dead band generator), UART, I^2C, SPI, and CRC generation/checking

Functional description

Figure 3.23 illustrates the UDB as a construct containing a pair of basic PLD logic blocks, a datapath, and control, status, clock and reset functions.

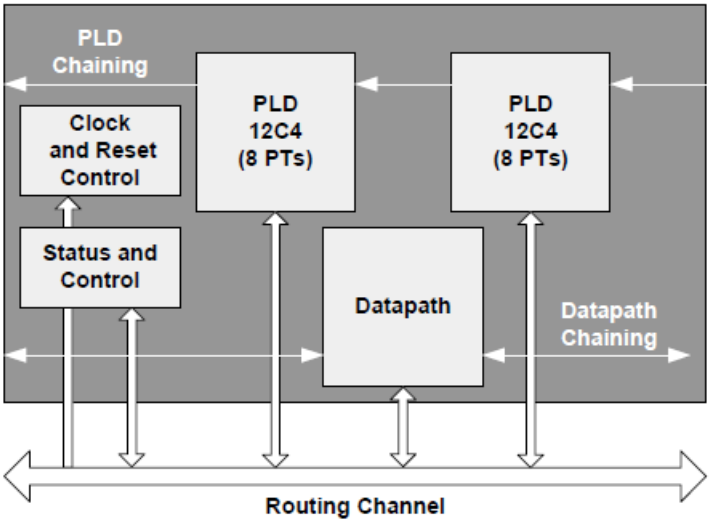


Figure 3.23: UDB block diagram

PLDs

There are two "12C4" PLDs in each UDB [21] [23]. PLD blocks, shown in figure 3.24, can be used to implement state machines, perform input or output data conditioning, and to create lookup tables (Look-Up Tables (LUTs)). PLDs may also be configured to perform arithmetic functions, sequence the datapath, and generate status. General purpose RTL can be synthesized and mapped to the PLD blocks. This section presents an overview of the PLD design.

A PLD has 12 inputs which feed across eight Product Term (PT) in the AND array. In a given product term, the true (T) or complement (C) of the input can be selected. The output of the PTs are inputs into the OR array. The 'C' in 12C4 indicates that the OR terms are constant across all inputs, and each OR input can programmatically access any or all of the PTs. This structure gives maximum flexibility and ensures that all inputs and outputs are permutable.

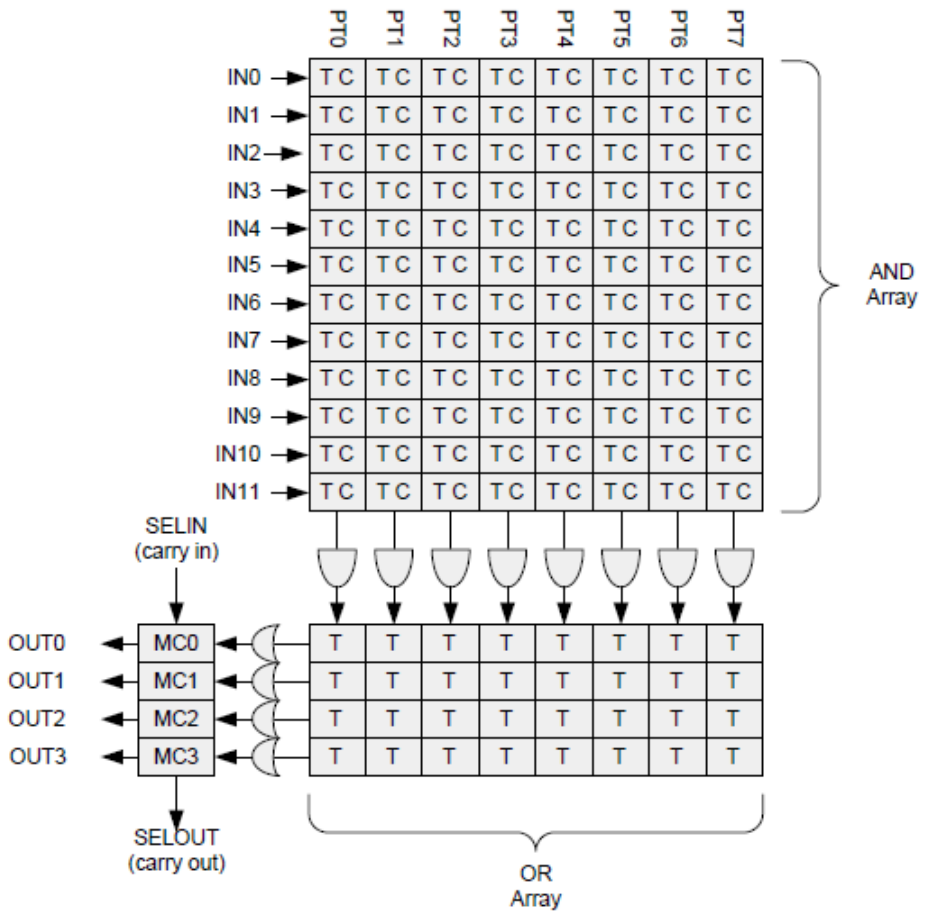


Figure 3.24: PLD 12C4 structure

Macrocells The macrocell architecture is shown in figure 3.25. The output drives the routing array, and can be registered or combinational. The registered modes are D Flip-Flop with true or inverted input, and Toggle Flip-Flop on input high or low. The output register can be set or reset for purposes of initialization, or asynchronously during operation under control of a routed signal [21] [23].

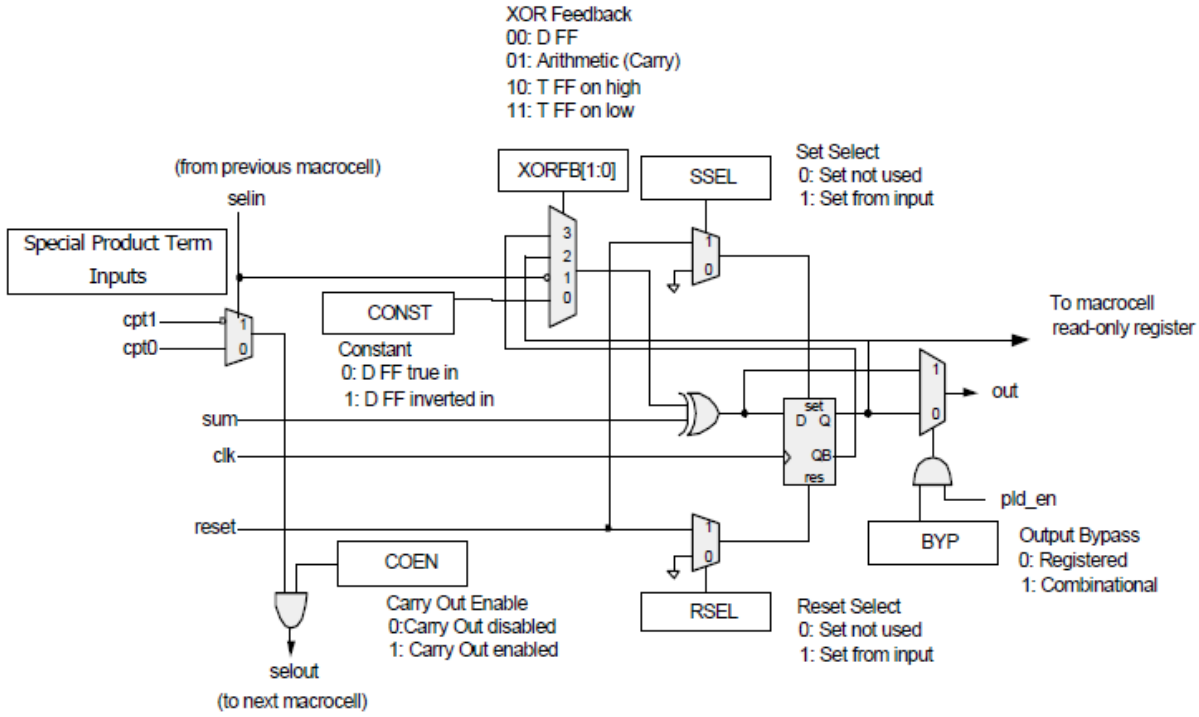


Figure 3.25: Macrocell architecture

PLD Macrocell Read Only Register In addition to driving the routing array, the outputs of the macrocells from both PLDs are mapped into the address space as an 8-bit read only register, which can be accessed by the CPU or DMA.

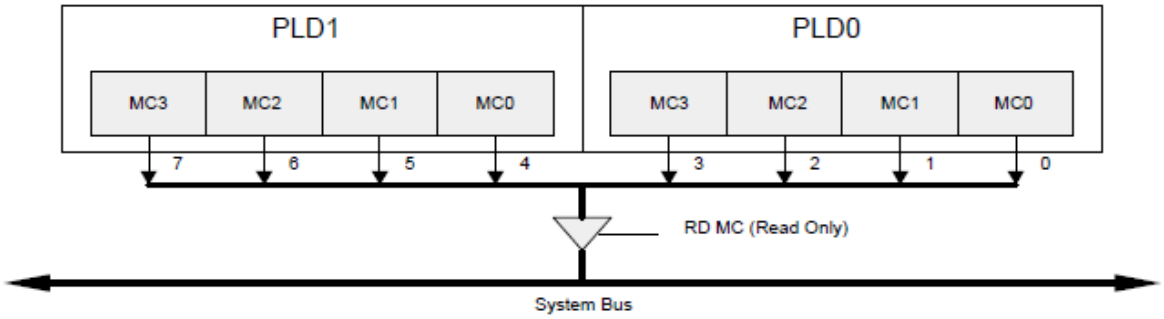


Figure 3.26: PLD Macrocell Read-Only Register

PLD Carry Chain PLDs are chained together in UDB address order. As shown in figure 3.27 the carry chain input "selin" is routed from the previous UDB in the chain, through each macrocell in both of the PLDs, and then to the next UDB as the carry chain out "selout". To support the efficient mapping of arithmetic functions, special product terms are generated and used in the macrocell in conjunction with the carry chain.

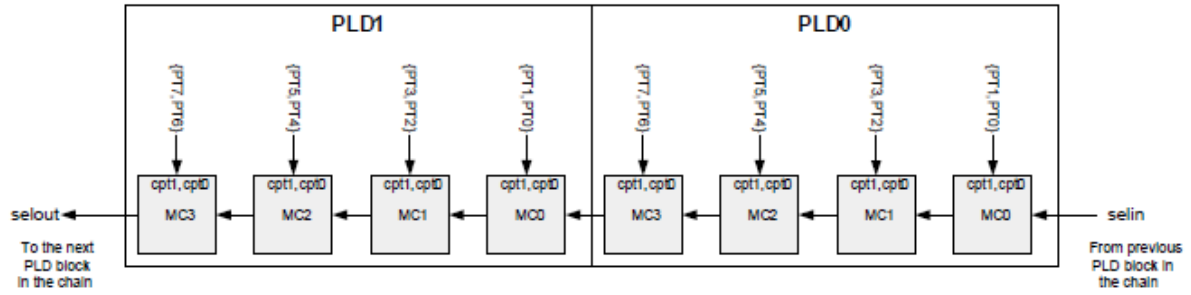


Figure 3.27: PLD Carry Chain and Special Product Terms inputs

Datapath

The datapath, shown in 3.28, contains an 8-bit single-cycle ALU, with associated compare and condition generation circuits [21] [23]. A datapath may be chained with datapaths in neighboring UDBs to achieve higher precision functions. The datapath includes a small RAM-based control store, which can dynamically select the operation to perform in a given cycle.

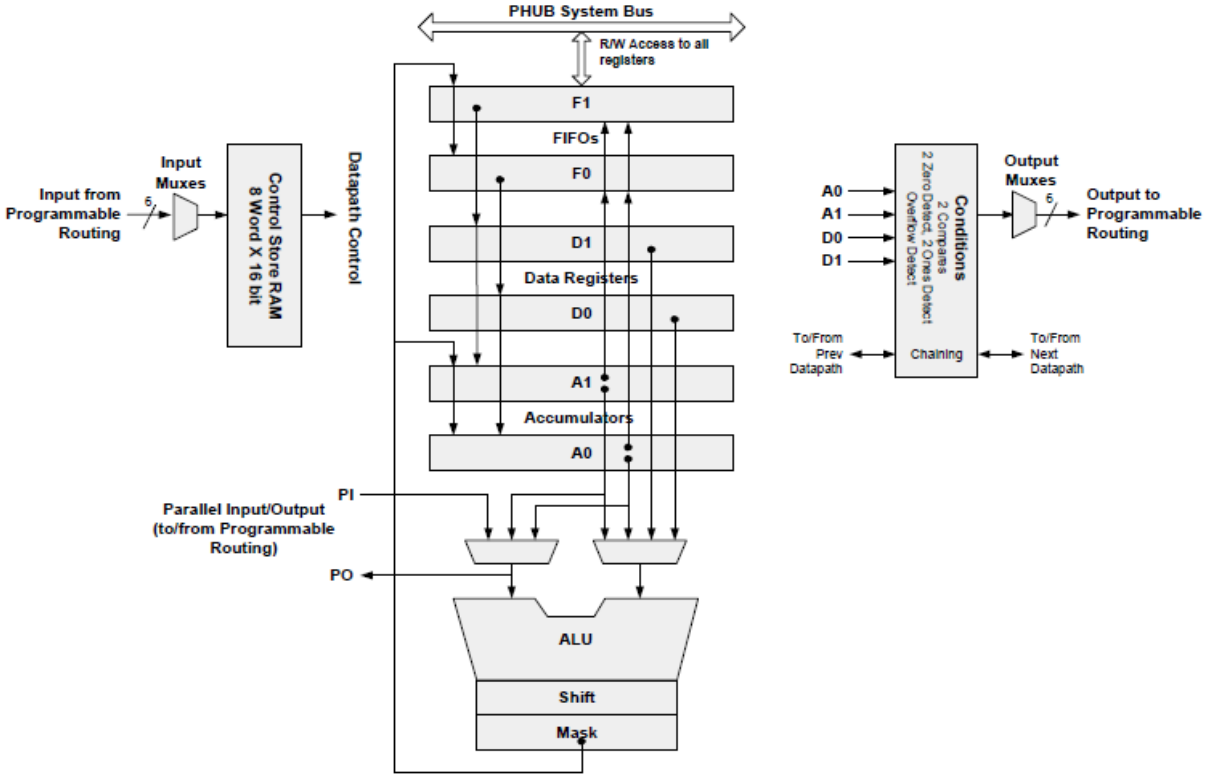


Figure 3.28: Datapath Top Level

The datapath is optimized to implement typical embedded functions such as timers, counters, PWMs, Pseudo Random Sequence (PRS), Cyclic Redundancy Check (CRC), shifters and dead band generators. The addition of add and subtract functions allow support for digital delta-sigma operations.

The following sections present an overview description of key datapath features. All of these

features are highly tweakable, for full documentation please refer to PSoC Technical Reference Manuals (TRMs) [21] [23].

Dynamic Configuration Dynamic configuration is the ability to change the datapath function and interconnect on a cycle-by-cycle basis, under sequencer control. This is implemented using the configuration RAM, which stores eight unique configurations. The address input to this RAM can be routed from any block connected to the routing fabric, most typically PLD logic, I/O pins, or other datapaths.

ALU The ALU can perform eight general-purpose functions:

- Increment
- Decrement
- Add
- Subtract
- AND
- OR
- XOR
- PASS

Function selection is controlled by the configuration RAM on a cycle-by-cycle basis. Independent shift (left, right, nibble swap) and masking operations are available at the output of the ALU.

Conditionals Each datapath has two comparators, with bit masking options, which can be configured to select a variety of datapath register inputs for comparison. Other detectable conditions include all zeros, all ones, and overflow. These conditions form the primary datapath output selects to be routed to the digital routing fabric or inputs to other functions.

Built in CRC/PRS The datapath has built-in support for single-cycle Cyclic Redundancy Check (CRC) computation and Pseudo Random Sequence (PRS) generation of arbitrary width and arbitrary polynomial specification. To achieve longer than 8-bit CRC/PRS widths, signals may be chained between datapaths.

This feature is controlled dynamically, and therefore can be interleaved with other functions.

Variable MSB The most significant bit of an arithmetic and shift function can be programmatically specified. This supports variable width CRC/PRS functions and, in conjunction with ALU output masking, can implement arbitrary width timers, counters, and shift blocks.

Input/Output FIFOs Each datapath contains two 4-byte First Input, First Output (FIFO) buffers, which can be individually configured for direction as an input buffer (CPU or DMA writes to the FIFO, datapath internals read the FIFO), or an output buffer (datapath internals write to the FIFO, the CPU or DMA reads from the FIFO). These FIFOs generate status that can be routed to interact with sequencers, interrupt, or DMA requests.

Chaining The datapath can be configured to chain conditions and signals with neighboring datapaths. Shift, carry, capture, and other conditional signals can be chained to form higher precision arithmetic, shift, and CRC/PRS functions.

Time Multiplexing In applications that are oversampled, or do not need the highest clock rates, the single ALU block in the datapath can be efficiently shared between two sets of registers and condition generators. ALU and shift outputs are registered and can be used as inputs in subsequent cycles. Usage examples include support for 16-bit functions in one (8-bit) datapath, or interleaving a CRC generation operation with a data shift operation.

Datapath Inputs The datapath has three types of inputs: configuration, control, and serial and parallel data. The configuration inputs select the control store RAM address. The control inputs load the data registers from the FIFOs and capture accumulator outputs into the FIFOs. Serial data inputs include shift in and carry in. A parallel data input port allows up to eight bits of data to be brought in from routing.

Datapath Outputs There are a total of 16 signals generated in the datapath. Some of these signals are conditional signals (for example, compares), some are status signals (for example, FIFO status), and the rest are data signals (for example, shift out).

These 16 signals are multiplexed into the six datapath outputs and then driven to the routing matrix. By default the outputs are single synchronized (pipelined). A combinational output option is also available for these outputs.

Control and Status registers

A high level view of the Status and Control module is shown in figure 3.29. The Control register drives into the routing to provide firmware control inputs to UDB operation. The Status register read from routing provides firmware a method of monitoring the state of UDB operation [21] [23].

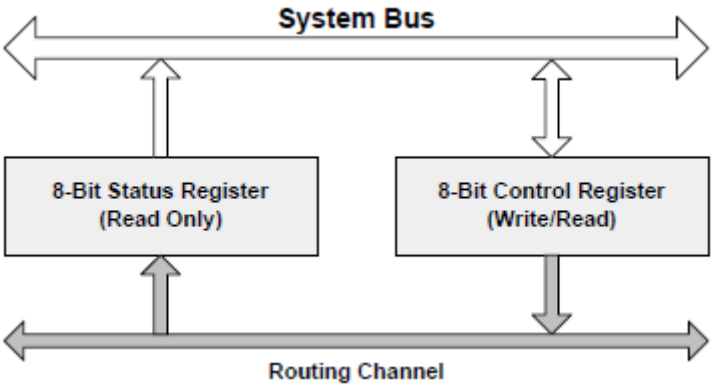


Figure 3.29: Status and Control registers

A more detailed view of the Status and Control module is shown in figure 3.30. The primary purpose of this block is to coordinate CPU firmware interaction with internal UDB operation. However, due to its rich connectivity to the routing matrix, this block may be configured to perform other functions.

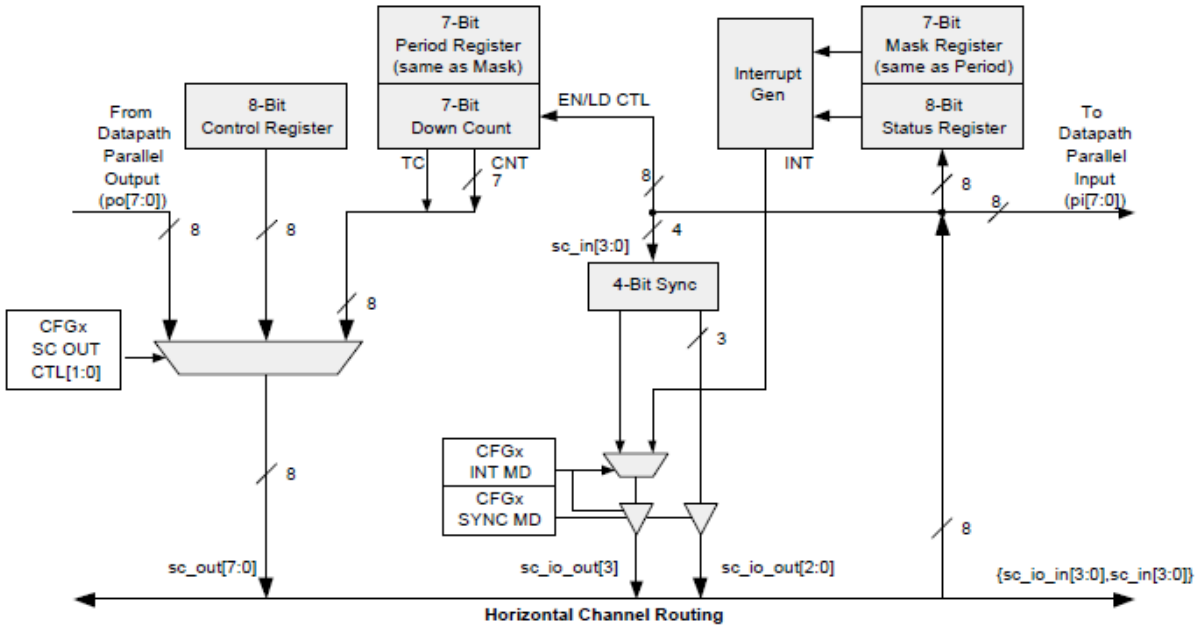


Figure 3.30: Status and Control module

Modes of operation include:

Status Input
The state of routing signals can be input and captured as status and read by the CPU or DMA.

Control Output
The CPU or DMA can write to the control register to drive the state of the routing.

Parallel Input
To datapath parallel input.

Parallel Output
From datapath parallel output.

Counter Mode
In this mode, the control register operates as a 7-bit down counter with programmable period and automatic reload. Routing inputs can be configured to control both the enable and reload of the counter. When this mode is enabled, control register operation is not available.

Sync Mode
In this mode, the status register operates as a 4-bit double synchronizer. When this mode is enabled, status register operation is not available.

Reset and Clock control

The primary function of the reset and clock block is to select a clock from the available global system clocks or bus clock for each of the PLDs, the datapath, and the status and control block [21] [23]. It also supplies dynamic and firmware-based resets to the UDB blocks.

There are a total of 10 clocks that can be selected for each UDB component: 8 global digital clocks, bus clock, and the selected external clock (`ext clk`). Any of the routed input signals (`rc_in`) can be used as either a level sensitive or edge sensitive enable. The reset function of this block provides a routed reset for the PLD blocks and Status and Control counter, and a firmware reset capability to each block to support reconfiguration.

System Bus Interface

UDB registers have dual access modes [21] [23]:

- System bus access, where the CPU or DMA is reading or writing a UDB register.
- UDB internal access, where the UDB function is updating or using the contents of a register.

3.3.3 Digital Signal Interconnect

Universal Digital Blocks (UDBs) are organized in the form of a two-dimensional array with programmable interconnect provided by the Digital Signal Interconnect (DSI) [21] [23]. In addition to connecting UDB components, the DSI routing also provides connection between other hardware resources on the device, such as I/O pins, interrupts, and fixed function blocks.

Figure 3.31 illustrates the programmable digital architecture for PSoC.

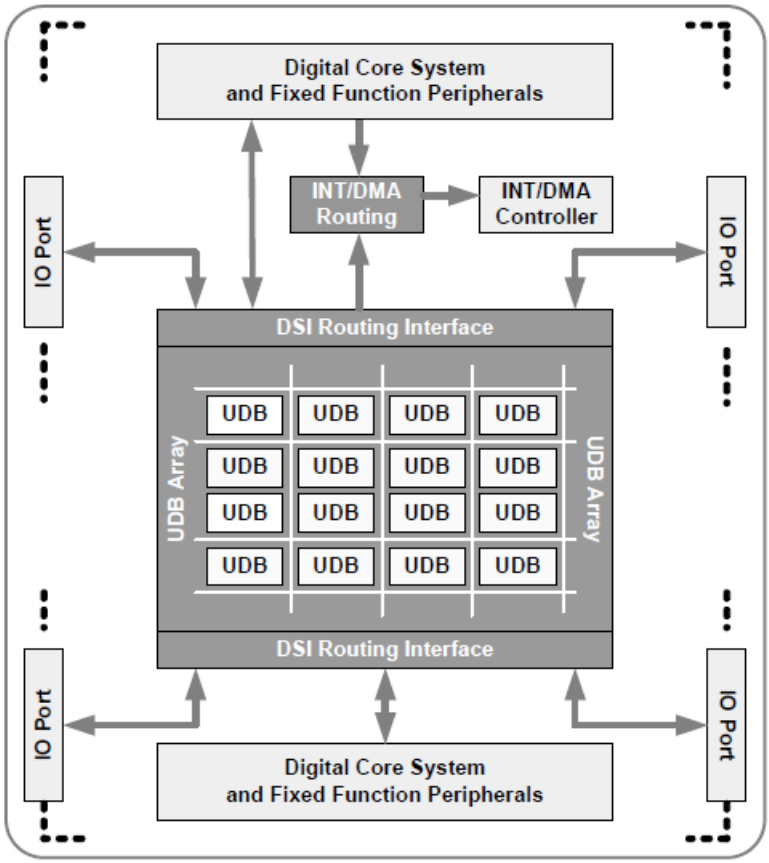


Figure 3.31: Programmable Digital Architecture

The main components of this system are:

UDB Array
UDB blocks are arrayed within a matrix of programmable interconnect. UDB pairs consisting of 2 UDBs are the basic building blocks of the UDB array. UDB pairs are tiled to create an array. UDB pairs can connect with neighboring UDB pairs in seamless fashion.

DSI
Routing interface tiled at top and bottom of UDB array core. Provides general purpose programmable routing between device peripherals, including UDBs, I/Os and fixed function blocks.

System Interface (not shown)
Built in 8/16-bit bus interface with parallel access to all registers to support fast configuration. Also provides clock distribution and clock gating functionality.

3.4 PSoC Creator

PSoC Creator is the IDE provided by Cypress to design and code for PSoC3 and PSoC5LP devices. It is provided for free.

PSoC Creator integrates several development tools:

Schematics capture tool to provide the user with a friendly interface to describe his or her designs in a drag-and-drop fashion.

Firmware development tool with code completion and syntax highlighting. *

Debugging tool with memory scopes, breakpoints...

Verilog tool for the description of custom logic.

Configuration of System-Wide Resources (I/Os, clocks, DMA...) through a graphical user interface.

* Keil μVision IDE is also available, with PSoC Creator automating the task of exporting all design information and sources.

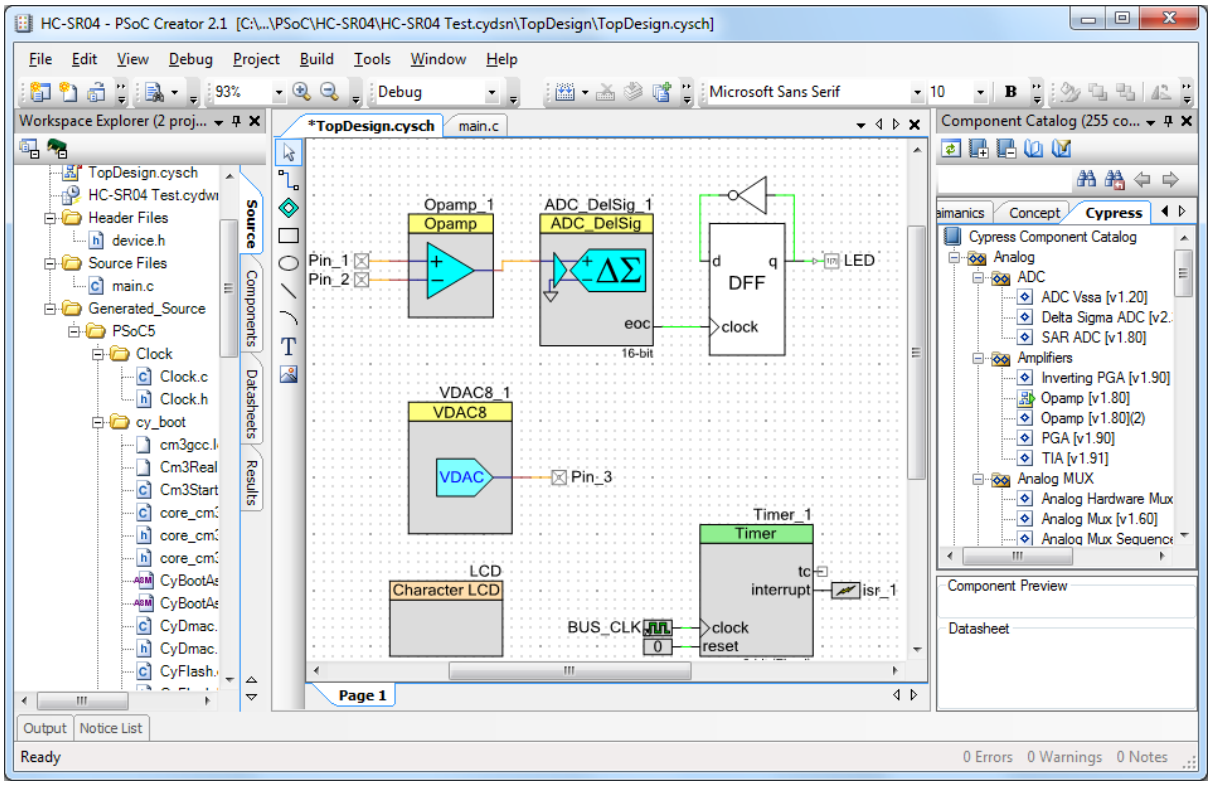


Figure 3.32: PSoC Creator window

And, behind the scenes, PSoC Creator features:

Synthesis and Place&Route for designs described either in Verilog or with schematics capture tool.

Code compilation using ARM Keil or GCC ARM compilers (both included *).

Programming of PSoC devices.

* ARM Keil code optimization up to level 4 with the included license.

3.4.1 Cypress Component Catalog

PSoC supports a wide variety of functions (UART, timers, SPI...), called components. Many of these components are implemented using the programmable logic inside the PSoC seen in 3.3.2.

PSoC Creator provides a component catalog by Cypress. A component catalog is a collection of components arranged in a tree-hierarchy. Many of these components are interfaces to physical peripherals on PSoC, while many others are implemented on PSoC's UDBs and other are encapsulated software or even just annotation symbols for the schematics.

Appendix A has an example of Cypress Component Catalog content. Taking a look to this appendix is advised in order to understand how powerful PSoC is right out-of-the-box.

3.4.2 Custom components

Though Cypress provides an extensive library of ready to drag-and-drop components, custom components can be designed and used in user's projects.

The simplest form of a component will merely wrap some lower level components, with *high degree of encapsulation* thus *boosting portability.*

More complex components make use of UDBs to perform their tasks, this *unburdens CPU* performing its operations on hardware instead of software. UDB-based tasks *drain less power* than CPU-based.

In sections 4.3 and 5.3.5, custom components are created using part the available features.

Creating a custom component

Creating a custom components involves different tasks:

Create the library project Initially, there will only be one PSoC Creator library project containing components. Component authors will create components separately in that project. As more components are developed over time, it may be necessary to create separate library projects for different kinds of components.

Application projects will refer to this library to use its components.

The first step in the process of creating a component is to add one of several types of component items to your project, such as a *symbol*, *schematic*, or *Verilog* file. When you add a component item, you also create a component indirectly. The process used to add any of these component items is the same; however, each of the component items requires a different set of steps to complete.

Create a component/symbol A symbol is a visual representation of the component for the schematics capture. A component is automatically instantiated when it is dropped on a schematic, even if it has no connections with other devices.

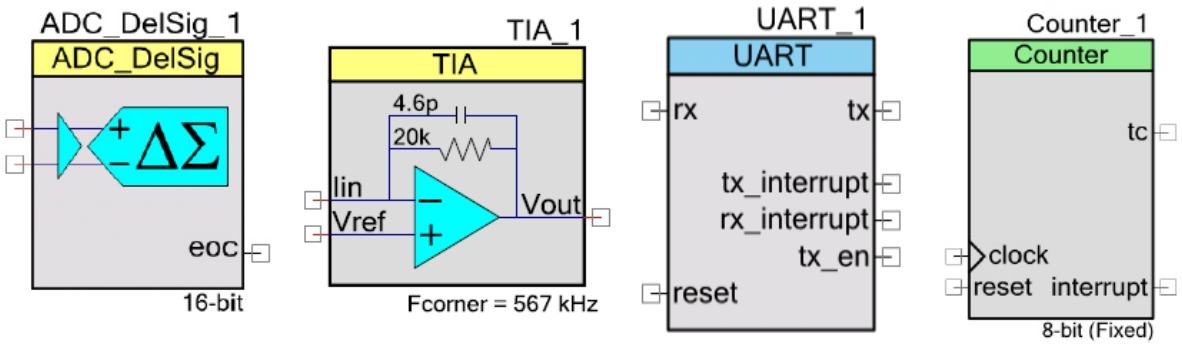


Figure 3.33: View of several component symbols from the Cypress Component Catalog

PSoC Creator includes the tools to create a component symbol. This include drawing shapes, giving colors, adding text, lines, and ports.

Define symbol information The symbol contains general information about the component itself.

Parameters are one of these units of information about the component. Every component has some pre-defined parameters, such as *component name*, and other parameters might be user defined to let the final user customize the component for the target design without need to modify the component itself.

PSoC Creator has an expression interpreter to make decisions upon these parameters. Parameters can affect the appearance of a symbol and will also be accessible by inner schematic components and the software pre-processor, changing the component behavior on a instance-by-instance basis.

The properties of a symbol contains information that do not affect the component behavior, such as the reference to the component datasheet, the base instance name and placement in the component catalog, among others.

Create the implementation The implementation of a component specifies the device and how the component is implemented in that architecture, family or device. There must be at least one implementation in a component, with the exception of annotation components.

There are several methods to implement the component: schematic, Verilog, and software.

The schematic is a graphical diagram that represents the internal connectivity of the component. Other components can be instantiated in a schematic view, providing ability to create complex nested components.

Verilog can be used to describe the functionality of your component. PSoC Creator allows adding a Verilog file to the component, as well as write and edit the file. There are certain requirements, and that PSoC Creator only supports the synthesizable subset of the Verilog language; refer to the Warp Verilog Reference Guide [27]. You also may need to use the Datapath

Configuration Tool to edit datapath instance configurations; see Datapath Configuration Tool on the Component Author Guide [18].

Designs are likely to be directly ported between all versions of PSoC, however PSoC Creator provides mechanisms to discriminate between silicon revision so code can be synthesized adapting to the specific differences between them.

Simulate the hardware Synthesis of Verilog source designs is made possible with Warp. Although Cypress does not at this time provide a Verilog simulator, some system files necessary for third party simulators to provide pre-synthesis simulations are available.

These files contain functional description for PSoC primitives such as registers, timers, datapaths... so Verilog-based designs can be simulated accurately.

Cypress does not provide a simulator for the schematics either.

Create API files Application Programming Interface (API) files define the component interface, which provides the various functions and parameters for a component. The component author creates API file templates for the component that become instance-specific for an end-user.

API generation is the process by which PSoC Creator creates instance-based code, including the generation of symbolic constants used to define hardware addresses. The hardware addresses are a result of placement (one of the steps in the fitter, which is part of the build process). PSoC Creator generates instance-specific macros for these addresses, so references to registers remain consistent across different placements and devices.

Parameters are accessible from within source code as stated before. Using parameters PSoC Creator can (besides accessing the values themselves), accomplish the correct naming of generated files and functions so different instances of the same component do not affect the other.

To understand how this is done, assuming that we are creating a component, its *start* function will be coded as:

```
1 void `$INSTANCE_NAME` _Start();
```

When instantiated as `foo_1`, the automatically generated source will rename this function to:

```
1 void foo_1_Start();
```

The same can be done with variables and macro definitions.

Customize the component Customizing components refers to the mechanism of allowing custom code (C♯) to augment or replace the default behavior of an instantiated component within PSoC Creator. The code is sometimes referred to as *customizers*, which may:

- Customize the Configure dialog

- Customize symbol shapes / display based on parameter values
- Customize symbol terminal names, counts, or configuration based on parameters
- Generate custom Verilog code
- Generate custom C/assembly code
- Interact with the clocking system (for clock and PWM components)

PSoC Creator provides C♯ specific libraries for component customization.

Add tuning support (advanced) Tuning support is a highly advanced feature that is used for a few limited components, such as CapSense, that require the ability for the user to tune the component while it is running in the design.

Tuning requires communication between the firmware component and the host application. The firmware sends scan values to the Graphical User Interface (GUI) and the GUI sends updated tuning parameters down to the firmware. This is an iterative process in order to get the best possible results out of the device.

PSoC Creator supports this requirement by providing a flexible API that component authors can use to implement tuning for their component and communication to the device is done using an I^2C or EzI^2C component in the end user's design.

Add bootloader support (as needed) For a component to support bootloading (see section 3.4.3), it must implement five functions used by the bootloader for setting up the communications interface and relaying packets back and forth with the host:

- CyBtldrCommStart
- CyBtldrCommStop
- CyBtldrCommReset
- CyBtldrCommWrite
- CyBtldrCommRead

Add/create documentation and other files/documents Documentation can be added to the component in any of PDF, HTML, and XML formats. Following paragraphs are an overview of PSoC Creator features for component authoring. Refer to the Component Author Guide [18] for details.

The component datasheet must be a PDF file. When a user opens the datasheet from the tool, it will always open the PDF file associated with the component. The main requirement is that a PDF file must have the same name as the component in order to be viewed from the Component Catalog.

A control file is used to define fixed placement characteristics for a component instance. It allows a component author to define the target datapath, PLD and status/control register resources.

A debug XML file is an optional mechanism for creating new debugger tool windows for any component. To enable this feature on a specific component, the component author has to add an XML description of the component. This information can consist of any blocks of memory used or individual registers that are important to the component.

The CyState XML file provides component state information. It is an optional feature used to generate Design Rules Check (DRC) errors, warnings and notes if a component is not of production quality, or if the component is used with incompatible versions of silicon.

Build and test the component When all necessary component items have been added and completed, it must be instantiated in a design in order to build and test it. When a component project is built, all portions of it are built, including the family-specific portions. After a build, the build directory will contain all aspects of the component that the component author has specified via the project manager.

3.4.3 Bootloader

A bootloader enables field updates of device firmware without the need for dedicated external programming hardware [15]. PSoC Creator supports firmware updates through the USB and I^2C communication channels. PSoC Creator also allows you to customize the bootloader communication channel to use other protocols such as UART or SPI for bootloading.

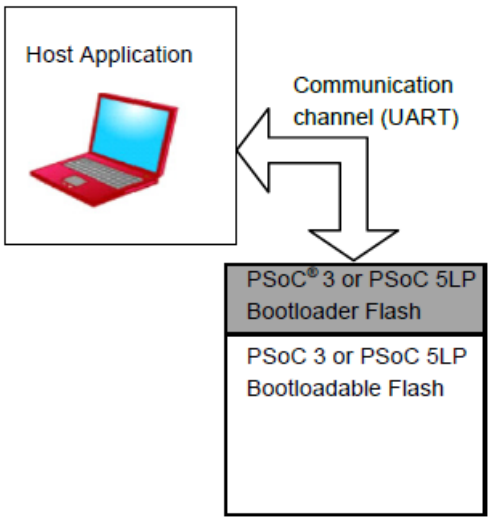


Figure 3.34: UART Bootloader System

A typical PSoC bootloader system has the following elements:

Bootloader host

It can be a PC or an embedded host capable of communicating with the target PSoC

device. It sends the new application code that needs to be written to the target over the selected communication interface.

Bootloader project
This resides on the PSoC device and manages the process of updating device flash memory with the new application code and data. It includes a communication interface such as I^2C, USB, or UART that communicates with the host to get the new application code and data.

Bootloadable project
This is the actual application that runs on PSoC, which is upgraded during bootload operation.

4

PSoC and sensors

Across the following sections, different sensors have been arranged and their signal conditioning problem has been solved using PSoC capabilities.

4.1 Resistive sensing application examples

4.1.1 Temperature reading using a thermistor

In this example, a thermistor is the bottom half of a voltage divider. Both parts of the voltage divider are then connected to the *Delta-Sigma ADC* via an analog multiplexer [14].

The schematic is illustrated on figure 4.1. Blue components and wires are off-chip, included just for annotation.

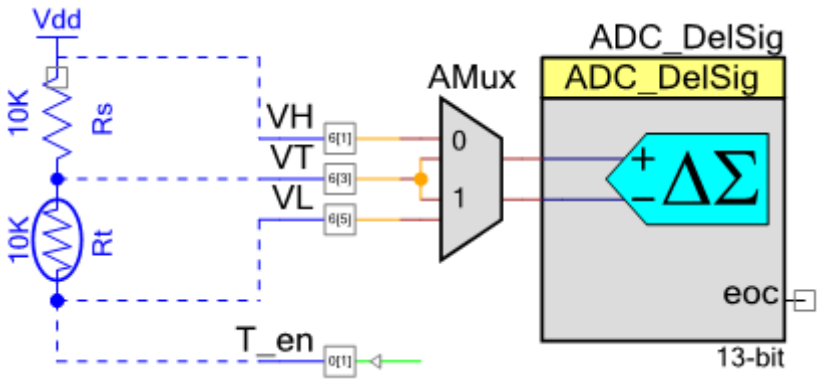


Figure 4.1: Thermistor schematic on PSoC Creator

The bottom node of the voltage divider is dynamically (un)tied to ground via a digital output pin. This pin is configured as *open-drain drives low*, as illustrated on figure 4.2. While this pin drives low, it sinks the current from the voltage divider (up to $8mA$), while it "drives high",

the node just floats avoiding any current drain through it. This prevent the thermistor from self-heating while reading is not in progress. It is also a power-saving method, topic discussed on chapter 6.

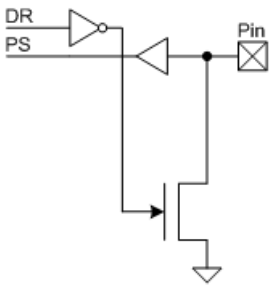


Figure 4.2: Open-drain drives low T_en pin configuration

In this circuit, thermistor resistance can be calculated by measuring voltage drop in R_t and R_s, then

$$R_t = R_s \frac{V_T - V_L}{V_H - V_T}$$

Then a LUT is contrasted to get the temperature value for the calculated resistance.

Cypress Component Catalog provides a *thermistor calculator* component (figure 4.3), which helps creating this LUT (figure 4.4) and generates methods for converting between temperature, voltage and resistance.

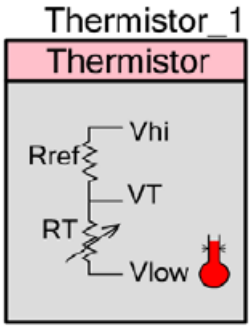


Figure 4.3: Thermistor Calculator component symbol

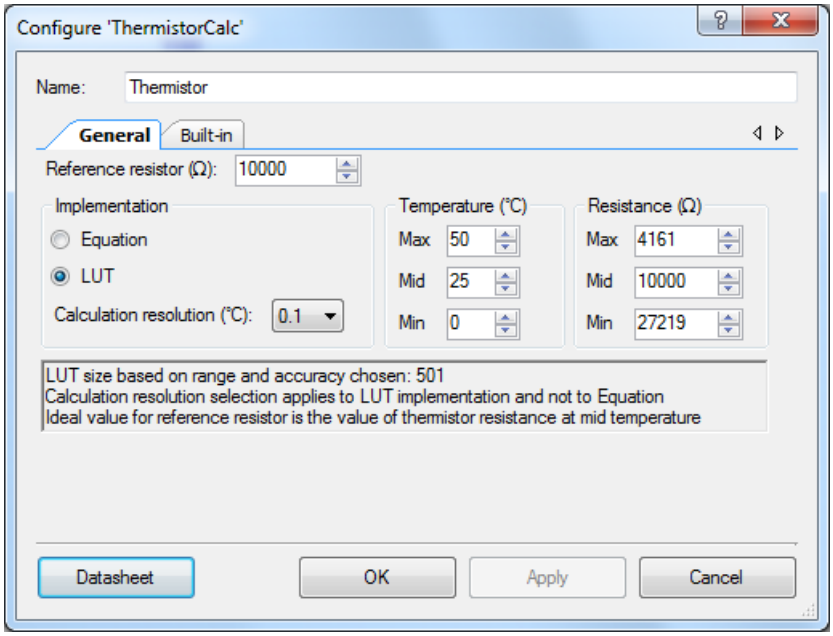


Figure 4.4: Thermistor Calculator component configuration dialog

4.1.2 Light detection with an LDR and programmable threshold

In this example, the output of a Light Dependent Resistor (LDR) placed in a voltage divider is wired to the non-inverting input of a comparator to test against a threshold value. The inverting input is connected to the output of a VDAC in order to get a programmable value of the threshold.

The output of the comparator has inverted polarization and is directly connected to a status register which can be read by the CPU as a boolean value for light detection or not.

The schematic is illustrated on figure 4.5. Cypress Components Catalog does not provide an LDR, it has been created within PSoC Creator.

How it works

1. Start *Comparator*.
2. Start VDAC. VDAC has already set the threshold value.
3. LDR_en pin drives low.
4. CPU reads the *Light* register.
5. CPU turns on/off a Light Emitting Diode (LED) according to register value.
6. CPU sleeps for 1 second before starting the process again.

As seen on section 4.1.1, the bottom node of the voltage divider is connected to ground via an open-drain drives low pin to save power while LDR is not in use.

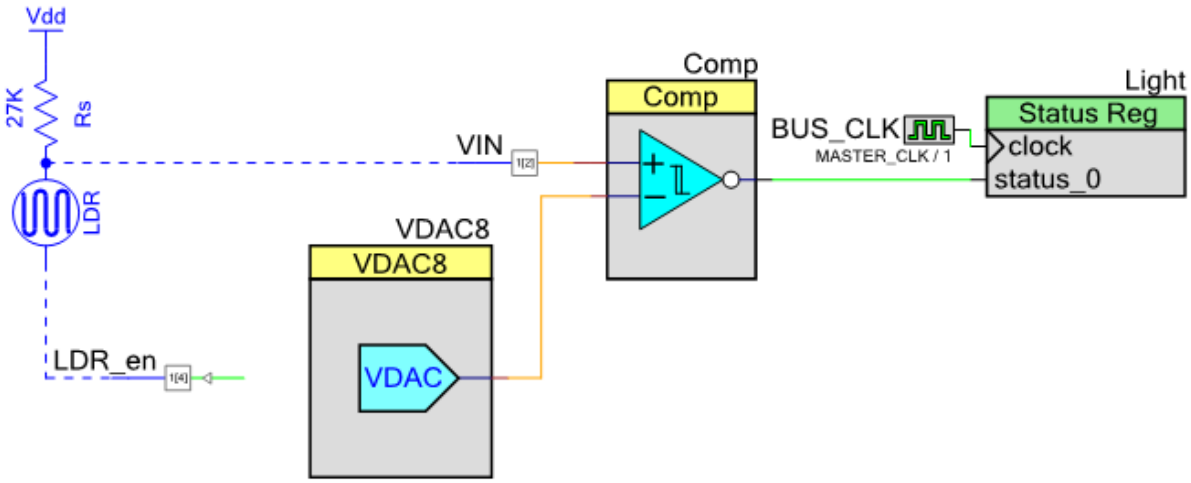


Figure 4.5: Light detection schematic on PSoC Creator

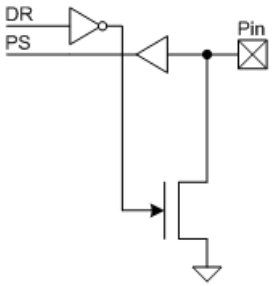


Figure 4.6: Open-drain drives low LDR_en pin configuration

4.2 Current sensing application example

4.2.1 Measuring current from a phototransistor

In this example a *TCRT5000 Reflective Optical Sensor with Transistor Output* is used to measure the current generated by the photoreceptor using a Trans-Impedance Amplifier (TIA) from the SC/CT block.

The TCRT5000 is a reflective sensor which include an infrared emitter and phototransistor in a leaded package which blocks visible light [32]. Figure 4.7 shows the sensor and its internal structure.

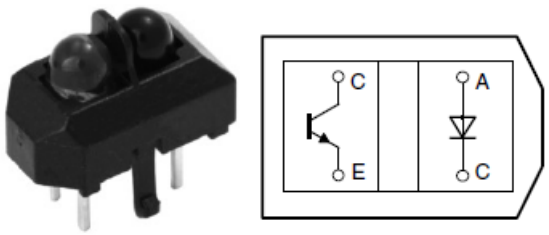


Figure 4.7: TCRT5000 sensor and internal structure

Focusing on the phototransistor part, it is biased connecting its collector to a $5.0V$ source.

The emitter is connected to the inverting input of the TIA, which has a reference of $1.024V$ connected to its non-inverting input. Output of the TIA is connected to an external GPIO to monitor the resulting signal with an oscilloscope. Schematic of the excitation and conditioning circuits is illustrated in figure 4.8.

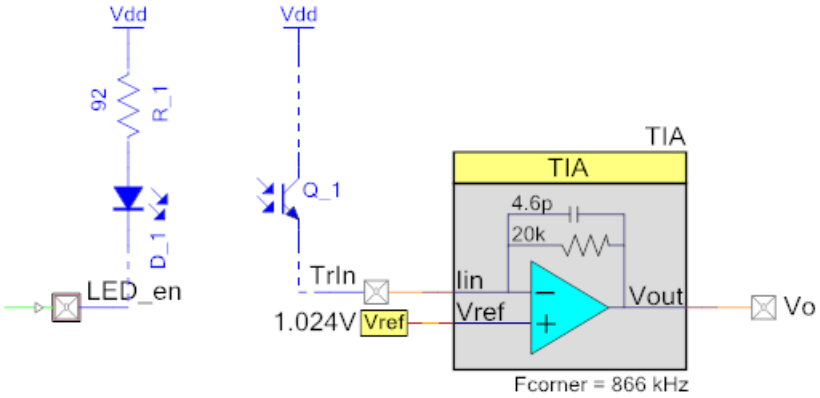


Figure 4.8: Phototransistor excitation and signal conditioning schematic

As a result, the current driven by the phototransistor can be calculated using the equation

$$V_o = V_{ref} - I_c * R_{FB}$$

thus,

$$I_c = \frac{V_{ref} - V_o}{R_{FB}}$$

Channel 1 of the oscilloscope is connected to the TIA output. An square wave drives the IR diode to switch excitation on and off and observe the effect on the phototransistor. Channel 2 is connected to the cathode of the diode, so low means driving. Figure 4.9 shows the excitation and output signals.

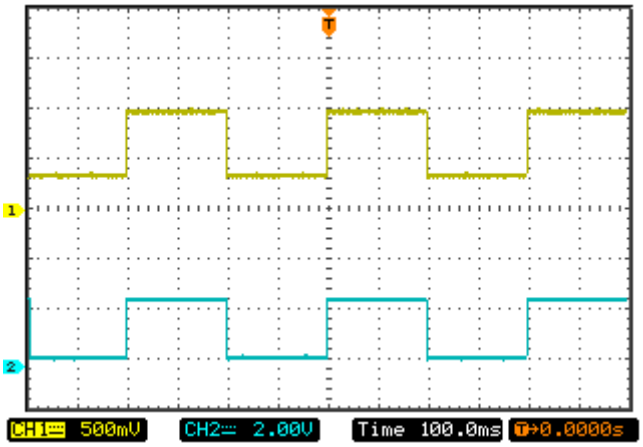


Figure 4.9: Phototransistor excitation and output signals

Solving for I_c,

$$I_c = \frac{1.024 - 0.340}{20e3} = 34.2\mu A$$

4.3 Measuring relative humidity and temperature with the DHT11 digital sensor

The DHT11 sensor is a digital sensor for relative humidity and temperature [29].

The DHT11 has a measuremente range of 20 to 90% of relative humidity and 0 to 50 celsius degrees of temperature, in steps of $\pm 1\%RH$ and $\pm 1^oC$. Each sensor is calibrated in laboratory and the calibration data is stored in its internal memory, making it fairly accurate for general purposes, $\pm 5\%RH$ and $\pm 2^oC$.

4.3.1 Technical specifications

Humidity measurement specifications:

Parameters	Conditions	Minimum	Typical	Maximum
Resolution		$1\%RH$	$1\%RH$	$1\%RH$
Repeatability			$\pm 1\%RH$	
Accuracy	25^oC		$\pm 4\%RH$	
	$0 - 50^oC$			$\pm 5\%RH$
Measurement range	0^oC	$30\%RH$		$90\%RH$
	25^oC	$20\%RH$		$90\%RH$
	50^oC	$20\%RH$		$80\%RH$
Long term stability			$\pm 1\%RH/year$	

Table 4.1: DHT11 humidity measurement specifications

Temperature measurement specifications:

Parameters	Conditions	Minimum	Typical	Maximum
Resolution		1^oC	1^oC	1^oC
Repeatability			$\pm 1^oC$	
Accuracy		$\pm 1^oC$		$\pm 2^oC$
Measurement range		0^oC		50^oC
Long term stability			$\pm 1\%RH/year$	

Table 4.2: DHT11 temperature measurement specifications

4.3.2 Typical configuration

A single-wire bus is used to transfer data back and forth between the DHT11 and PSoC. Figure 4.10 shows the typical configuration of the DHT11 connected to the PSoC.

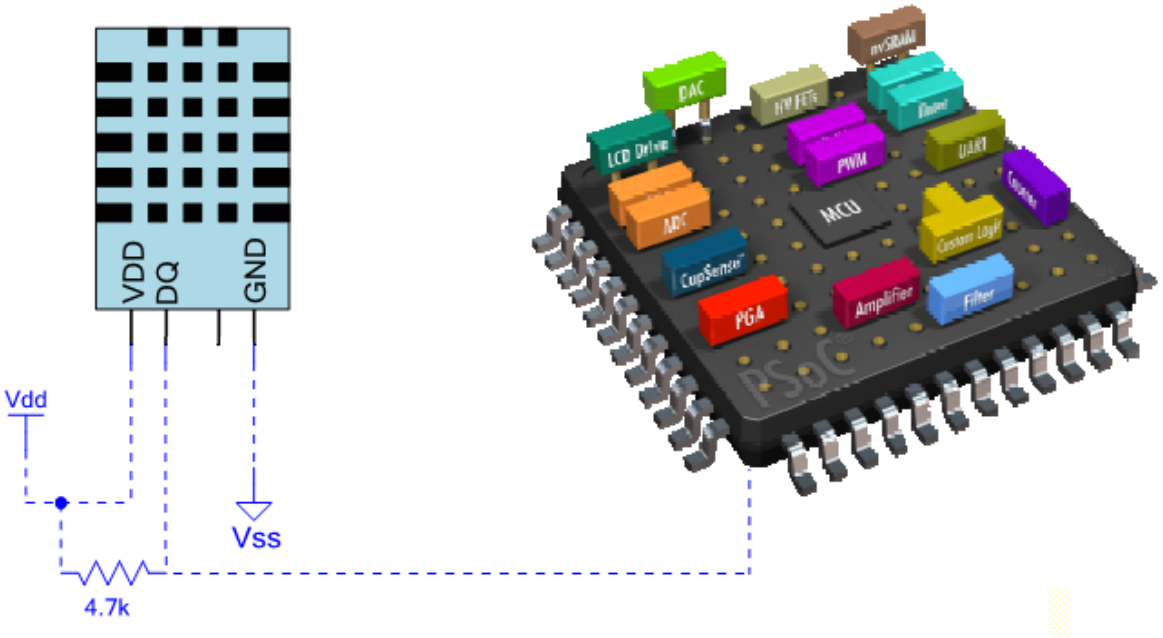


Figure 4.10: DHT11 typical configuration

The frame format is illustrated on figure 4.11.

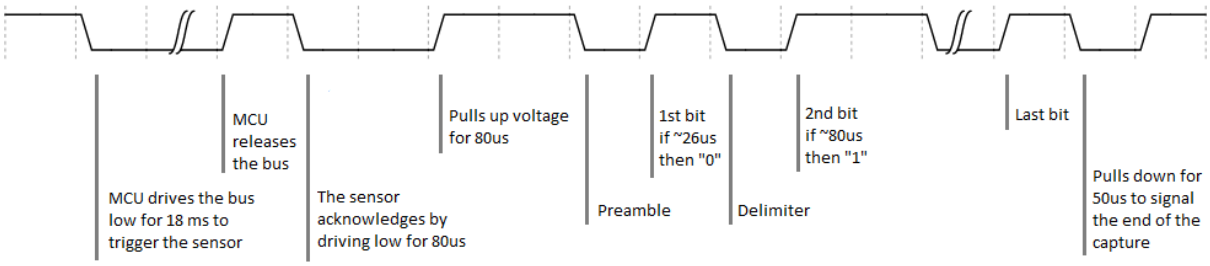


Figure 4.11: DHT11 frame format

A total of 40 bits are transferred in each transmission in the following order: 8 bits for the integral part of humidity, 8 bits for the decimal part of relative humidity, 8 bits for the integral part of temperature, 8 bits for the decimal part of temperature plus 8 bits of checksum which are the less significant ones of the sum of the previous 32.

The whole communication process takes less than 25 ms.

To read the data from the component, three ways have been studied:

Bit-banging The CPU reads the data pin continuously, timing its transitions and thus following the sensor response and deciding between 0's and 1's. The CPU is active and busy during the entire read process. Mishandled interrupts from other sources might be harmful.

Interrupt-driven The CPU starts a timer and goes to sleep being only woken up by an edge-sensitive interrupt on each data pin transition. Then it can measure the width of the pulse and restart the counter for the next bit. The CPU is sleeping much part of the process, but many interrupts, up to 84, are being triggered during the read process. This solution

may be better than the bit-banging option, but it still is power demanding, complex to code and special care should be taken if mixing interrupts from other sources during the process.

Component-driven A PSoC custom component is developed to fit into the UDB blocks and perform the entire read with no CPU intervention, allowing the CPU either sleep or attend other tasks. This solution is the most power-efficient, and most complex to implement though.

Since the *bit-banging* and *interrupt-driven* options are equally available for every other MCU than PSoC, the *component-driven* solution has been implemented to show the real power of PSoC addressing the task.

4.3.3 DHT11 Component

Approach Figure 4.12 shows the schematic of the top-level component. It has been highly modularized to ease the implementation and debugging.

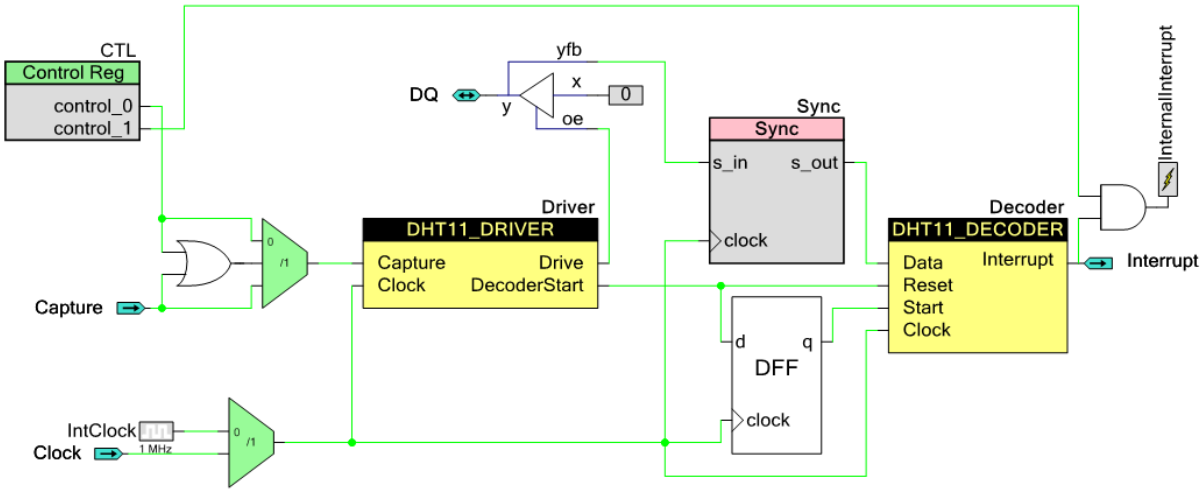


Figure 4.12: DHT11 component internal schematic

It has to be fed with a 1 MHz clock and handles all the sensor excitation, response capturing and data storing. Thanks to its hardware pins for *Capture* and *Interrput*, toghether with a harware timer and the DMA controller it could work in a stand-alone fashion, completely **without CPU intervention**.

Driver The driver is the component in charge of sensor excitation. It sends the *init* signal for the correct amount of time, 18 ms, and flags other parts of the component to be ready for the response.

It uses a UDB-based 16-bit counter and a control unit featuring a finite state machine described using Verilog that manages the *init* process.

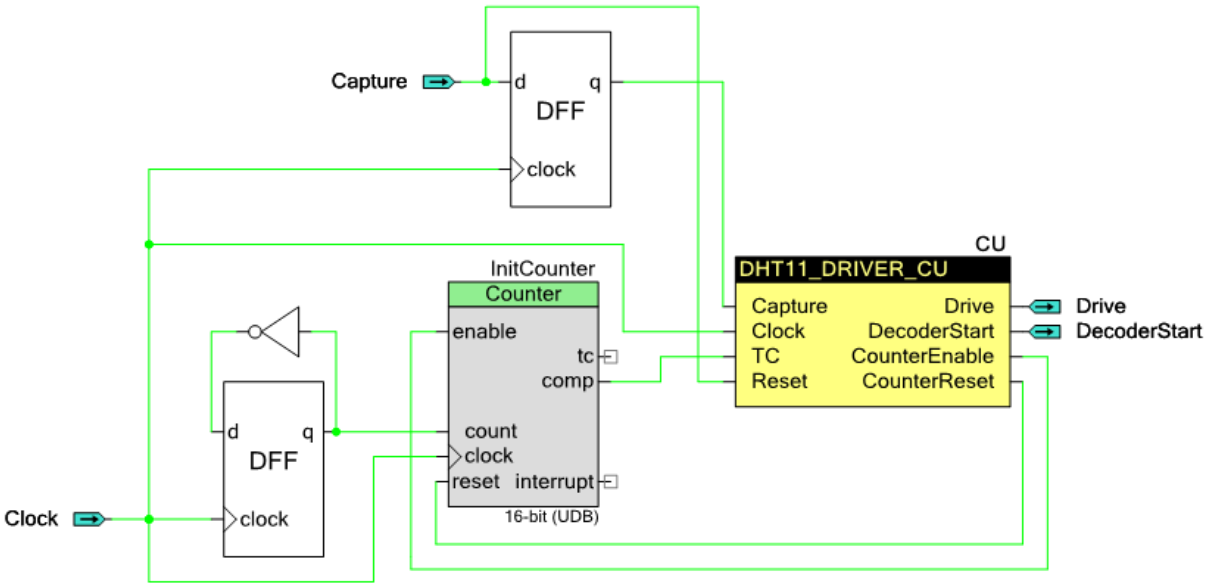


Figure 4.13: DHT11 Driver internal schematic

Decoder The decoder is mainly a finite state machine that interprets the sensor response, a datapath to store the values and a state machine to drive the datapath. Figure 4.14 represents the state machines states and transitions.

The component leverages the four-bytes-depth FIFO of the datapath to store the sensor output.

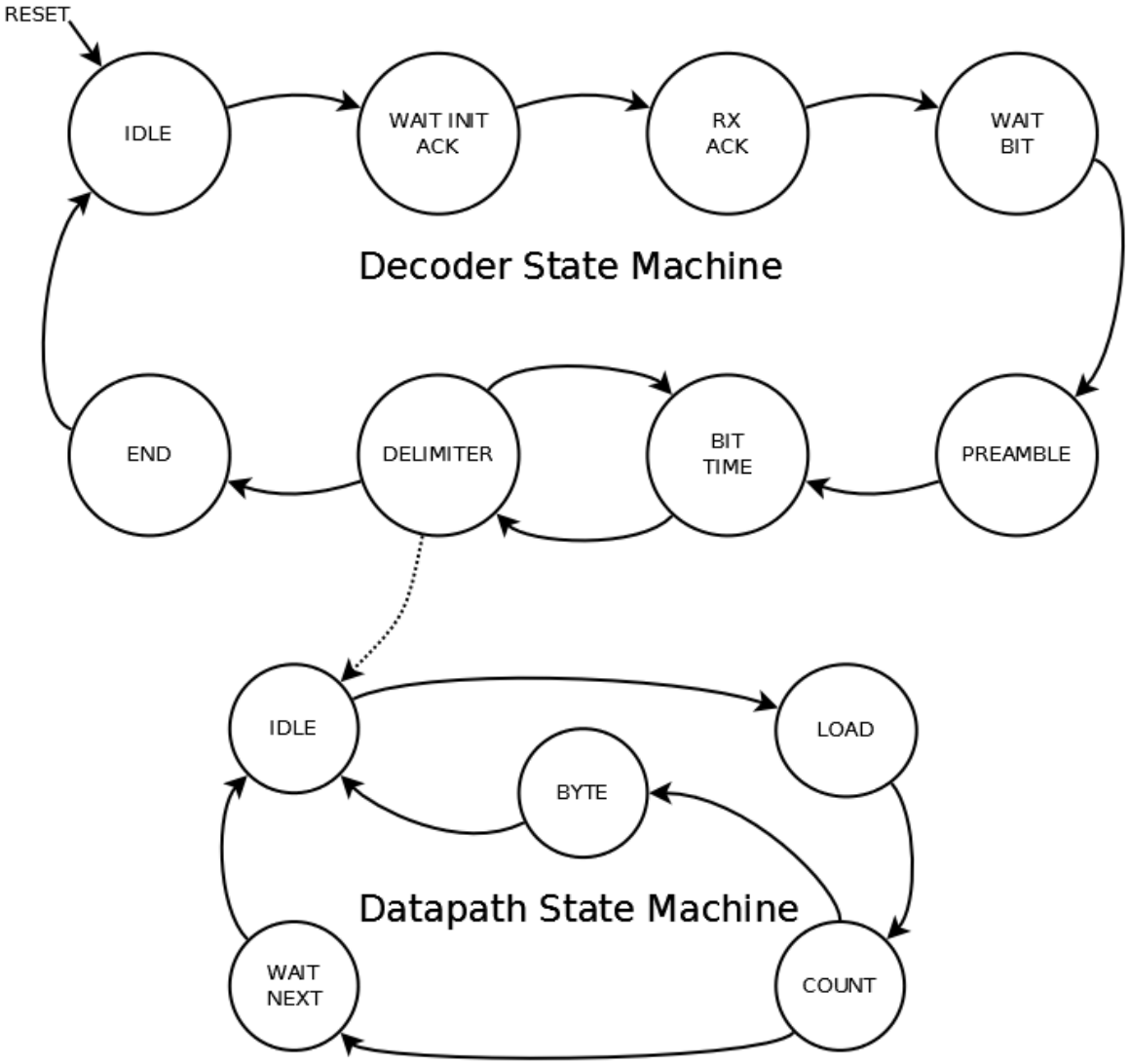


Figure 4.14: DHT11 Decoder finite state machine

5

PSoC communication capabilities

5.1 Dedicated peripherals

Cypress PSoC3 and PSoC5LP feature a common set of communications peripherals. These are easily accessible from the Cypress' Component Catalog on PSoC Creator.

5.1.1 Full Speed USB 2.0

There is one USB controller available on the PSoC.

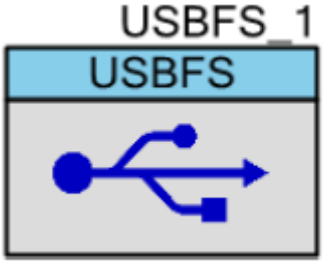


Figure 5.1: USB component symbol

The USBFS component [26] provides a USB full-speed Chapter 9 compliant device framework. It provides a low-level driver for the control endpoint that decodes and dispatches requests from the USB host. Additionally, this component provides a USBFS customizer to make it easy to construct your descriptor.

You have the option of constructing a Human Interface Device (HID)-based device or a generic USB Device. Select HID (and switch between HID and generic) by setting the Configuration/Interface descriptors.

- USB 2.0 Full Speed (12 Mbps)

- 8 unidirectional endpoints
- Shared 512-byte buffer for the eight data endpoints
- 1 bidirectional control endpoint 0 (EP0)
- Dedicated 8-byte buffer for EP0
- Transfer Types: Control, Interrupt, Bulk, Isochronous
- DMA access capable
- Wake from sleep
- Internal 48 MHz oscillator locks to USB bus clock
- USB Reset, Suspend, and Resume operations
- Bus powered and self-powered modes

Cypress provides Windows drivers along other development tools for the USB controller.

5.1.2 CAN 2.0b

The Controller Area Network (Controller Area Network (CAN)) controller implements the CAN2.0A and CAN2.0B specifications as defined in the Bosch specification and conforms to the ISO-11898-1 standard.

The CAN protocol was originally designed for automotive applications with a focus on a high level of fault detection. This ensures high communication reliability at a low cost. Because of its success in automotive applications, CAN is used as a standard communication protocol for motion-oriented machine-control networks (CANOpen) and factory automation applications (DeviceNet). The CAN controller features allow you to efficiently implement higher-level protocols, without affecting the performance of the microcontroller CPU.

The PSoC CAN bus controller offers:

- Ports are routable to any GPIO
- CAN 2.0A/B compliant
  - Remote Transmission Request (Remote Transmission Request (RTR)) support
  - Programmable bit rate up to 1 Mbps
  - External CAN Physical (PHY) connects to any GPIO
- Transmit path
  - 8 transmit message buffers
  - Programmable priority for each (Round Robin, fixed priority)
  - 16 receive message buffers
  - 16 acceptance filters/masks
  - DeviceNet addressing support
  - Option to link multiple receive buffers to/from a hardware FIFO

5.1.3 I²C

PSoC features an on-chip I²C transceiver as well as an UDB-based solution, both accesible through the I²C component from the Cypress' Component Catalog.

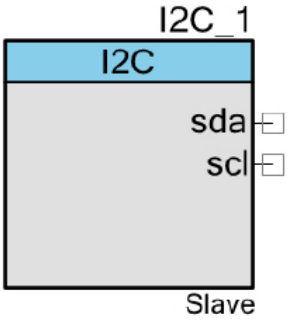


Figure 5.2: I²C component symbol

The I²C component supports I²C slave, master, and multi-master configurations [2]. The I²C bus is an industry-standard, two-wire hardware interface developed by Philips. The master initiates all communication on the I²C bus and supplies the clock for all slave devices.

The I²C component supports standard clock speeds up to 1000 kbps. It is compatible with I²C Standard-mode, Fast-mode, and Fast-mode Plus devices as defined in the NXP I²C-bus specification. The I²C component is compatible with other third-party slave and master devices.

- Slave, master or multi-master
- Hardware or firmware address decode
- 7 or 10-bit addressing (10-bit addressing requires firmware support)
- Interrupt or polling CPU interface
- Interrupts for variety of bus events
- Standard 100 Kbps and Fast 400 Kbps (fixed function), and high-speed 3.4 Mbps (UDB-based)
- System Management BUS (SMBus) operation (through firmware support)
- Routes Serial Data (SDA)Serial Clock (SCL) to any GPIOSIO pins
- Wake from low power modes on address match if SIO connections used

The I²C component is an ideal solution when networking multiple devices on a single board or small system. The system can be designed with a single master and multiple slaves, multiple masters, or a combination of masters and slaves.

Figure 5.3 illustrates a typical use case.

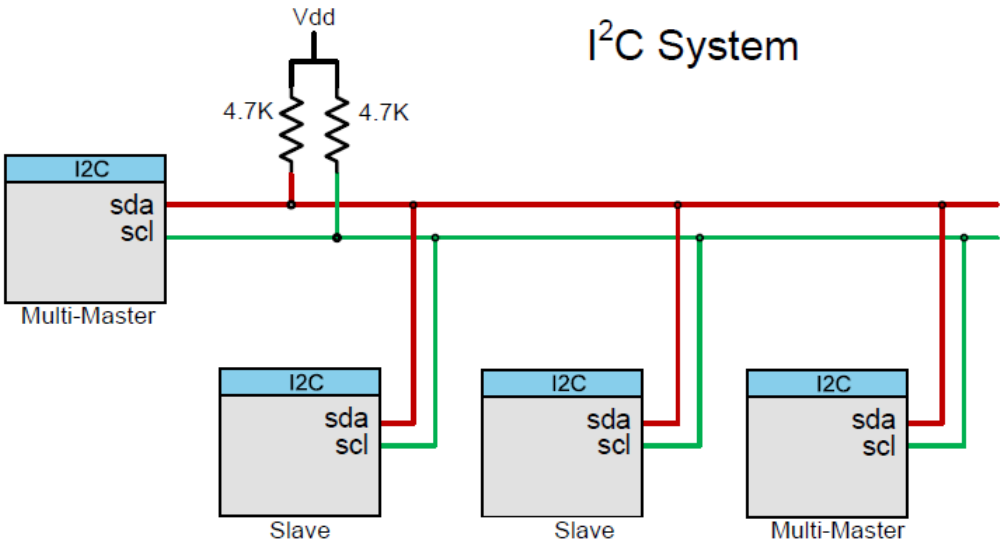


Figure 5.3: I^{2}C peripherals example configuration

5.2 UDB-based peripherals

Cypress' Components Catalog offers a wide range of communications peripherals which are mapped into UDB arrays.

An user can instantiate as many USB-based peripherals as PSoC Creator can make them fit into the chip.

The catalog has plenty of communication components, and it is growing as releases of PSoC Creator are coming out. Thus in the following sections only some of them will be described to give an idea of which are the capabilities of PSoC.

5.2.1 Universal Asynchronous Receiver Transmitter

The UART provides asynchronous communications commonly referred to as RS-232 serial communications standard (RS232) or RS-485 serial communications standard (RS485) [9]. The UART component provided by Cypress can be configured for Full Duplex, Half Duplex, Reception (RX) only, or Transmission (TX) only versions. All versions provide the same basic functionality and they differ only in the amount of resources used.

To assist with processing of the UART receive and transmit data, independent size configurable buffers are provided. The independent circular receive and transit buffers in SRAM and hardware FIFOs help to ensure that data will not be missed. This allows the CPU to spend more time on critical real time tasks rather than servicing the UART.

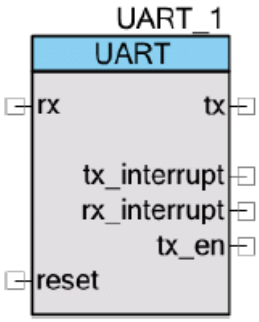


Figure 5.4: UART component symbol

Features of the UART component

- 9-bit address mode with hardware address detection
- Baud rates from 110 to 921600 bps or arbitrary up to 4 Mbps
- RX and TX buffers from 4 (hardware FIFO) to 65535 (SRAM) bytes
- Detection of Framing, Parity, and Overrun errors
- Full Duplex, Half Duplex, TX only, and RX only optimized hardware
- Two out of three voting per bit
- Break signal generation and detection
- 8x or 16x oversampling

The UART can be used any time a compatible asynchronous communications interface is required, especially RS232 and RS485 and other variations. The UART can also be use used to create more advanced asynchronous based protocols such as DMX512 stage lightning control communications standard (DMX512), Local Interconnect Network (LIN), and Infrared Data Association (IrDA), or customer or industry proprietary.

Resources

Configuration	Datapaths	Macrocells	Status regs	Control regs	Interrupts
Full UART	3	59	2	2	2
Simple UART	3	22	2	1	0
Half duplex	1	21	1	2	0
RX only	1	12	1	1	0
TX only	1	10	1	1	0

Table 5.1: UART component resource usage
Refer to section 3.3 for information on resources.

5.2.2 Serial Peripheral Interface

Cypress provide components for SPI communications for both *master* and *slave* modes [6] [7].

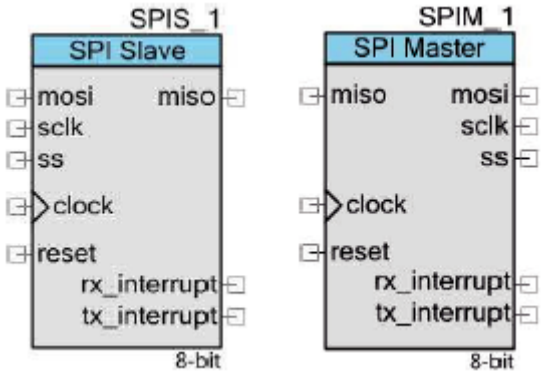


Figure 5.5: SPI slave (left) and master (right) component symbols

The SPI components provide an industry-standard, 4-wire slave SPI interface. They can also provide a 3-wire (bidirectional) SPI interface. Both interfaces support all four SPI operating modes, allowing communication with any SPI device. In addition to the standard 8-bit word length, the SPI components support a configurable 3 to 16-bit word length for communicating with nonstandard SPI word lengths.

SPI signals include the standard Serial Clock (Serial Clock (SCLK)), Master In Slave Out (Master In / Slave Out (MISO)), Master Out Slave In (Master Out / Slave In (MOSI)), bidirectional Serial Data (Serial Data (SDAT)), and Slave Select (SS).

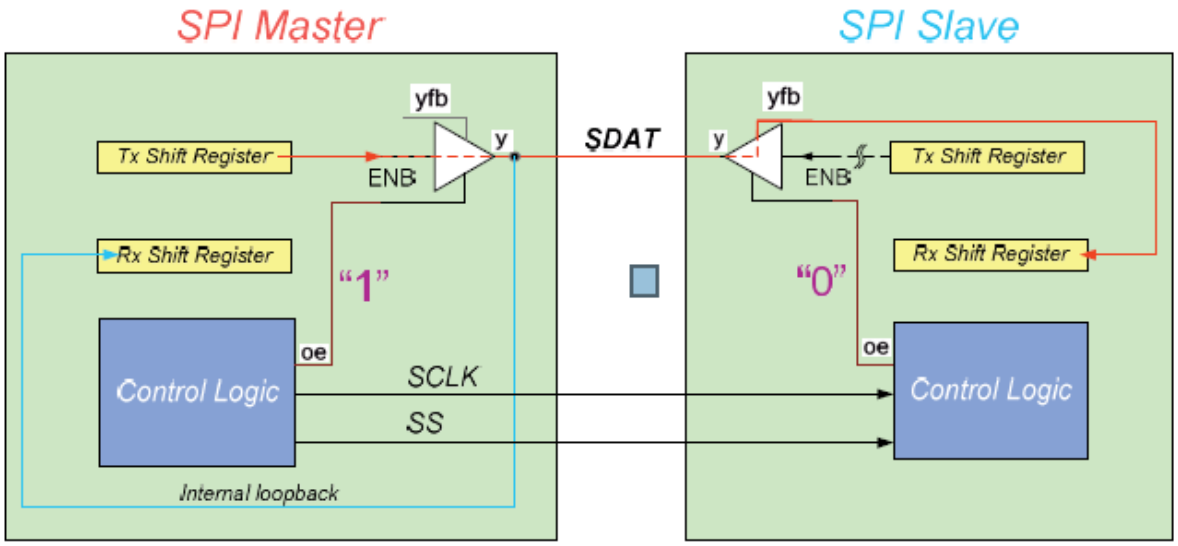


Figure 5.6: SPI component master to slave communication

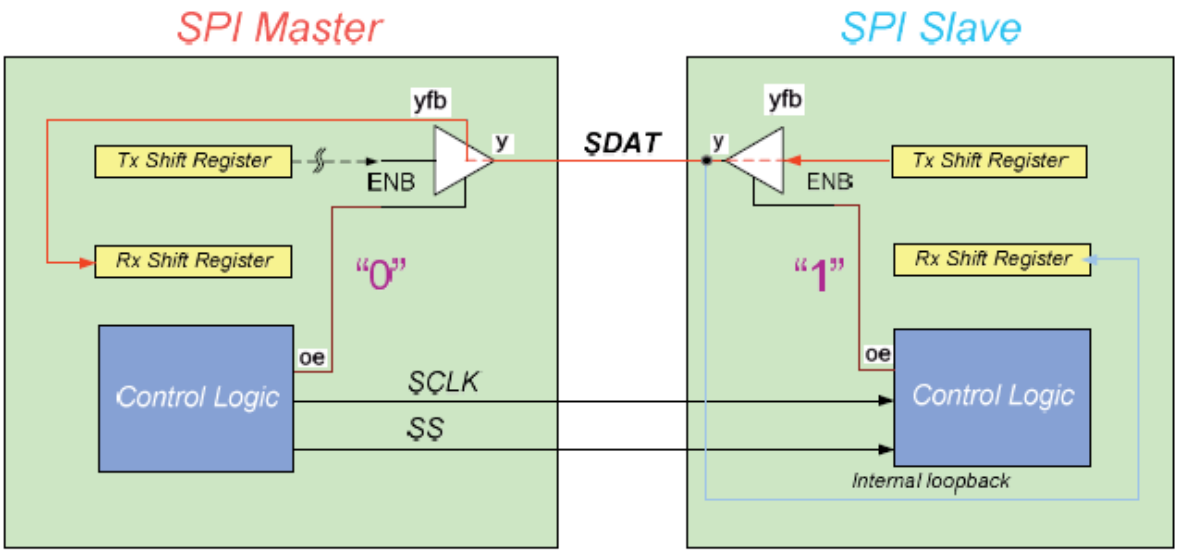


Figure 5.7: SPI component slave to master communication

To communicate with multiple devices, the master's SS output must be demultiplexed. An easy way to do this with PSoC is described in figure 5.8. Remember that the SS is active-low.

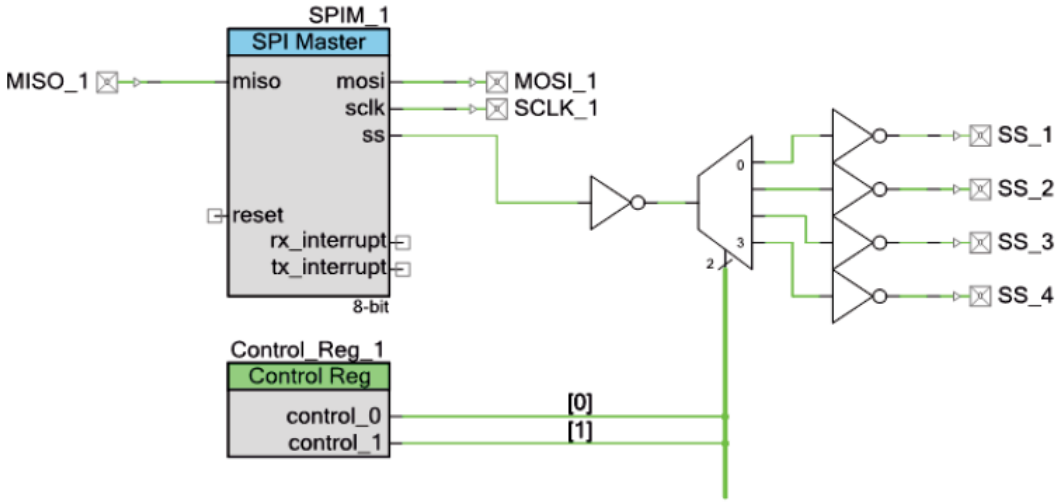


Figure 5.8: SPI component SS signal demultiplexing

The user can demultiplex the SS signal as he or she wishes though, due to the highly flexible DSI on PSoC.

Resources

Resource usage is summarized in table 5.2.

Configuration	Datapaths	Macrocells	Status regs	Control regs	Interrupts
8-bit (MOSI+MISO)	1	12	2/3*	1	2
8-bit (Bidirectional)	1	12	2/3*	2	2
16-bit (MOSI+MISO)	2	12	2/3*	1	2
16-bit (Bidirectional)	2	12	2/3*	2	2
8-bit High Speed** (MOSI+MISO)	1	18	2	1	2
8-bit High Speed** (Bidirectional)	1	18	2	2	2
16-bit High Speed** (MOSI+MISO)	2	18	2	1	2
16-bit High Speed** (Bidirectional)	2	18	2	2	2

Table 5.2: SPI component resource usage
* Master/Slave values
** Mode only available for master

5.3 Wireless communication

This section proves how easy is to integrate wireless modules with PSoC. For the example, the XBee family has been chosen for its high acceptation and presence among open-source projects.

XBee modules use the UART to communicate with their host, but the procedure can be extrapolated for other transceivers regardless the communication peripheral they use.

5.3.1 XBee modules

XBee is the brand name from Digi International for a family of form factor compatible radio modules. There are many different versions, some of them are compatible with each other, but others not. Table 5.3 holds information about available models, specs and features.

All XBee radios have in common that they encapsulate radio operation in the module and provide an interface to the user for easy operation via UART communication.

There are two modes of operation:

Transparent
When operating in transparent mode, the modules act as a serial line replacement. All UART data received through the Data In (DIN) pin is queued up for Radio Frequency (RF) transmission. When RF data is received, the data is sent out through the Data Out (DOUT) pin. The module configuration parameters are configured using the Hayes command (AT) command mode interface.

API Mode
API operation is an alternative to transparent operation. The frame-based API extends the level to which a host application can interact with the networking capabilities of the module. When in API mode, all data entering and leaving the module is contained in frames that define operations or events within the module.

Product	Frequency	Power	Range	Data Rate	Protocol	Multipoint/Mesh
XBee ZB	$2.4GHz$	$1.25/2mW$	$120m$	$250Kbps$	ZigBee	Mesh
XBee-PRO ZB	$2.4GHz$	$63mW$*	$3.2km$	$250Kbps$	ZigBee	Mesh
XBee ZB SMT	$2.4GHz$	$3.1/6.3mW$	$1200m$	$250Kbps$	ZigBee	Mesh
XBee-PRO ZB SMT	$2.4GHz$	$63mW$	$3.2km$	$250Kbps$	ZigBee	Mesh
XBee 802.15.4	$2.4GHz$	$1mW$	$90m$	$250Kbps$	802.15.4	Multipoint
XBee-PRO 802.15.4	$2.4GHz$	$63mW$*	$1.6km$	$250Kbps$	802.15.4	Multipoint
XBee-PRO XSC	$900MHz$	$100mW$	$24km$**	$9.6Kbps$	Proprietary	Multipoint
XBee-PRO 900	$900MHz$	$50mW$	$10km$**	$156Kbps$	Proprietary	Multipoint
XBee-PRO DigiMesh 900	$900MHz$	$50mW$	$10km$**	$156Kbps$	DigiMesh	Mesh
XBee DigiMesh 2.4	$2.4GHz$	$1mW$	$90m$	$250Kbps$	DigiMesh	Mesh
XBee-PRO DigiMesh 2.4	$2.4GHz$	$63mW$	$1.6km$	$250Kbps$	DigiMesh	Mesh
XBee-PRO 868	$868MHz$	$500mW$	$80km$**	$24Kbps$	Proprietary	Multipoint
XBee Wi-Fi	$2.4GHz$	$16dBm$	$300m$	$65Mbps$	802.11bgn	Multipoint
XBee 865LP	$865MHz$	$12dBm$	$4km$	$80Kbps$	Proprietary	Both
XBee 868LP	$865MHz$	$12dBm$	$4km$	$80Kbps$	Proprietary	Both

Table 5.3: XBee modules specifications
* International variants of 2.4 GHz XBee-PRO modules are limited to a 10 mW transmission power
** Range with high gain antennas (not included)

5.3.2 XBee/XBee-PRO ZB module

Studying PSoC abilities for communication between motes, an XBee ZB module has been chosen for its cost and interoperability with ZigBee networks. Following sections will refer to the XBee/XBee-PRO ZB modules.

These modules have, besides the normal operation of an XBee module, 13 GPIO pins that allow for remote actuation or status acquisition and a 4-channel ADC using some of those pins. This functionality works without the need of an external MCU.

5.3.3 ZigBee networks

ZigBee defines three different device types [33]: coordinator, router, and end devices.

A coordinator has the following characteristics: it

- Selects a channel and Personal Area Network (PAN) Identification (ID) (both 64-bit and 16-bit) to start the network
- Can allow routers and end devices to join the network
- Can assist in routing data

- Cannot sleep. Should be mains powered.

A router has the following characteristics: it

- Must join a ZigBee PAN before it can transmit, receive, or route data
- After joining, can allow routers and end devices to join the network
- After joining, can assist in routing data
- Cannot sleep. Should be mains powered.

An end device has the following characteristics: it

- Must join a ZigBee PAN before it can transmit or receive data
- Cannot allow devices to join the network
- Must always transmit and receive RF data through its parent. Cannot route data.
- Can enter low power modes to conserve power and can be battery-powered.

An example of such a network is shown in figure 5.9.

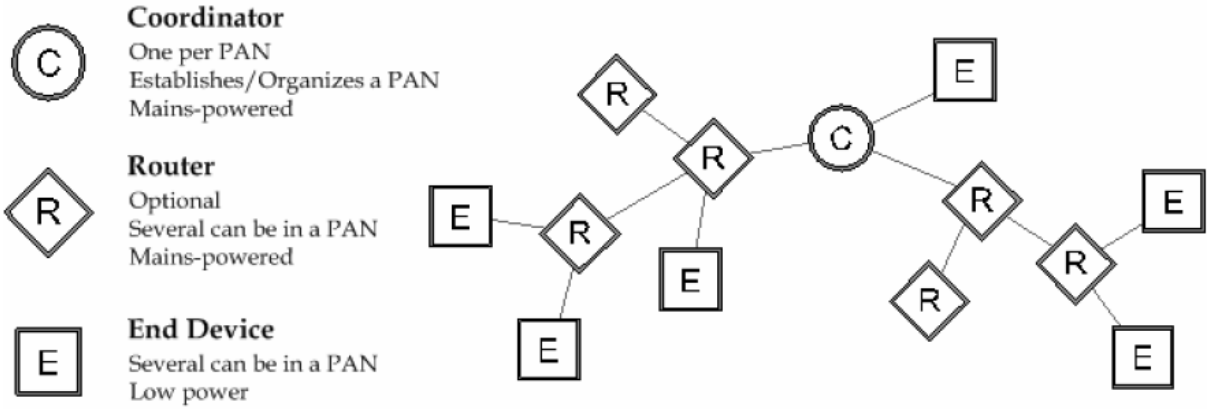


Figure 5.9: ZigBee network architecture

5.3.4 XBee operation

For more efficient integration of the XBee module with PSoC, API operation mode has been selected as the only mode supported. API firmware is recommended by the manufacturer when a device:

- sends RF data to multiple destinations.
- sends remote configuration commands to manage devices in the network.
- receives IO samples from remote devices.
- receives RF data packets from multiple devices, and the application needs to know which device sent which packet.

- must support multiple ZigBee endpoints, cluster IDs, and/or profile IDs.
- uses the ZigBee Device Profile services.

For API operation, communication between PSoC and XBee is wrapped on packets with a determined structure. The frame format is depicted in figure 5.10.

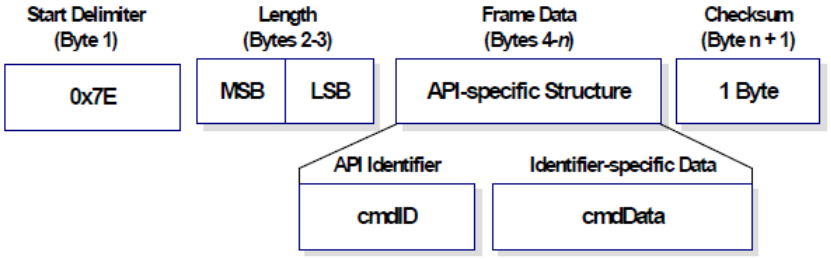


Figure 5.10: XBee API mode frame format

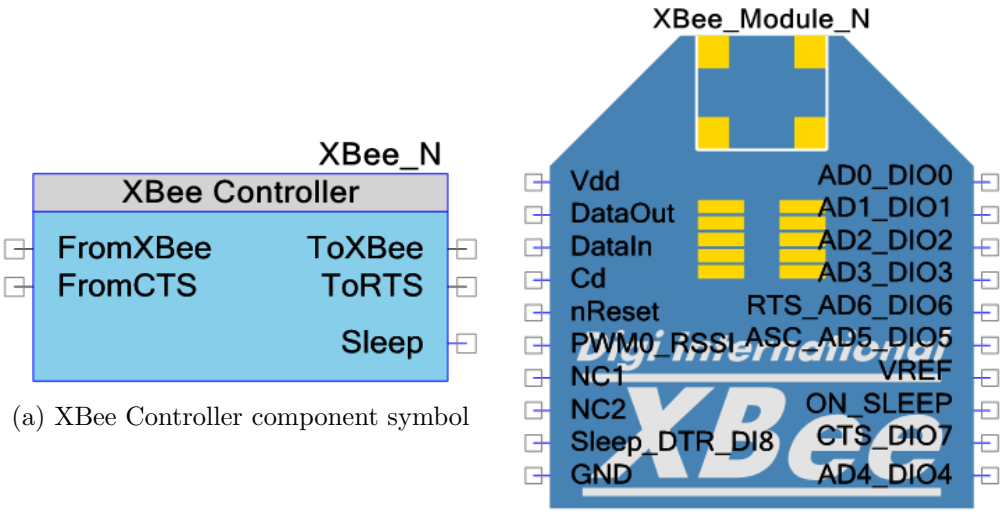
5.3.5 XBee custom component

To integrate XBee functionality with PSoC there is nothing like a custom component:

1. It encapsulates all XBee functionality in a single drag-and-drop component.
2. It is portable between PSoC projects. This is presumably useful for the heterogeneity of nodes.
3. Updates can be done to the component with no further changes to the project itself most of the times.

Components symbols

For clarity, an *XBee Controller* and an *XBee Module* have been created (figure 5.11). The XBee Controller (5.11a) encapsulates all the functionality, while the XBee Module (5.11b) is just for annotation on the schematics.



(a) XBee Controller component symbol

(b) XBee Module component symbol

Figure 5.11: XBee components

Typical Application

Figure 5.12 illustrates the typical configuration both modules will have in a schematic on PSoC Creator. Only the XBee Controller component is needed to operate an XBee module.

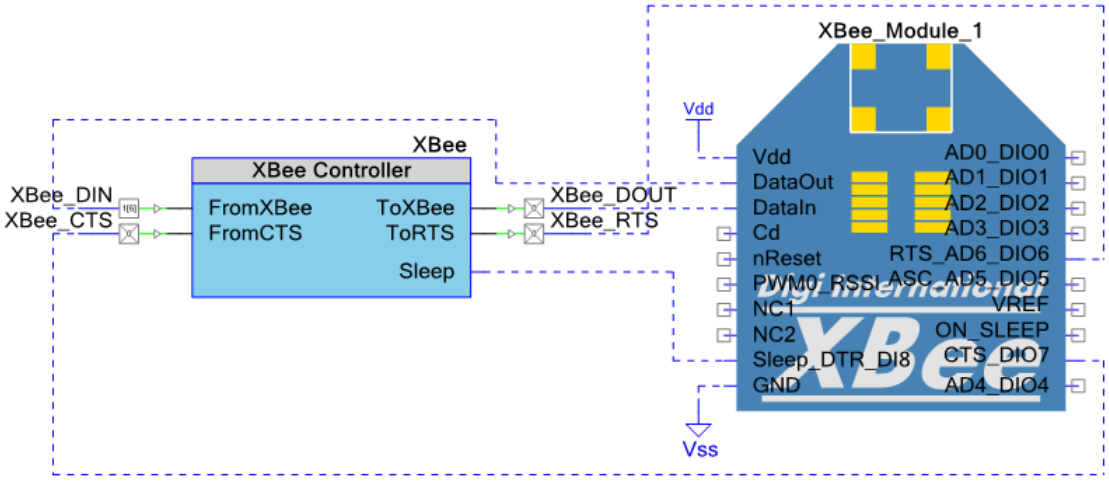


Figure 5.12: XBee typical application

The XBee Controller

The XBee Controller consists on an UART component to transmit and receive data from the module, an interruption to manage the packet reception, and a buried pin for the *sleep* function. Figure 5.13 illustrates the internal schematic of the XBee Controller.

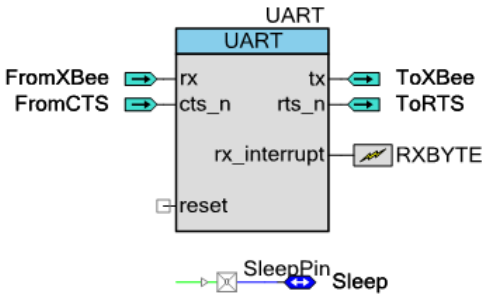


Figure 5.13: XBee Controller internal schematic

How it works

The component provides an API to operate with the XBee modules, abstracting the user from the inner workings of packets and communication.

From PSoC to XBee With simple function calls the firmware can transmit data and change module parameters. There are specific calls for each function.

The *sleep* functionality has a *buried pin*. A buried pin is instantiated inside a component. The user of the component can not operate the pin itself, just assign it to a physical pin on PSoC. This pin has been configured this way so the firmware can operate it directly, the only way to keep it driving when PSoC enters one of its power-saving modes. Power management is discussed on chapter 6.

From XBee to PSoC When XBee receives a packet or has to reply to its host, it sends it through the *DataOut* pin. In the XBee Controller an interrupt is triggered each time a byte is received from the module, buffering the packet this byte by byte. When the end of the packet has been reached, a user-defined callback function is called to perform the appropriate tasks.

6

Power management on PSoC

6.1 Power System

6.1.1 Supply voltage range

PSoC devices can run from a voltage source as low as $0.5V$ with full analog capabilities. Figure 6.1 illustrates the full working range and typical power sources [21] [23].

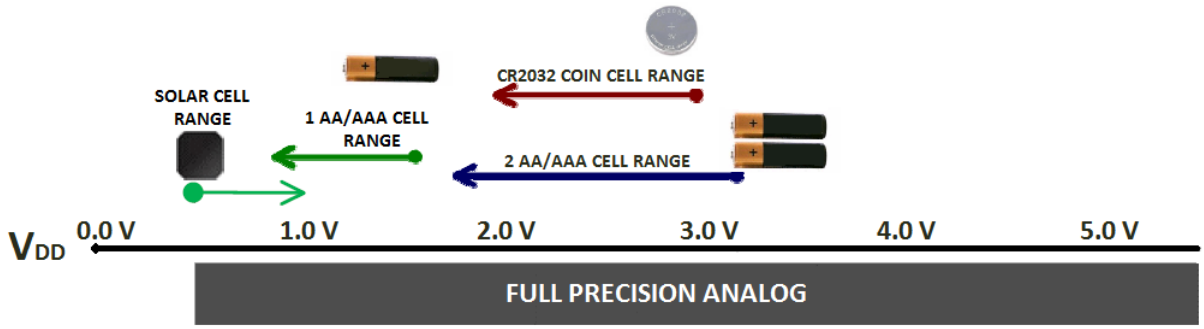


Figure 6.1: PSoC Supply Voltage Range

6.1.2 Power Distribution

PSoC devices have separate external analog and digital supply pins, labeled `Vdda` and `Vddd`, respectively. The devices have two internal $1.8V$ regulators that provide the digital (`Vccd`) and analog (`Vcca`) supplies for the internal core logic.

Figure 6.2 illustrates the power distribution on PSoC.

`Vdda`
: Must be highest voltage in system. Supplies analog high voltage domain and core regulator.

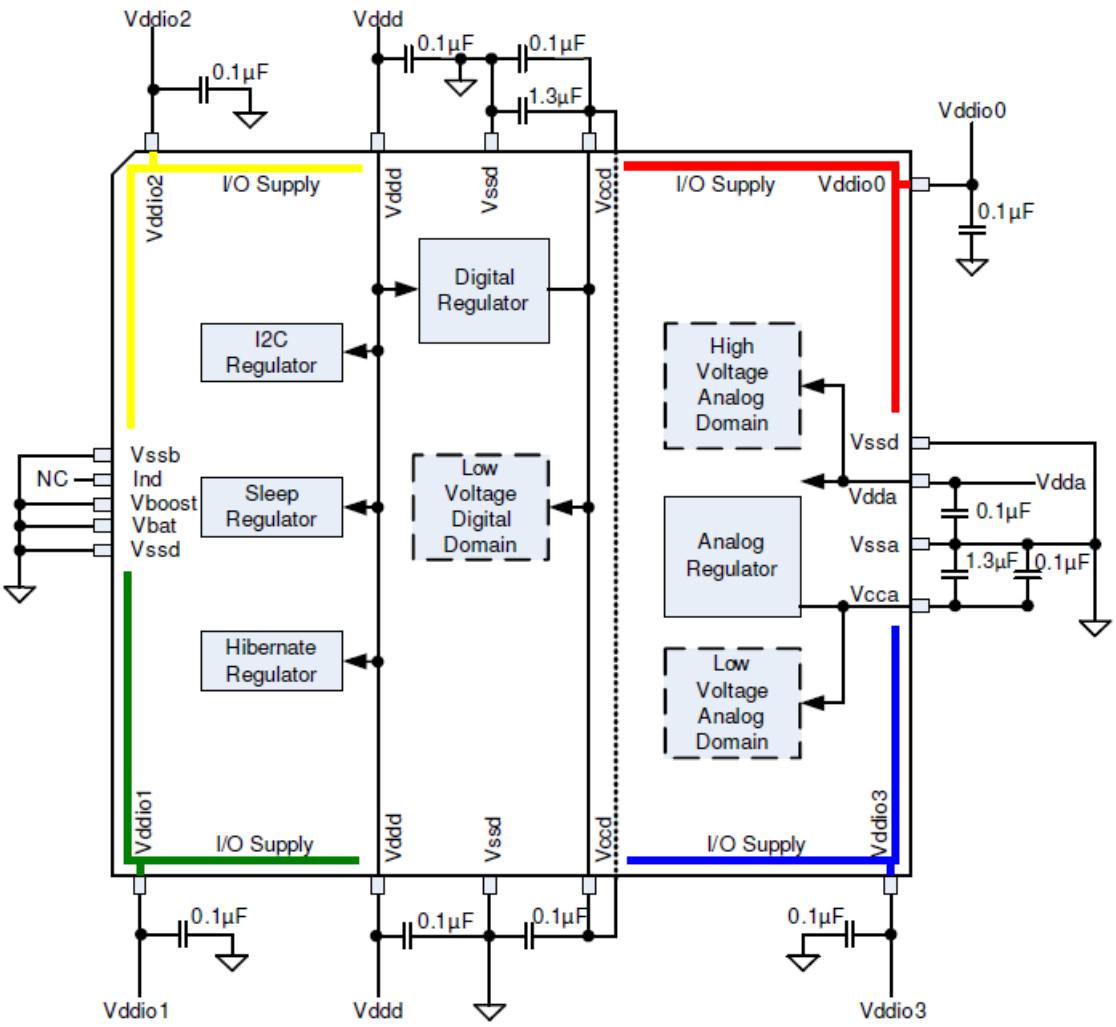


Figure 6.2: PSoC Power Distribution

`Vddd`
: Supplies digital system core regulators.

`Vcca`
: Output of the analog core regulator.

`Vddio0/1/2/3`
: Independent I/O supplies. May be any voltage in the range of $1.8V$ to `Vdda`.

6.1.3 Boost Converter

The boost converter enables input voltages that are lower than the desired system voltage to be boosted to the desired system voltage level.

The boost converter is able to power the $1.8V$ PSoC3 or $1.9V$ PSoC5LP cores from a source as low as $0.5V$. It can also be used to power external devices up to $5.25V$ in while the system runs at lower voltage.

6.2 Power modes

This section describes the power modes available on PSoC3 and PSoC5LP with the goal of reducing the average power consumption of the device.

The available power modes [21] [23], in order of decreasing power consumption, are:

- Active
- Alternate Active
- Sleep
- Hibernate

Active and alternate active are the main processing modes, and the list of enabled peripherals is programmable for each mode.

Sleep and hibernate modes are used when processing is not necessary for an extended time. All subsystems are automatically disabled in these two modes, regardless of the settings in the active template register. Some subsystems have an additional available bit (`PM_Avail_CRx`) that can mark a subsystem as unused and prevent it from waking back up. This reduces the power overhead of waking up the part, in that not all subsystems are repowered.

The allowable transitions between power modes are illustrated in figure 6.3.

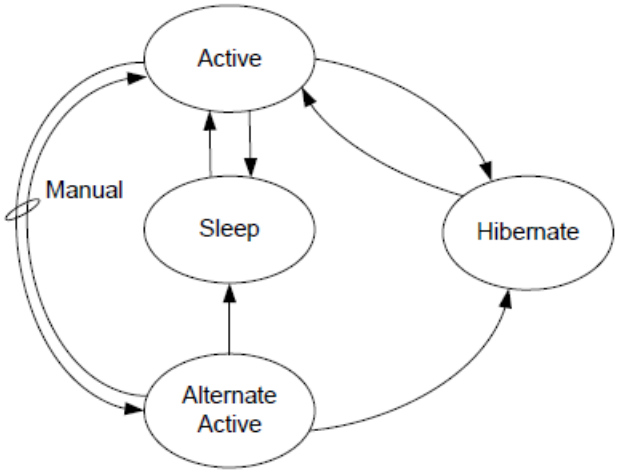


Figure 6.3: State diagram of allowable power transitions

6.2.1 Active mode

Description

Active mode is the primary power mode of the PSoC device. This mode provides the option to use every possible subsystem/peripheral in the device. Firmware may be used to dynamically enable or disable subsystems.

Current

With all subsystems turned off: PSoC3: Starting at $1.2mA$ @$6MHz$
PSoC5LP: Starting $2mA$ @$6MHz$

Entry Condition
Wakeup, reset, manual register entry. Any pending wakeup source prevents the device from exiting active mode.

Wakeup Source
Any interrupt.

Clocks
Any (programmable).

Regulator
All regulators available. Digital and analog regulators can be disabled if external regulation used.

6.2.2 Alternate Active mode

Description
Similar to Active mode, and is typically configured to have fewer peripherals active to reduce power. One possible configuration is to use the UDBs for processing, with the CPU turned off. The essential difference between active and alternative active mode is that the device cannot wake up from sleep/hibernate mode into the alternative active mode.

Current
This mode is meant to enable only some subsystems, so the measure depends on the configuration.

Entry Condition
Manual register entry.

Wakeup Source
Any interrupt.

Clocks
Any (programmable).

Regulator
All regulators available. Digital and analog regulators can be disabled if external regulation used.

6.2.3 Sleep mode

Description
Sleep mode powers down the CPU and other internal circuitry to reduce power consumption. When a wakeup event occurs, the system reactivates in a single phase and returns to active mode.

Current
PSoC3: $1\mu A$
PSoC5LP: $2\mu A$

Entry Condition
Manual register entry.

Wakeup Source
Comparator, Port Interrupt Control Unit (PICU), I^2C, RTC, CTW, or reset event.

Clocks
ILO / kHzECO.

Regulator
The analog and digital Low-dropout (LDO) regulators are disabled during sleep mode. If the core supplies are configured for internal regulation, a weak keeper is used to hold the external capacitors at $1.8V$ (nominal). Both regulators can be periodically activated (buzzed) to provide supervisory features for voltage monitoring and brownout detect.

6.2.4 Hibernate mode

Description
Hibernate mode consumes/dissipates the lowest power, and nearly all internal functions are disabled. Configuration state and all memory contents are preserved in hibernate mode. GPIOs configured as digital outputs maintain their previous values, and pin interrupt settings are preserved. The voltage used to retain state is lower than the nominal core voltage.

Current
PSoC3: $200nA$
PSoC5LP: $300nA$

Entry Condition
Manual register entry.

Wakeup Source
PICU, comparator, or reset event.

Clocks
None.

Regulator
There is no buzzing, and the external capacitors are permitted to discharge. The hibernate-regulator is always active to generate the keep-alive voltage (V_{pwrka}) used to retain the system state.

7

Node design considerations

7.1 Internal analog routing

All PSoC3 and PSoC5LP devices can be divided into an analog and digital section [11] [12].

The top section of the silicon is mostly analog and the bottom section is digital. The analog section consists of several analog blocks such as the Delta-Sigma ADC (DSM), comparators, DACs, and SC/CT blocks. The digital section contains the CPU, RAM, ROM, DMA, UDBs, Clocks, and so on.

The entire chip is surrounded by pins. Most of these pins are general purpose I/O or GPIO pins. All GPIO pins can be configured for eight different modes (see section 3.1.5): seven digital and one analog input/output mode.

PSoC have up to seven full 8-pin ports or 56 GPIOs that can be used for analog input and output, an additional port populated with SIOs, and another port with 6 GPIOs and the USB data lines (which can act as digital GPIO). Some of these pins share functionality with fixed modules, such as the debugger, External Crystal Oscillators inputs, OpAmps dedicated pins and more to be reviewed in the sections below.

This section studies the functionality of possibilities of the I/O and makes a proposal on pins distribution when designing the node.

7.1.1 Analog routing network

Analog Globals (AGs) and Analog Multiplexer Buses (AMUXBUSes) provide analog connectivity between GPIOs and the various analog blocks [12]. As shown in figure 7.2, there are 16 AGs that are divided between four quadrants. Each quadrant contains four analog globals: (`AGR[7:4]`, `AGR[3:0]`, `AGL[7:4]`, and `AGL[3:0]`). The AMUXBUS may be connected to any of the GPIO pins and most of the analog block inputs and outputs. It may be divided between the left and right half of the chip or configured as a single bus that wraps around the entire chip. Together, the Analog Multiplexer Bus and Analog Globals provide up to 18 paths between

the analog blocks and GPIO pins. Additionally, there are about 20 dedicated paths between GPIOs and analog blocks. The dedicated routes provide low resistive paths between GPIOs and devices such as current DACs and uncommitted OpAmps.

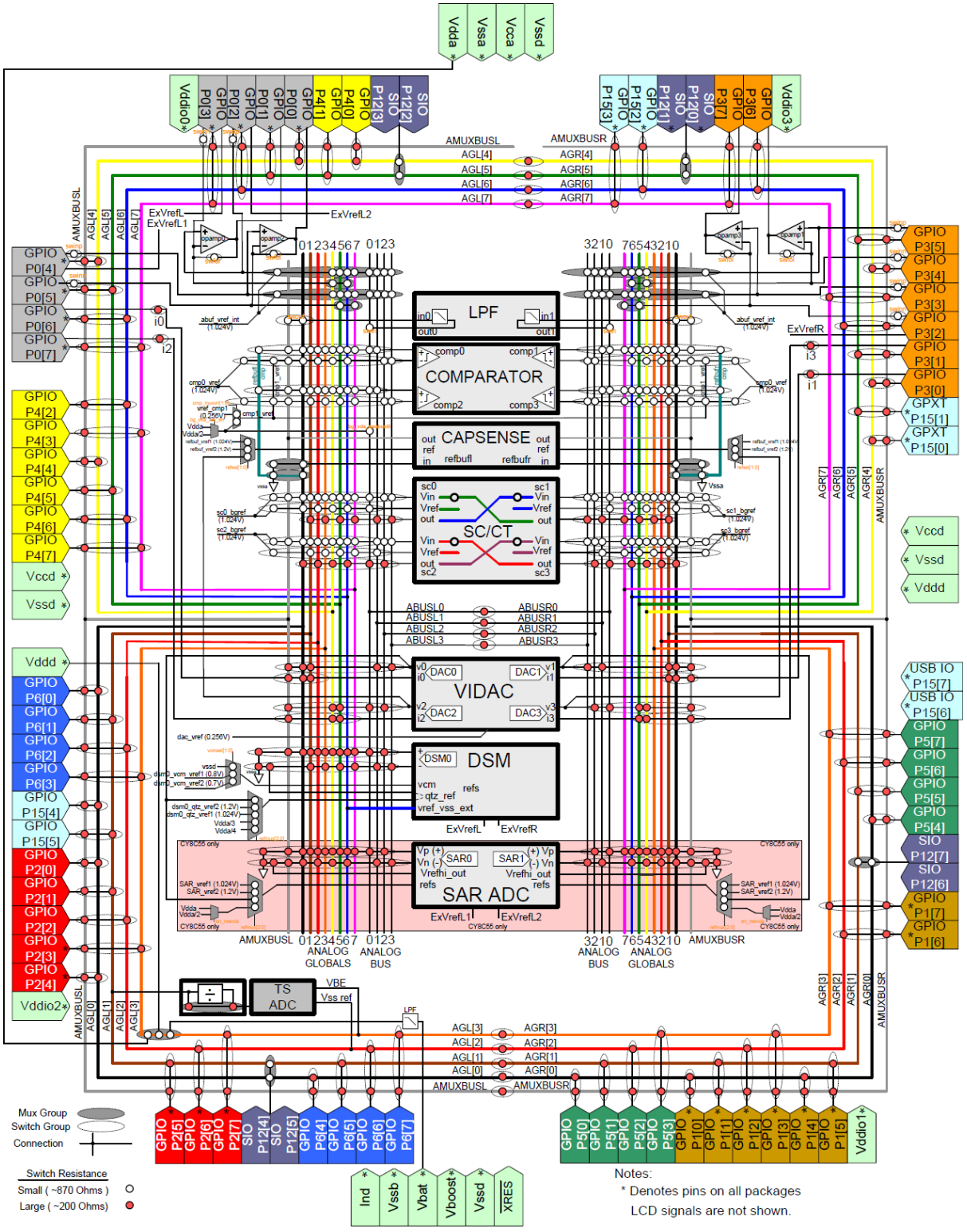


Figure 7.1: PSoC3 and PSoC5LP Analog Interconnect Diagram

7.1.2 Best analog ports

Of all pin-ports on a PSoC, three have a slight analog performance advantage `P0[7:0]`, `P3[7:0]`, and `P4[7:0]`. These ports reside in the analog upper portion of the chip. The analog globals, `AGL[7:4]` and `AGR[7:4]` that connect to these ports, also reside only in the upper analog section of the parts, which give these ports a slight signal-to-noise.

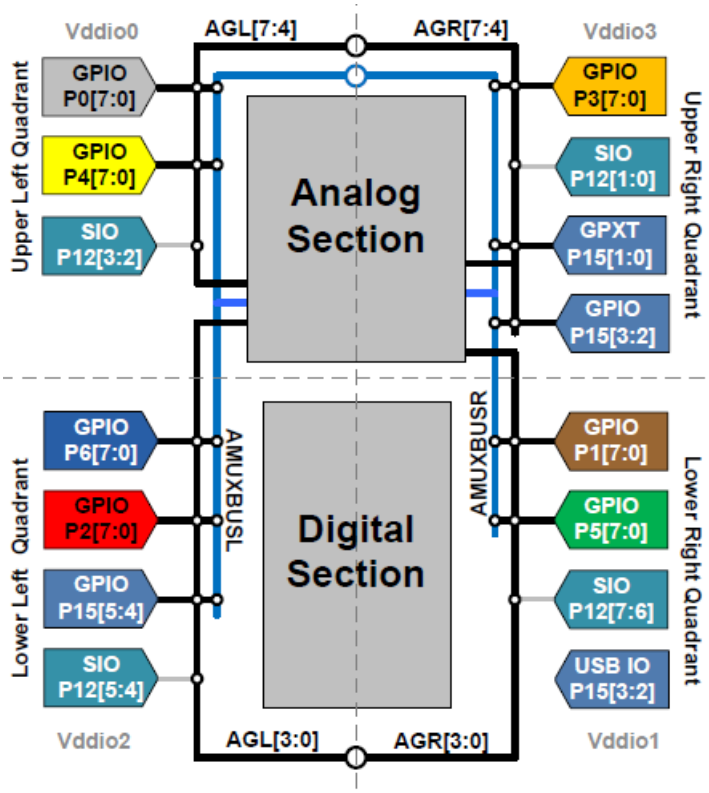


Figure 7.2: PSoC3 and PSoC5LP Analog/Digital Layout

7.1.3 Direct Routes to GPIO pins

The signal paths and switches required to provide the flexibility PSoC offers add some resistance between the signal source and its destination in the order of hundreds of Ω [13]. Many times the currents flowing are so small that the voltage drops in these switches can be ignored. However, when dealing with current sources or inputs that do not have very input resistance, extra attention should be paid to the signal path and any voltage drops that may occur. Table 7.1 illustrates some examples of typical input resistances to analog blocks.

To circumvent this issue, several direct routes bypass the AMUXBUS, Analog Globals, and the analog local bus to connect some of these analog blocks directly to GPIOs. The OpAmps and VIDACs both have direct connections that are used to maximize performance. Figure 7.3 is a simplified section of the analog routing diagram that shows some of these direct connections.

Analog Block	**Input Resistance**	**Affected by Routing Resistance**
DelSig ADC (Buffered)	$> 100M\Omega$	No
DelSig ADC (Unbuffered)	$> 80K\Omega$	Input resistance is a function of ADC clock and input capacitor
OpAmp	$> 100M\Omega$	No
PGA	$> 100M\Omega$	No
Inverting PGA	$20K\Omega$ or $40K\Omega$	Yes
Mixer	$20K\Omega$ or $40K\Omega$	Yes
SAR ADC	$> 150K\Omega$	Input resistance is a function of the sample rate
Comparator	$> 100M\Omega$	No
TIA (Iin input)	$< 10\Omega$	Yes, routing resistance adds to the input resistance
TIA (Vref input)	$> 100M\Omega$	No

Table 7.1: Typical input resistance to some analog blocks

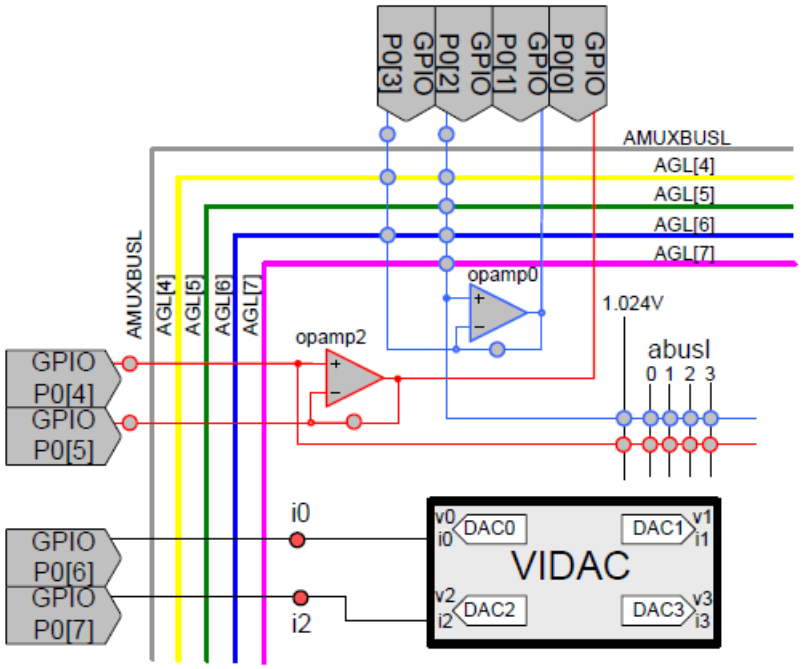


Figure 7.3: Dedicated GPIO connections

Current DACs

Each IDAC output may be routed to a GPIO using the Analog Globals or the AMUXBUS. This works fine as long as the maximum current is limited to the two lower ranges of $32\mu A$ or $255\mu A$. When using the high current range of $2mA$, the resistance in the path and analog switches (which can be 500Ω or higher) may cause an excessive voltage drop and limit the compliance voltage of the current source at the pin.

To minimize this voltage drop, one of the dedicated pins listed in table 7.2 should be used

to connect the output of the IDAC. This dedicated route is less than 200Ω to GPIO.

Once one of these pin is assigned, PSoC Creator selects the appropriate IDAC to make sure that connection is made.

DAC	GPIO pin
DAC0	P0[6]
DAC1	P3[0]
DAC2	P0[7]
DAC3	P3[1]

Table 7.2: Dedicated IDAC connections

Operational amplifiers

After the Operational Amplifier is enabled, the dedicated output GPIO will always be driven by the output of the OpAmp, even if this signal is only used internally. This guarantees that the resistance between the OpAmp and GPIO pin is low, about 5Ω. In the case the OpAmp is used as a buffer for internally generated signals, the dedicated input pins may be ignored and use for another purpose. A representation of this particular connection can be appreciated in the already seen figure 7.3 and a summary dedicated GPIO to the OpAmps is shown in table 7.3.

OpAmp	GPIO non-inverting input	GPIO inverting input	GPIO output
OpAmp0	P0[2]	P0[3]	P0[1]
OpAmp1	P3[5]	P3[4]	P3[6]
OpAmp2	P0[4]	P0[5]	P0[0]
OpAmp3	P3[3]	P3[2]	P3[7]

Table 7.3: Dedicated OpAmp connections

Voltage Reference

External references can be used to enhance the accuracy or change the range of the ADCs. There are connections for external references for both the Delta-Sigma and SAR ADCs.

The internal reference is accurate up to 0.1% over the operating temperature range depending on the device selected. An external reference can increase the accuracy beyond 0.1%.

The same pins for the Delta-Sigma ADC external reference may be used for bypassing the internal reference voltage with a filter capacitor.

Table 7.4 summarizes the pins dedicated for external voltage references. Note that these pins are shared with some OpAmps dedicated inputs.

ADC	GPIO pin
Delta-Sigma	P0[3] & P3[2]
SAR1	P0[4]
SAR2	P0[2]

Table 7.4: Dedicated IDAC connections

7.1.4 Connecting to the Delta-Sigma ADC

The analog blocks are equally distributed between the left and right sides of the chip. Currently, there is just one Delta-Sigma ADC, which is placed on the left side and only has direct connections to the routing resources on the left side. Signals may be routed from the right side of the part, but only by connecting the left and right analog globals together. When pins from the right side of the part must connect to the ADC, the analog globals on the right side must be connected to the corresponding analog globals on the left side.

For example, if pin P3[7] needs to be routed to the ADC, the pin is first connected to AGR[7]. Then AGL[7] must then be connected to AGR[7] to route to the ADC. However, if pin P0[7] is selected, then only analog global AGL[7] is used instead of both AGR[7] and AGL[7].

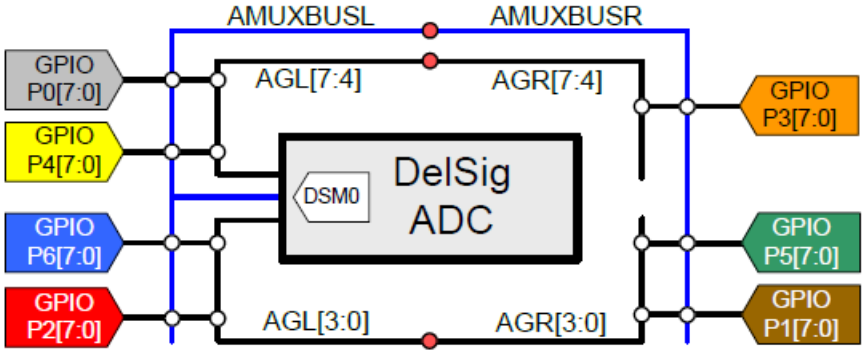


Figure 7.4: Delta-Sigma ADC Analog Connections

7.2 External routing on the node

Recalling the architecture of the node proposed on section 2.4, sensors are independent and interchangeable devices on the main board. Thus pin arrangements and resources reservation for these devices have to be performed with no prior knowledge of the sensors or communication interfaces involved.

Sensing platforms will mostly use analog resources on PSoC such as the ADCs, OpAmps and DACs; and low speed digital signals for circuit switching and digital sensors. It is also safe to assume that communication interfaces and power modules will not require the highest analog performance, if they need any analog resource at all.

Reviewing the main resources on the device:

- Seven 8-pin ports.

- Four OpAmps.
- Four Comparators.
- Four DACs.
- Four SC/CT blocks.
- Two SAR ADCs with external reference inputs.
- One Delta-Sigma ADC with external reference and bypass connection.
- One Digital Filter Block (DFB).
- One CapSense block.
- Eight SIO pins.

OpAmps, DACs, ADCs reference voltages, and SIOs, have dedicated pins. Ports `P0` and `P3` are completely populated with pins of the first three types, while P12 features the eight SIO pins. Each of these ports connects to two OpAmps, two DACs and one reference voltage for the Delta Sigma ADC. The voltage reference inputs for the SAR ADCs are present only on port `P0`. Both ports will be available for signal conditioning circuits.

To make all slots, hereinafter bays, pinout-compatible enabling the daughter cards to be exchangeable among them, 2 ports can be routed to each slot giving 3 full bays. Two for sensing platforms and one for the communication interface. The power module will presumably need fewer GPIOs. This way bays are enabled to adopt other functionality than the initially intended, e.g. a mote could have two communications interfaces and behave as a bridge between networks.

A system I^2C bus can be routed to the slots. This allows the PSoC to communicate with *intelligent* devices in the daughter cards. This feature is described in section 7.2.4.

7.2.1 External routing to sensors

Port `P4` is placed on the analog region of the silicon, and thus have a slightly better signal-to-noise ratio than others. Ports `P0` and `P3` have dedicated pins to analog functions, so it is reasonable to use these ports for the sensing bays. Since `P4` is in the left side of the silicon, `P5` could be the remaining port for sensing. `P5` is located in the right side of the chip, in the lower-right quadrant, maximizing the use of analog routing resources in the device.

There are enough resources for two *small* sensor platforms, or a big one monopolizing all resources. The distribution of the pins among these bays is summarized in tables 7.5 and 7.6.

PIN	Functions
P4[7:0]	A/D GPIO on the upper-left quadrant for precision analog
P0[7:0]	A/D GPIO + OpAmps 0 and 2 (left side) + DACs 0 and 2 (left side) + External voltage reference inputs for all ADCs
P12[1:0]	System bus
P12[3:2]	SIO pins on the upper-left quadrant of the silicon
Vdd	Polarization voltage from system regulator ($3.3V$)
Vssd	Return path for digital signals
Vssa	Return path for analog signals
Vbat	Unregulated voltage from power source for high current drain
GND	Ground pin from power source

Table 7.5: Pinout proposal for Sensing Bay 0

PIN	Functions
P5[7:0]	A/D GPIO on the lower-right quadrant
P3[7:0]	A/D GPIO + OpAmps 1 and 3 (right side) + DACs 1 and 3 (right side) + External voltage reference input for Delta-Sigma ADC
P12[1:0]	SIO pins in the lower-right quadrant
Vdd	Polarization voltage from system regulator
Vssd	Return path for digital signals
Vssa	Return path for analog signals
Vbat	Unregulated voltage from power source for high current drain
GND	Ground pin from power source

Table 7.6: Pinout proposal for Sensing Bay 1

With this distribution, *Sensing Bay 0* has been optimized for high-precision analog, having the best placed pins on the silicon and external voltage reference inputs for all ADCs. Most applications will not require this degree of precision, and will have available the *Sensing Bay 1*.

SIO pins have to be placed close to the other GPIO ports used for each bay. SIOs are strong-drive digital pins, and these should not be placed close to analog routes to avoid crosstalk. However, being these pins (which are already close to the other GPIOs in the silicon) in the same sensing bay, the user can determine their behavior, so their state changes do not affect analog performance. If these SIOs are routed to any other function, the user would be able to do little to minimize the crosstalk.

A daughter card made for one bay is absolutely interchangeable with the other as long as it does not make use of the SAR ADCs external voltage reference feature (if it does, the reference voltage for SAR ADCs will do nothing).

7.2.2 External routing to communication interfaces

The bottom half of the silicon is mainly digital. Communication interfaces will make the most use of it, probably with high speed digital signals running back and forth. Rarely a communication module will have analog needs, and if it does, it will not require the highest

precision. Based on this, P6 and P2 are ports placed in the lower-left quadrant, isolated from those reserved for sensing.

PIN	Functions
`P2[7:0]`	A/D GPIO on the lower-left quadrant
`P6[5:4]`	A/D GPIO on the lower-left quadrant
`P12[5:4]`	SIO pins + Wake-on-I^2C in the lower-left quadrant
`Vdd`	Polarization voltage from system regulator
`Vssd`	Return path for digital signals
`Vssa`	Return path for analog signals
`Vbat`	Unregulated voltage from power source for high current drain
`GND`	Ground pin from power source

Table 7.7: Pinout proposal for Communications Bay

Table 7.7 summarizes the pinout of the communications bay. This pinout includes a total amount of 16 GPIO pins and the dedicated SIOs for wake-on-address-match using the I^2C, which can be also used as regular SIOs.

7.2.3 Routing to power sources

Power sources do not always stick to the classic ± connector. For optimal power management some control may be required such as battery or power source temperature and voltage monitoring, fans, DC-DC converters, or solar panel controllers.

A low-dropout voltage regulator can be included on-board as the system regulator for default configurations. Having independent regulators for analog and digital domains is encouraged [11], though this may not be practical because of its cost and additional power consumption. Including mechanisms to disable the use of the on-board regulator(s) allows power modules to feature their owns.

Regarding the monitoring and control, port `P1` can be reserved. `P1` is placed in the lower-right quadrant of the silicon and features 8 GPIO and shares functionality with the SWD connector for programming and debugging.

PIN	Functions
`P1[7:0]`	A/D GPIO + SWD on the lower-right quadrant
`Vbat`	Unregulated voltage to the on-board regulator
`GND`	Ground reference from power source
`Vddio`	Voltage to the `Vddio` (regulated by power module)
`Vdda`	Supply for analog peripherals and analog core regulator (regulated by power module)
`Vddd`	Supply for digital peripherals and digital core regulator (regulated by power module)
`Vssd`	Return path for digital signals
`Vssa`	Return path for analog signals

Table 7.8: Pinout proposal for Power Bay

7.2.4 On-board purposes

This section describes the pins reserved for on-board purposes.

Port `P15`

Port `P15` has several pins dedicated for second purposes reviewed in the following paragraphs. These pins can be used as GPIOs when no special function is taking place.

USB

USB dedicated pins identified as `P15[6]` (data +) and `P15[7]` (data -) should must go to an on-board micro-USB female connector. The routing of these lines to daugther cards might not be liable because of the high-speed of the bus.

These pins can be used as digital GPIOs when not used for USB.

These pins are also used as an alternative connection for SWD programming and debugging [20]. Special care should be taken with these pins when the device is being programmed using the dedicated pins in `P1`.

External Crystal Oscillators

There are two dedicated inputs for External Crystal Oscillators (ECOs): `P15[1:0]` for MHzECO and `P15[3:2]` for kHzECO.

A high precision 32-kHz External Crystal Oscillator connected to kHzECO input is the preferred method to keep track of time on PSoC during sleep states [24]. Since nodes are presumably keeping track of time for periodical wakeups, a high-prescision crystal could be included on-board.

The MHzECO is not used for particular purposes, and thus the pins could be held unconnected for future inclusion of the desired crystal by the user.

These pins can be used as GPIOs when not used by ECOs.

System I^2C bus

`P12[1:0]` are SIOs located in the upper-right quadrant of the silicon with wake-on-address-match I^2C capability.

A system bus can be implemented on the board linking sensing, communication, and power module bays for external communication using an I^2C bus. For example, a sensing platform could feature an I^2C EEPROM where calibration data is stored, and via a system bus these data could be read by the PSoC and external programmers in a standardized way among daughter cards.

These pins can be used as regular SIOs when they are not working as I^2C.

General Purpose Header

A general purpose header can be included on-board to allow the use of all GPIOs described in this section when they are not being used for their particular purposes. This would allow the user to connect specific application devices such as a hardware RTCs, on-board sensors, and so forth, without occupying an entire bay.

All these pins are summarized in table 7.9.

PIN	Functions
`P12[1:0]`	System Bus
`P15[7:6]`	GPIO + USB
`P15[5:4]`	GPIO
`P15[3:2]`	kHzECO
`P15[1:0]`	GPIO + MHzECO

Table 7.9: Pins assigned to on-board functions

8 Conclusions and future work

8.1 Technical

There are several features of PSoC making it ideal for the case of this highly flexible wireless mote:

The internal routability or like Cypress advertises *any pin, any purpose*, is absolutely great for a node where sensors and communication options are not pre-arranged nor even purpose intended. Any pin may route any signal to any block, either analog or digital (with the restrictions reviewd in 3.1.5). With the powerful *DSI*, precious resources such as the Delta-Sigma ADC can be shared between multiple pins no matter their placement on the chip.

The configurable analog blocks have an excellent sensitivity for small input signals. Besides, they reduce the R&D time and BOM by boosting the prototyping process and minimizing the number of external components required for signal conditioning.

Regarding the UDB-based components the main advantage is the non-restriction to a pre-defined number of peripherals of given types. This implies that any external modules (either digital sensors, communication interfaces or any other device) are able to interface PSoC no matter which communication protocol they use. As an example for a use case, four UARTs may be needed; for any other it could be useful to have two SPIs, one with full bandwith for the communication interface and the other with a narrower bandwith for some digital sensors.

8.1.1 Signal conditioning

As it has been stated, PSoC has great capabilities for analog signal conditioning. In this document some simple circuits have been reviewed, but the possibilities go way beyond. When required by the application, fine grained readings can be acquired making complex excitation and conditioning circuits without adding off-chip components to the BOM.

In the thermistor example, the `Vddio` has been used to excite the sensor, this is enough since the calculation method eliminates any deviation, and the thermistor has a coarse precision. But in the case of a high precision resistive sensor, a Voltage DAC could be used buffered by a voltage follower, using exactly the same amount of external parts.

The SC/CT block has also proven very powerful, integrating active conditioning resources. The TIA is so precise that it can output notable voltages from currents in the order of tens of nanoamperes. The mind-breaking problem will be having an error-free circuit. Cypress provides the required documentation and tools to calculate deviations due to internal routing and components precision.

Everything has a cost though. And all this precision in a low-power device has the downside of making the PSoC unable to handle big signals, by the order of several milliamperes or over the 5 volts. For signals above these thresholds external conditioning is required. Anyway for a low-power sensor node, this generally will not be a problem, since the sensors themselves will be as low-power as possible.

8.1.2 Digital sensors

Regarding digital sensors, PSoC UDBs-based components offer great technical advantages over software-driven solutions typical from classic MCUs:

Power consumption: UDB-based components are more likely to consume less power than software-driven solutions.

Task scheduling: Several components may work in parallel at the same time. Plus, during their operation, the CPU is free to carry whatever tasks it needs to.

Portability: As already seen, PSoC components provide full encapsulation of their functions enabling drag-and-drop portability between projects or instantiating several components of the same type.

However, one should keep in mind that developing custom components is a time-consuming task, probably several times longer than their software-driven counterparts, and requires a set of different skills on both hardware and software fields that altogether may make their development unpractical for some applications. It is up to the developers to evaluate the benefits and withdraws of both implementations on a case by case basis.

8.1.3 Communications

Sensor nodes connected to CAN buses are common in industrial environments, for which PSoC has a fixed function block. The RS-485 is also very common in the same environment, and PSoC fulfills this use case with an UDB-based UART component from the Cypress Components Catalog.

For wireless communication, seamless integration of a ZigBee module has been performed. Then again proving itself valid for industrial use.

And talking about chip-to-chip communications, I^2C, SPI and UART components have been reviewed, very common protocols for on-board or board-to-board communication.

For a full list of components ready to work out-of-the-box, refer to the Cypress Components Catalog in the Appendix A. And, in case the component you are looking for is not there, remember that you might be able to implement it using PSoC configurable resources reviewed on section 3.3.

8.1.4 Power Management

Despite the high flexibility and computing power of PSoC, its power consumption remains fairly low. Consuming up to $300\mu A$ in hibernate mode, the device can wake-up on any GPIO, and retain the state of the CPU and SRAM.

Off-loading the CPU from tasks that can be performed by UDB-based components is one of the most remarkable power-saver features when used during the *Alternate Active* state with CPU off.

The boost technology allows the system to be powered on as little as $0.5V$ with full analog capabilities. Ideal for single cell battery operation.

8.2 User Experience

8.2.1 PSoC Creator

PSoC Creator is a friendly and powerful tool, provided free of charge without code size limitations. Paid licenses would have set this project off of its goals.

It generates all the API files to operate the device, so the user does not have to deal with low-level register operation but for the special cases. The learning curve rises quicker compared with other MCUs. PSoC is much easier and faster when it comes to firmware development, while encouraging portability and maintenance.

Custom components offer the ideal degree of encapsulation and portability of new functions that an open-source project requires.

8.2.2 External Software

External software, not included, is required for some tasks.

Simulation of Verilog-based components

PSoC Creator does not integrate any simulation tool for custom components described with Verilog, hence a third party simulator is needed. These simulators tend to be very expensive if the user is not affiliated to an educational institution, and free solutions are often poorly featured.

Component customizers development

The component customizer dialog box has to be described using C♯. It is possible to describe it from PSoC Creator, but Visual Studio approaches the design of the GUI in a drag-and-drop graphical fashion, and enables all the features of a typical IDE (code completion, code

generation, error highlighting, etc.). The free version of Visual C♯ is capable of these tasks, but the standard suite of Visual Studio is required to debug the customizer.

This reliability on external sofware makes the PSoC less accessible for the target users of this project. However, their use is not a requirement to get full-fledged developments using the platform, and their tasks can be partially fulfilled using free alternatives.

8.2.3 Documentation

There is plenty of documentation, training and examples on Cypress website [25]. There are highly detailed datasheets for every component, and application notes regarding lot of them interfacing sensors and actuators.

There are some blogs edited by Cypress employees with application examples, tutorials, and more, regarding PSoC. As well as an official forum where employees also participate. Furthermore, there are some independent websites about PSoC where some manuals and help can be found.

For advanced applications the user may feel slightly alone though. PSoC3 and PSoC5LP architecture has been in the market for barely 3 years and documentation is constantly under work. It can be seen that more advanced topics are being covered by new releases.

8.3 Future work

To get the system built, further work has to be done regarding the server, the gateway and the motes.

Server-side, a deep study of Zabbix capabilities and possible customizations have to be done before having a fully functional system. Prototyping, a minimal installation has been performed and Zabbix is working for the purpose right out-of-the-box, but there are still many features that need to be studied or developed. E.g. the ability to send commands to the nodes, doable through custom scripts that Zabbix is able to trigger.

On the gateway, a one-way communication channel has been established using a Python script that controls an XBee and translates to Zabbix protocol. Further work can be performed on this script to allow for duplex communication, and even further to abstract its core from the communication interface, making it independent from a single communication interface.

Regarding the node, there is a lot of work to go on with:

1. A board has to be designed, built and verified. Some guidelines have been already suggested in chapter 7.
2. Studies on state-of-the-art sensors, communication interfaces, and power supply units have to be done, with analysis of their real-life application and PSoC suitability.
3. Construction of the daughter cards and implementation of custom components and firmware needed to make them work with PSoC.

And, to make the system actually usable, a thorough documentation library:

1. Custom component datasheets
2. Node and daughter card datasheets
3. Application notes
4. Field application examples
5. Performance analysis
6. Technical reference manuals
7. System images

Glossary of Acronyms

A/D Analog/Digital

ACU Address Calculation Unit

ADC Analog to Digital Converter

AG Analog Global

AHB Advanced Microcontroller Bus Architecture (AMBA) High-performance Bus

ALU Arithmetic Logic Unit

AMBA Advanced Microcontroller Bus Architecture

AMUXBUS Analog Multiplexer Bus

API Application Programming Interface

ARM Advanced Reduced Instruction Set Computer (RISC) Machines

AT Hayes command

BOM Bill of Materials

CAN Controller Area Network

CMOS Complementary MetalOxideSemiconductor

CPLD Complex Programmable Logic Device

CPU Central Processing Unit

CRC Cyclic Redundancy Check

CT Continuous Time

CTW Central Timewheel

DAC Digital to Analog Converter

DFB Digital Filter Block

DIN Data In

DMA Direct Memory Access

DMX512 DMX512 stage lightning control communications standard

DOUT Data Out

DP Datapath

DRC Design Rules Check

DSI Digital Signal Interconnect

DSI Display Serial Interface

DSL Digital Subscriber Line

DSP Digital Signal Processing

ECC Error Correcting Code

ECO External Crystal Oscillator

EEPROM Electrically Erasable ROM

EMIF External Memory Interface

EzI^{2}C Easy I^{2}C

FIFO First Input, First Output

FPGA Field-Programmable Gate Array

FTW Fast Timewheel

GCC GNU C Compiler

GPIO General Purpose Input/Output

GPL General Public License

GPU Graphics Processing Unit

GUI Graphical User Interface

HDD Hard Disk Drive

HDMI High-Definition Multimedia Interface

HID Human Interface Device

I/O Input/Output

I^{2}C Inter-Integrated Circuit

I^{2}S Inter-IC Sound

ID Identification

IDAC Current intensity DAC

IDE Integrated Development Environment

ILO Internal Low-Speed Oscillator

IMO Internal Main Oscillator

IPMI Intelligent Platform Management Interface

IR Infrared

IrDA Infrared Data Association

JTAG Join Test Action Group

LCD Liquid Crystal Display

LDO Low-dropout

LDR Light Dependent Resistor

LED Light Emitting Diode

LIN Local Interconnect Network

LSB Less Significant Bit/Byte

LUT Look-Up Table

MAC Multiply and Accumulate

MCU Micro-Controller Unit

MISO Master In / Slave Out

MMC MultiMedia Card

MOSI Master Out / Slave In

MSB Most Significant Bit/Byte

NRZ Non-Return to Zero

OEM Original Equipment Manufacturer

OpAmp Operational Amplifier

PAN Personal Area Network

PCIe Peripheral Component Interconnect Express

PGA Programmable Gain Amplifier

PHUB Peripheral Hub

PHY Physical

PICU Port Interrupt Control Unit

PIR Pyroelectric Infrared

PLD Programmable Logic Device

PLL Phase-Locked Loop

PRS Pseudo Random Sequence

PSoC Programmable System-on-Chip

PSRAM Pseudo SRAM

PT Product Term

PWM Pulse Width Modulation

RAM Random Access Memory

RCA Radio Corporation of America

RISC Reduced Instruction Set Computer

RF Radio Frequency

R&D Research and Development

ROM Read-only Memory

RS232 RS-232 serial communications standard

RS485 RS-485 serial communications standard

RTC Real Time Clock

RTL Register Transfer Level

RTOS Real-Time Operative System

RTR Remote Transmission Request

RX Reception

S/H Sample and Hold

SAR Successive Approximation Register

SC Switched Capacitors

SCL Serial Clock

SCLK Serial Clock

SD Secure Digital

SDA Serial Data

SDAT Serial Data

SDIO Secure Digital Input/Output

SIO Special Input/Output

SMBus System Management BUS

SNMP Simple Network Management Protocol

SoC System-on-a-Chip

SPC Serial Protocol Channel

SPI Serial Peripheral Interface

SRAM Static RAM

SS Slave Select

SSH Secure Shell

SWD Serial Wire Debug

TIA Trans-Impedance Amplifier

TRM Technical Reference Manual

TTL Transistor-Transistor Logic

TX Transmission

UART Universal Asynchronous Receiver Transmitter

UDB Universal Digital Block

USB Universal Serial Bus

USBFS USB Full-Speed

VDAC Voltage DAC

VIDAC Voltage and Current Intensity DAC

WDT Watchdog Timer

WSN Wireless Sensor Network

Bibliography

[1] *Cypress ADC SAR component datasheet.* http://www.cypress.com/?docID=41605.

[2] *Cypress I^2C component datasheet.* http://www.cypress.com/?docID=41600.

[3] *Cypress IDAC8 component datasheet.* http://www.cypress.com/?docID=41556.

[4] *Cypress OpAmp component datasheet.* http://www.cypress.com/?docID=41557.

[5] *Cypress PGA component datasheet.* http://www.cypress.com/?docID=41550.

[6] *Cypress SPI master component datasheet.* http://www.cypress.com/?docID=41619.

[7] *Cypress SPI slave component datasheet.* http://www.cypress.com/?docID=41574.

[8] *Cypress TIA component datasheet.* http://www.cypress.com/?docID=39413.

[9] *Cypress UART component datasheet.* http://www.cypress.com/?docID=41582.

[10] *Cypress VDAC8 component datasheet.* http://www.cypress.com/?docID=41581.

[11] *Cypress AN57821: PSoC3 and PSoC5LP Mixed Signal Circuit Board Layout Considerations.* http://www.cypress.com/?rID=39677.

[12] *Cypress AN58304: PSoC3 and PSoC5LP Pin Selection for Analog Designs.* http://www.cypress.com/?rID=39974.

[13] *Cypress AN58827: PSoC3 and PSoC5LP Internal Analog Routing Considerations.* http://www.cypress.com/?rID=40247.

[14] *Cypress AN66477: PSoC3 and PSoC5LP Temperature Measurement with a Thermistor.* http://www.cypress.com/?rID=49052.

[15] *Cypress AN68272: PSoC3 and PSoC5LP Customizing the Bootloader Communication Channel.* http://www.cypress.com/?rID=50230.

[16] *Cypress AN82250: PSoC3 and PSoC5LP Implementing Programmable Logic Designs with Verilog* http://www.cypress.com/?rID=69773.

[17] *Cypress Comparator component datasheet.* http://www.cypress.com/?docID=41593.

[18] *Cypress Component Author Guide.* http://www.cypress.com/?docID=41594.

[19] *Cypress Delta-Sigma ADC component datasheet.* http://www.cypress.com/?docID=39346.

[20] *Cypress Knowledge Base article: Serial Wire Debug (SWD) using USB (!I/O) pins in PSoC3/5.* http://www.cypress.com/?id=4&rID=43483.

[21] *Cypress PSoC3 Architecture TRM.* http://www.cypress.com/?docID=40001.

[22] *Cypress PSoC5 Architecture TRM.* http://www.cypress.com/?docID=39997.

[23] *Cypress PSoC5LP Architecture TRM.* http://www.cypress.com/?docID=41088.

[24] *Cypress RTC component datasheet.* http://www.cypress.com/?rID=48907.

[25] *Cypress Semiconductor website.* http://www.cypress.com/.

[26] *Cypress USB component datasheet.* http://www.cypress.com/?docID=41598.

[27] *Cypress WARP Verilog Reference Guide. Included in PSoC Creator.*

[28] *Design and development of an ad-hoc data network prototype applied to cetaceans monitoring. Ramiro Utrilla.* http://upcommons.upc.edu/pfc/handle/2099.1/12921?locale=en.

[29] *DHT11 datasheet.*

[30] *Libelium products catalog 2012.*

[31] *Raspberry Pi Website.* http://www.raspberrypi.org/.

[32] *TCRT5000 datasheet.*

[33] *XBee/XBee-PRO ZB modules operation manual.*

[34] *Zabbix Website.* http://www.zabbix.com/.

A

Cypress Component Catalog

- Analog
  - ADC
    - Delta Sigma ADC
    - SAR ADC
  - Amplifiers
    - Inverting PGA
    - OpAmp
    - PGA
    - TIA
  - Analog MUX
    - Analog Hardware Mux
    - Analog Mux
    - Analog Mux Sequencer
    - Analog Virtual Mux
  - Comparator
  - DAC
    - Current DAC (8-bit)
    - Voltage DAC (8-bit)
  - Manual Routing
    - Mux Constraint
    - Net Constraint
    - Net Tie
    - Resource Reserve
    - Stay Awake
    - Terminal Reserve

  - Mixer
  - Sample/Track and Hold
  - VRef
- CapSense
- Communications
  - CAN Controller
  - emFile for SD cards
  - I^2C
  - I^2S
  - LIN
  - SMBus/PMBus Slave
  - SPDIF Transmitter
  - SPI
  - UART
  - USBFS
  - USBMIDI
  - ISBUART (CDC Interface)
- Digital
  - Functions
    * Counter
    * CRC
    * Debouncer
    * Glitch Filter
    * PrISM
    * PWM
    * Quadrature Decoder
    * Shift Register
    * Timer
  - Logic
    * And
    * Bufoe
    * D Flip Flop
    * D Flip Flop w/ Enable
    * De-Multiplexer
    * Digital Constant
    * Logic High '1'
    * Logic Low '0'
    * Lookup Table
    * Multiplexer

    - Nand
    - Nor
    - Not
    - Or
    - SR Flip Flop
    - Toggle Flip Flop
    - Virtual Mux
    - Xnor
    - Xor
  - Registers
    - Control Register
    - Status Register
  - Utility
    - Basic Counter
    - Digital Comparator
    - Edge Detector
    - Frequency Divider
    - Pulse Converter
- Display
  - Character LCD
  - Graphic LCD Controller
  - Graphic LCD Parallel Interface
  - Resistive Touch
  - Segment Display
  - Segment LCD - Static
  - Segment LCD
- Filters
  - Digital Filter Block Assembler
  - Filter
- Ports and Pins
  - Analog Pin
  - Digital Bidirectional Pin
  - Digital Input Pin
  - Digital Output Pin
- Power Supervision
  - Power Monitor (upto 32 rails)
  - Trim and Margin (upto 24 rails)
  - Voltage Fault Detector (upto 32 rails)

  - Voltage Sequencer (upto 32 rails)
- System
  - Boost Converter
  - Bootloadable
  - Bootloader
  - Clock
  - Die Temperature
  - DMA
  - EEPROM
  - External Memory Interface
    - Asynchronous External Memory Interface
    - Synchronous External Memory Interface
  - Global Signal Reference
  - Interrupt
  - RTC
  - Sleep Timer
  - Sync
  - UDBClkEn
- Thermal Management
  - Fan Controller
  - RTD Calculator
  - Thermistor Calculator
  - Thermocouple Calculator
  - TMP05 Interface
- Annotation
  - Active
    - Buffer
    - Comparator
    - Current Source
    - Inverter
    - OpAmp
    - Voltage Regulator
    - Voltage Source
  - Diodes
    - Bridge Rectifier
    - Diode
    - LED
    - Photo Diode

    - Schottki Diode
    - SCR
    - Triac
    - Varactor Diode
    - Zener Diode
- Electro-Mechanical
    - Connector (1-20 Pins)
    - Motor (3 Phase)
    - Motor (DC)
    - Relay
    - Speaker
    - Switch (SPDT-DPDT)
    - Switch (SPST)
    - Test Point
- Passive
    - Capacitor
    - Crystal
    - Fuse
    - Inductor
    - Potentiometer
    - Resistor
    - Transformer
- Power
    - Battery
    - Ground
    - Power
- Sensors
    - Microphone
    - Photo Resistor
    - Strain Gauge
    - Thermistor
    - Thermocouple
- Transistors
    - N-Channel FET (Depletion Mode)
    - N-Channel FET (Enhancement Mode)
    - N-Channel JFET
    - NPN Transistor
    - P-Channel FET (Depletion Mode)
    - P-Channel JFET
    - P-Channel MOSFET (Enhancement Mode)
    - Photo Transistor
    - PNP Transistor

B

Prototype of the system

A prototype of the system has been set up as shown in figure B.1.

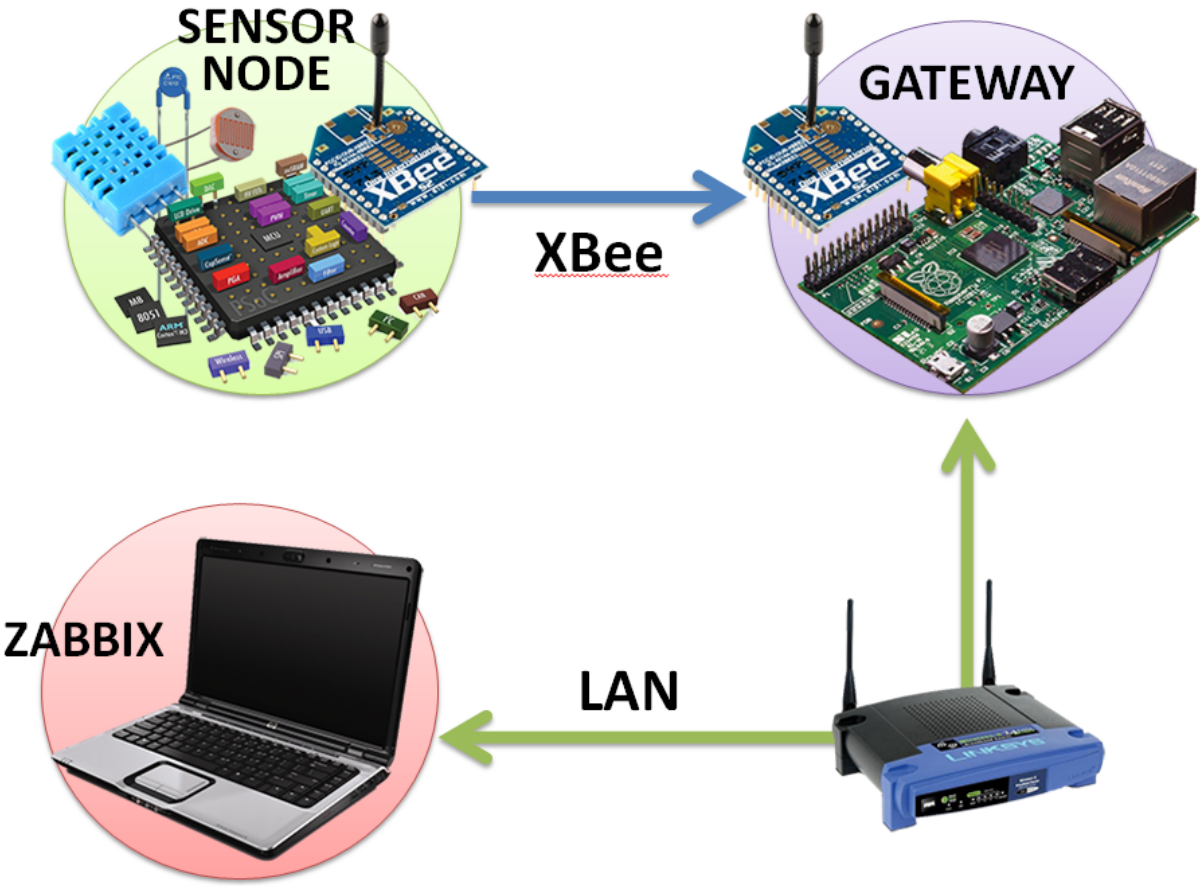


Figure B.1: Prototyped network diagram

B.1 Hardware

The hardware used to build this prototype is:

- Home router
- Laptop
- Raspberry Pi
- PSoC5 First Touch Development Kit
- Two XBee ZB modules
- Two DHT11 sensors
- Thermistor
- LDR

B.2 Setup

B.2.1 Zabbix

Hardware

To run Zabbix just a regular computer or server is required. Zabbix Server runs on any of the following systems:

- AIX
- FreeBSD
- HP-UX
- Linux
- Mac OS X
- Open BSD
- SCO Open Server
- Solaris
- Tru64/OSF

For the prototype, Zabbix has been installed on a laptop featuring:

- AMD Turion 64 x2 CPU
- 1 GB RAM
- 100 GB HDD
- Ubuntu server 12.04 64 bits

The laptop is connected to the LAN via Ethernet port.

Software

Set up

The server is running an Ubuntu Server 12.04 64 bits distribution, with a clean installation of Zabbix Server 2.0.4. Zabbix frontend uses Apache web server, and the data is stored in a MySQL database. An SSH server has been enabled for convenient operation.

Installation of Zabbix Server is thoroughly covered in Zabbix documentation [34].

Hosts and Agents

In Zabbix, a host is any monitored device. Generally, these are computers, routers, switches, load balancers, and other network electronic devices.

Zabbix Agent is a software program that can be installed on any computer or server, and it enables advanced monitoring features of hosts. It is the preferred method to monitor a computer host, though these are not the only devices in a computer network. Zabbix supports multiple management protocols, including SNMP and IPMI for those devices that cannot run an agent.

In this network, Zabbix recognizes the sensor nodes as hosts.

Templates

Zabbix implements the concept of *templates*. A *template* is a file that describes the available parameters for a host type, predefined graphs, and triggers.

As an example, Zabbix has a built-in template for Linux machines which covers parameters such as CPU load, available memory, HDD space, etc..

A template for sensor nodes has been created to acquire sensors' parameters as illustrated on figure B.2.

Templates	Applications	Items	Triggers	Graphs	Screens	Discovery	Linked templates	Linked to
Demo Sensors	Applications (4)	Items (7)	Triggers (8)	Graphs (4)	Screens (1)	Discovery (0)	-	Demo Node
Template App Zabbix Agent	Applications (1)	Items (3)	Triggers (3)	Graphs (0)	Screens (0)	Discovery (0)	-	Template OS Linux
Template App Zabbix Server	Applications (1)	Items (26)	Triggers (24)	Graphs (4)	Screens (1)	Discovery (0)	-	Zabbix server
Template OS Linux	Applications (10)	Items (32)	Triggers (15)	Graphs (4)	Screens (1)	Discovery (2)	Template App Zabbix Agent	Raspberry Pi, Zabbix server

Figure B.2: Templates

Graphs

Zabbix is able to plot graphs in real time upon the data received from the monitored hosts. It can plot automatic graphs from each parameter, or the administrator can specify custom graphs, to set the axes, plot multiple parameters in the same graph, or change the graph type.

Several graphs have been created according to the parameters measured by the sensor node. Figure B.3 illustrates an example.

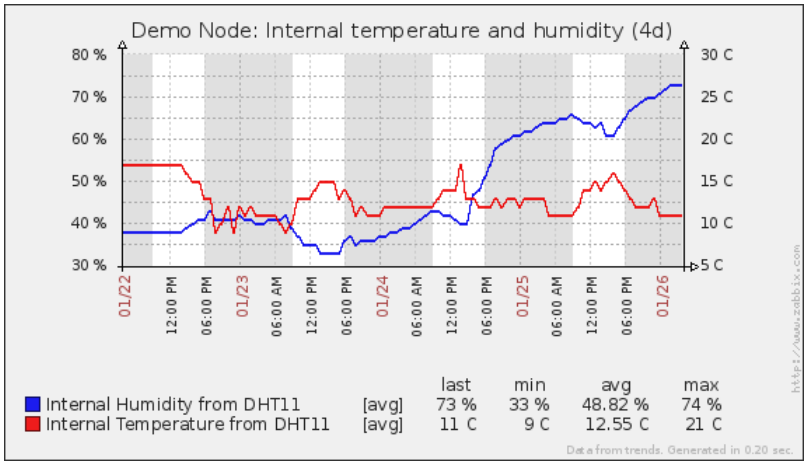


Figure B.3: Templates

Triggers

Triggers are conditions that fire an automatic response from Zabbix Server. This response can be a status message in the frontend, or a custom command being executed (with all the possibilities that implies).

Triggers have been configured for error conditions in the node operation or incorrect values received from sensors, as shown in figure B.4.

Severity	Name	Expression
Warning	DHT11 High humidity	{demo_sensors:sensor.demo.DHT11.humidity.**last**(0)}>90
Warning	DHT11 High temperature	{demo_sensors:sensor.demo.DHT11.temperature.**last**(0)}>49
Warning	DHT11 Low humidity	{demo_sensors:sensor.demo.DHT11.humidity.**last**(0)}<20
Warning	DHT11 Low Temperature	{demo_sensors:sensor.demo.DHT11.temperature.**last**(0)}=0
Warning	Lost packet	{demo_sensors:sensor.demo.thermistor.temperature.**nodata**(75)}=1
Disaster	Node down	{demo_sensors:sensor.demo.thermistor.temperature.**nodata**(200)}=1
High	Node Reset	{demo_sensors:node.event.reset.**last**(0)}=1

Figure B.4: Templates

B.2.2 Raspberry Pi

Raspberry Pi has connected one of the XBee ZB modules to its serial port. It is connected to the router via Ethernet port and powered by the router's USB port.

There are no other peripherals attached to the Raspberry Pi.

The Raspberry Pi is running the following software:

Zabbix Proxy

Zabbix Proxy is a piece of the Zabbix suite, it gathers all the agent requests, processes them and sends them packed to the server.

The use of a proxy isn't mandatory since agents are able to communicate with the server without intermediaries, but its use is recommended:

1. The proxy processes the incoming data and sends it packed in a single TCP connection to the server, reducing bandwidth usage and server load.
2. Behind firewalls, either server or proxy may initiate the connection creating a virtual bidirectional tunnel.
3. In case of link failure, the proxy can buffer the data until the connection with the server is reestablished.
4. It simplifies the maintenance of networks in different geographic places.

Zabbix Agent

Zabbix Agent is a process that gathers the information from its host and sends it to Zabbix Proxy. The way zabbix identifies the values is through string keys, e.g. the value for free memory is identified by the key `vm.memory.size[available]`.

An agent is installed on Raspberry Pi to monitor the device health. It also has a command-line tool called `zabbix_sender` to send information directly from the shell.

XBee-Zabbix translator

A custom script has been coded in Python operating the XBee module to translate the frames received to Zabbix protocol.

As seen in B.2.2, keys are identified by a human-readable string. Implementing this identification type in a microcontroller is poorly efficient, it uses less processor, memory and bandwidth if it is identified using a unique number. In the other hand, having the parameters named in a human readable-fashion is more handy to the user than hard-to-remember numbers. The same problem arises identifying hosts.

To solve this issue the gateway stores lookup tables, hence the user can set friendly keys or host names while keeping efficiency under the hood. Whenever a frame is received by the gateway, it identifies the source host and the parameter key, then it emulates being an agent to send the data to the proxy using the `zabbix_sender` tool through a local-loop socket.

B.2.3 PSoC node

Hardware

A prototype of a node has been set up featuring:

- PSoC5 First Touch Development Kit
- 1x XBee ZB module
- 2x DHT11 sensors
- 1x Thermistor
- 1x LDR

Design

An XBee module is connected to PSoC using the XBee controller developed in section 5.3.5. Figure B.5 illustrates the connection, where components on colored background are inside the PSoC, while components on the white background are off-chip.

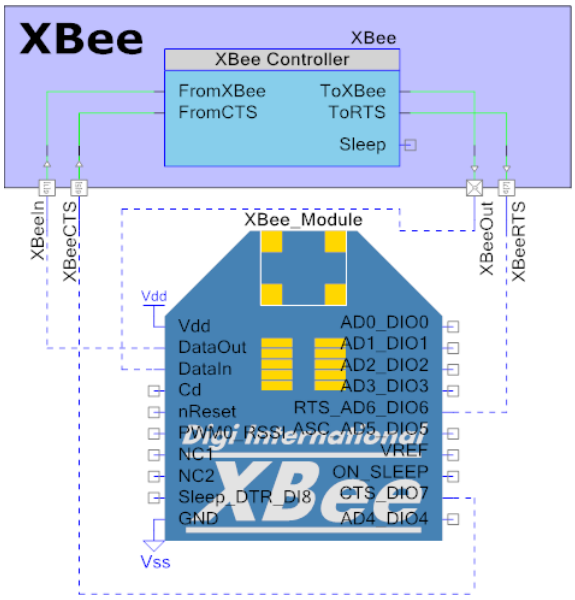


Figure B.5: Connection of XBee to PSoC

Two DHT11 sensors are connected to their respective controllers developed in section 4.3.3, as depicted in figure B.6. One of the sensors is place inside the node enclosure, while the other is an external probe.

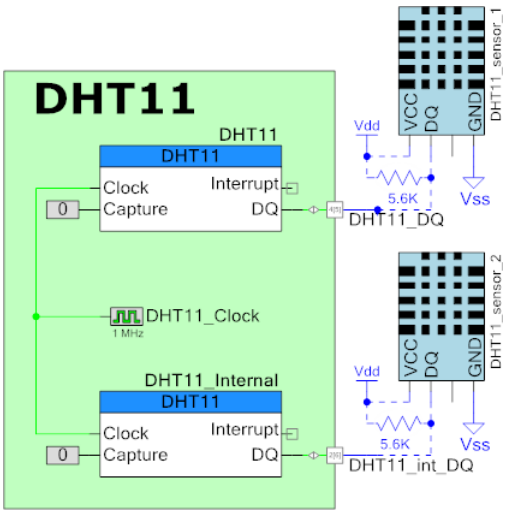


Figure B.6: Connection of DHT11 sensors to PSoC

A $10k\Omega$ thermistor is used to measure internal temperature of the node. The conditioning circuit is an improved version of the one seen in section 4.1.1, using a voltage DAC for precision polarization voltage. Figure B.7 illustrates the signal conditioning and acquisition circuit of the thermistor.

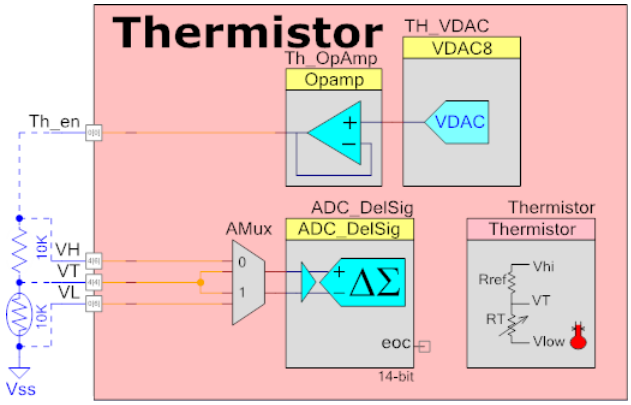


Figure B.7: Signal conditioning for thermistor

An LDR has been incorporated to measure light presence. The conditioning circuit has been reviewed in section 4.1.2 and is shown in figure B.8. Hysteresis has been implemented regulating the VDAC in firmware.

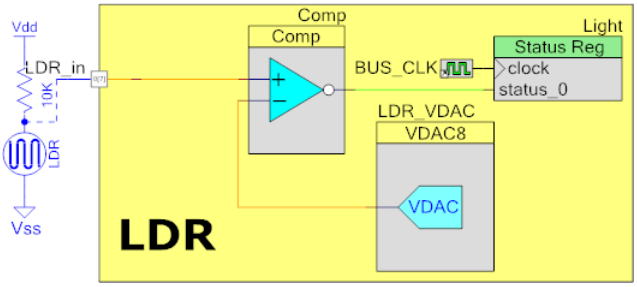


Figure B.8: Signal conditioning for LDR

Firmware

The node provides data according to the following criteria:

- Data from all sensors is transmitted once every minute.
- The light status is checked every 5 seconds to detect glimpses from artificial sources and the server is notified if there is any change.
- Reset is notified to the server.

Figure B.9 illustrates the program flow.

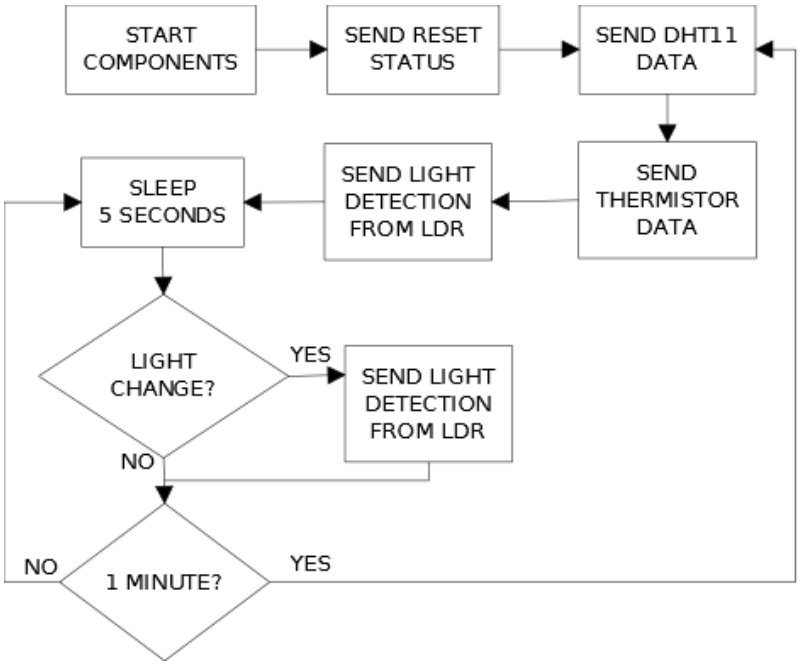


Figure B.9: Program flow

C

Introducción

C.1 Qué es una red de sensores

Una red de sensores es un conjunto de dispositivos distribuidos en un espacio con el fin de monitorizar condiciones físicas o ambientales, como temperatura, sonido, vibración, presión, movimiento o polución, que de forma cooperativa envían sus datos a través de la red hasta una localicación principal.

Las redes de sensores son con frecuencia inalámbricas, resultando en una instalación sencilla, rápida y de bajo coste. La vida de las baterías puede extenderse de meses a algunos años, y la distancia de comunicación puede comprender hasta varios kilómetros, o todo el planeta aprovechando las existentes infraestructuras de comunicación.

C.2 Aplicaciones

Las redes de sensores tienen un sin fin de aplicaciones, aquí se presentan algunas de ellas para ayudar a dar una idea de su utilidad.

Vigilancia
: Los nodos son desplegados en un área a vigilar, haciendo saltar alarmas cuando se detecta una intrusión o una anomalía en el perímetro. Pueden ser usadas para vigilancia de infraestructuras críticas como líneas de combustible, así como no críticas gracias a su bajo coste.

Medio ambiente
: Los nodos distribuidos en localizaciones diferentes miden la calidad del aire, suelo y agua. Estas redes de sensores se emplean, por ejemplo, en plantas de tratamiento de aguas residuales donde la electricidad y redes de comunicaciones no están siempre disponibles. Pueden ser usadas para detección temprana de incendios forestales, inundaciones o deslizamientos de tierra. Pueden incluso desplegarse en el mar para monitorizar el tráfico marítimo y proteger la fauna marina [28].

Automatización industrial
Las redes de sensores se emplean en automatización de procesos industriales y monitorizacin de salud de la maquinaria. En este último punto son de especial interés ya que los nodos pueden ser emplazados en lugares peligrosos para las personas, así como, siendo inalámbricos, en partes móviles.

Agricultura
Las redes inalmbricas se despliegan en campos de cosecha e invernaderos, mejorando la gestión de recursos. Pueden automatizar el riego optimizando el uso de agua, prevenir heladas y detectar enfermedades de forma temprana.

Estructuras
Los nodos colocados en puntos de estrés mecánico de edificios, puentes o infraestructura de transportes, monitorizan vibraciones, estiramientos y roturas, sin necesidad de parar la actividad y de forma más precisa que la inspección visual.

C.3 Estado del arte

Hay un alto número de soluciones en el mercado de las redes de sensores. En esta sección se revisan algunas de las soluciones líderes que permiten al usuario personalizar de algún modo el sistema desplegado.

C.3.1 Soluciones de la comunidad y sin ánimo de lucro

Las soluciones de la comunidad están lideradas por Arduino. La mayoría de sistemas se basan en nodos compatibles con la plataforma y comparten herramientas, librerías y hardware.

Arduino

Arduino es una plataforma libre de prototipado de electrónica. Su microcontrolador se programa empleando un lenguaje de programación propio basado en Wiring y el IDE Arduino basado en Processing.

Las placas pueden ser construidas a mano o comprarse ensambladas. El software de desarrollo puede ser descargado de manera gratuita. Los diseos (archivos CAD) están disponibles bajo una licencia libre.

A pesar de ser una plataforma de prototipado de propósito general y no tener capacidades radio integradas, Arduino se ha convertido en la base de muchas redes de sensores desarrolladas por aficionados y organizaciones sin ánimo de lucro. La mayor parte de estos proyectos emplean radios XBee, u otros sistemas cuyo control se realice a través de la UART.

Arduino se ofrece en múltiples versiones con formas y MCUs diferentes. El modelo original está disponible con un precio de EUR 20.

panStamp

panStamps son unas pequeas placas con capacidad de comunicación inalámbrica integrada, diseadas para aplicaciones de bajo consumo, con la filosofía según su fabricante de *fácil de programar y de trabajar con ellas.*

Hay un catálogo de placas que contienen sensores y actuadores donde un panStamp puede ser conectado. La comunicación se realiza mediante un protocolo llamado SWAP en las bandas de 868 y 915 MHz.

Las panStamps son clones de Arduino, por lo que emplean el mismo IDE y librerías. Se provee además una solución completa en redes de sensores mediante su servidor Lagarto, un software hecho a medida para operar la red.

C.3.2 Soluciones comerciales

Las soluciones comerciales están con frecuencia basadas en el MSP430 de Texas Instruments y la serie ATmega de Atmel.

WaspMote by Libelium

WaspMote es un nodo sensor basado en Arduino y producido por Libelium, una compañía española de Zaragoza.

El núcleo del WaspMote es un ATmega1281, tiene un conector para un *sensor shield* que consiste en una placa con los sensores a emplear, y un conector compatible con XBee para comunicaciones. Además tiene algunos periféricos integrados en la placa principal como un RTC y un acelerómetro.

Su firmware se desarrolla empleando un IDE propietario basado en el Arduino IDE, y el control de la red se realiza también mediante un software propietario, ambos desarollados por el fabricante.

Tiene una API abierta, pero el hardware y el software de monitorización son de naturaleza cerrada.

WaspMote es ofrecido con diferentes configuraciones radio empezando en EUR 130 [30].

Tinynode

Tinynode es una compañíia con base en Suiza, que desarrolla redes de sensores para la detección de vehículos.

Adem´a la empresa ofrece su plataforma, Tinynode 584, dirigida por un MSP430 ejecutando TinyOS y una radio Semtech para comunicaciones. La intención de esta plataforma es proveer de fácil comunicación a sensores, actuadores y controladores.

El sistema emplea el software TinyOS Oscilloscope para la muestra de los datos.

La placa es propietaria y con fuentes no liberadas, a un precio de EUR 109.

nCore by Nebusens

Esta empresa con base en España ofrece varios dispositivos para desplegar redes de sensores, desde pequeñas etiquetas RFID hasta nodos personalizables.

La red está lista para funcionar de fábrica, los nodos están preprogramados con un firmware específico, con la flexibilidad suficiente como para leer sensores externos sin necesidad de repro-

gramación. Todo el desarollo debe realizarse en el lado del servidor, donde un ordenador envía los parámetros de configuración a los nodos y solicita las capturas o actuaciones.

El firmware, API y hardware son propietarios y de fuentes no libres. El precio de venta no es público.

C.3.3 Tendencias

Después de analizar algunas de las soluciones disponibles en el mercado, puede llegarse a las siguientes conclusiones:

- Arduino es la plataforma l'ider en las soluciones libres.
- Para sistemas comerciales, los MCUs más empleados son el MSP430 y la serie ATmega.
- La flexibilidad de los nodos está limitada a las capacidades de estas MCUs, con limitados recursos y potencia de cálculo.
- Software desarrollado a medida es la solución habitual para interactuar con los sistemas.
- La comunicación se realiza empleando protocolos estándares y propietarios indistintamente.

C.4 Objetivos y enfoque

Este proyecto presenta una solución flexible y potente, y aún así asequible, en redes de sensores, aprovechando tecnologías ampliamente accesibles libres y de bajo coste.

Innovativa
Sin seguir las tendencias clásicas de las redes de sensores.

Bajo coste
Accesible por todos, desde corporaciones a aficionados.

Multi-propósito
Sin aplicación o entorno de trabajo intencionado. Los nodos pueden ser desplegados de forma fija o móvil, enterrados o en el mar... Con la posibilidad de integrar tantos tipos de sensores dieferentes como sea posible.

Desatendida
Una solución completa comprende nodos, puertas de enlace, almacenamiento de datos y una interfaz amigable para su visualización.

Fácil de desplegar
No deben requerirse amplios conocimientos del funcionamiento del sistema para poner a funcionar una red.

Extensible
Para los casos especiales que necesiten algo más allá de las posibilidades listas para funcionar.

C.5 Estructura de la memoria

El presente documento se organiza de la siguiente forma:

1. Introducción La introducción de este documento, lo que el lector acaba de leer.
2. Enfoque de una red de sensores flexible y fácilmente personalizable. Se presenta el sistema bajo estudio y con una introducción a las tecnologías empleadas.
3. Introducción a PSoC Una introducción extensa sobre PSoC, el núcleo del sistema, con detallada información técnica.
4. PSoC y sensores Varios sensores se clasifican por su principio de funcionamiento, solucionando el problema de acondicionamiento de señal que presentan usando las capacidades disponibles en PSoC.
5. Capacidades de comunicación de PSoC Introducción a algunas de las capacidades de comunicacion de PSoC. Explicación mediante ejemplo de cómo éstas pueden ser ampliadas.
6. Gestión de energía en PSoC Una vista general sobre la distribución de energía en PSoC y los modos de operación en bajo consumo.
7. Consideraciones para el diseño de un nodo Se dan algunas indicaciones para la construcción de un nodo. Detallada información de la organización interna de PSoC.
8. Conclusiones y trabajo futuro Conclusiones y consideraciones son expuestas en este capítulo.

Después del estudio, un prototipo del sistema propuesto ha sido construido satisfactoriamente. Detalles sobre este prototipo se encuentran en el Anexo B.

D

Conclusiones y trabajo futuro

D.1 Técnica

Hay varias características de PSoC que lo hacen ideal para el caso de este nodo altamente flexible:

Su rutabilidad interna es perfecta para un nodo donde los sensores e interfaces de comunicación no están prefefinidos. Cualquier pin puede rutarse a cualquier bloque, ya sea analógico o digital. Con el poderoso DSI, preciados recursos como el ADC Delta-Sigma pueden ser compartidos entre múltiples pines, sin importar su emplazamiento en el chip.

Los bloques analógicos configurables tienen una sensibilidad excelente para seales pequeñas. Además, reducen el tiempo de I+D y la BOM al mejorar el proceso de prototipado y minimizando el número de componentes externos requeridos para el acondicionamiento de señal.

Respecto a los componentes basados en UDB, una de sus mayores ventajas es la no restricción a un número fijo de periféricos de un mismo tipo. Esto implica que cualquier módulo externo (ya sean sensores digitales, interfaces de comunicación o cualquier otro dispositivo) pueda conectarse a PSoC sin importar qué protocolo de comunicación emplee. Como ejemplo, en un caso de uso particular cuatro UARTs pueden ser necesitadas; en otro puede ser útil tener dos SPIs, uno de alta tasa de transferencia para la interfaz de comunicación y otro de menor ancho de banda para algunos sensores digitales.

D.1.1 Acondicionamiento de señal

Como se ha expresado anteriormente, PSoC tiene unas capacidades muy buenas para el acondicionamiento de señal. En este documento algunos circuitos simples han sido estudiados, pero las posibilidades van mucho más allá. Cuando la situación lo requiera, lecturas precisas puden ser tomadas realizando circuitos de excitación y acondicionamiento muy complejos sin añadir componentes externos a la BOM.

En el ejemplo del termistor, la fuente `Vddio` se ha usado para excitar el sensor, lo que es suficiente ya que el modo de cálculo elimina cualquier desviación, y el termistor elegido tiene una precisión relativamente pobre. Pero en el caso de un sensor de alta precision, un DAC de voltaje puede ser empleado junto a un seguidor de tensión, empleando exactamente el mismo número de partes externas.

El bloque SC/ST ha demostrado ser de gran utilidad, integrando recursos de acondicionamiento activo. El TIA es tan preciso que puede dar lugar a voltages notorios a partir de corrientes en las decenas de nanoamperios. El problema del desarrollador queda en tener un circuito con un nivel tan bajo de error. Cypress provee la documentación y herramientas necesarias para calcular desviaciones debidas al enrutamiento interno y la precisión de los componentes.

Todo tiene un coste. Y esta precisión en un dispositivo de bajo consumo tiene la limitación en la señal de entrada por debajo de los $5V$ o unos pocos miliamperios. Para señales por encima de estos límites, acondicionamiento externo es requerido. A pesar de esto, para el caso de un nodo de bajo consumo se asume que generalmente los sensores mismos tendrán un consumo tan bajo como sea posible.

D.1.2 Sensores digitales

Sobre los sensores digitales, los bloques UDB han probado ofrecer una gran ventaja técnica sobre soluciones realizadas en firmware típicas de los MCUs clásicos:

Consumo: Los componentes basados en UDBs tienen a consumir menos energía que las soluciones software.

Planificación de tareas: Varios componentes pueden trabajar al mismo tiempo en paralelo. Durante su operación la CPU está libre para realizar otras tareas.

Portabilidad: Los componenentes proveen una encapsulación completa de sus funciones, siendo su portabilidad tan simple como arrastrar-y-soltar en un nuevo proyecto, pudiendo instanciar varios componentes del mismo tipo sin cambios en los mismos.

A pesar de ello, se debe tener en cuenta que el desarrollo de componentes es una tarea que consume mucho tiempo, posiblemente varias veces más que la solución en firmware, y requiere un conjunto de conocimientos en hardware y software que juntos pueden hacer su desarrollo poco práctico para algunas aplicaciones. Queda a elección de los desarrolladores evaluar los beneficios e inconvenientes de ambas implementaciones caso por caso.

D.1.3 Comunicaciones

Los nodos sensores conectados a un bus CAN son comunes en los entornos industriales, para lo que PSoC tiene un bloque dedicado. El protocolo RS-485 es otro de los más comunes en el mismo entorno, y PSoC suple esta necesidad con una UART basada en UDBs disponible en el Cypress Component Catalog.

Para comunicaciones inalámbricas, integración con un módulo ZigBee ha sido desarrollada con éxito. De nuevo probando su aptitud para entornos industriales.

Y, sobre comunicaciones chip-a-chip, los componentes para I^2C, SPI y UART han sido estudiados. Siendo estos protocolos comunes en comunicaciones dentro de una misma placa, o entre placas cercanas.

Para una lista completa de los componentes listos para funcionar, consulte el Anexo A. Y, en caso de que el componente que busca no se encuentre en la lista, recuerde que puede ser capaz de implementarlo usando los recursos configurables de PSoC.

D.1.4 Gestión de energía

A pesar de la alta flexibilidad y potencia de cálculo de PSoCm su consumo de energía se mantiene relativamente bajo. Consumiendo hasta $300\mu A$ en modo hibernación, PSoC puede despertar desde cualquier GPIO, manteniendo el estado de la CPU y SRAM.

Descargar la CPU de algunas tareas es posible empleando componentes basados en los bloques UDB. Ésta es una de las posibilidades para bajar el consumo más remarcables cuando se emplea con el estado *Alternate Active* y la CPU desconectada.

La tecnología de *boost* permite al dispositivo funcionar con funciones analógicas completas desde los $0.5V$. Ideal para operar con baterías de botón.

D.2 Experiencia de Usuario

D.2.1 PSoC Creator

PSoC Creator es un entorno de desarrollo muy amigable y potente, que se provee de forma gratuita sin limitaciones en el tamaño del código. Licencias de pago hubiesen descartado a PSoC de esta solución.

Genera toda la API necesaria para operar con el dispositivo, de forma que el usuario no tiene que operar directamente con registros de bajo nivel, salvo para operaciones especiales. La curva de aprendizaje aumenta más rápido que con otros MCUs. PSoC es mucho más sencillo y amigable en lo que respecta al desarrollo del firmware, a la vez que favorece su portabilidad y el mantenimiento.

La creación de componentes propios ofrece el grado ideal de encapsulación y portabilidad que un proyecto open-source requiere.

D.2.2 Software externo

Software externo, no incluido, se requiere para algunas tareas.

Simulación de componentes descritos en Verilog

PSoC Creator no integra una herramienta de simulación para los componentes descritos usando verilog, por lo que una herramienta de un tercero es necesaria. Estos simuladores suelen ser muy caros si el usuario no está afiliado a ninguna institución educativa, y las soluciones libres o gratuitas suelen escasear de capacidades.

Desarrollo de *Component Customizers*

Los diálogos de personalización de los componentes deben ser descirtos en C♯. Es posible realizar esta tarea desde PSoC Creator, pero Visual Studio facilita la tarea de diseñar la GUI de una

manera gráfica, además de permitir todas las características de un IDE completo (generación de código, auto-completar, detección de errores...). La versión gratuita de Visual C♯ es apta para esta tarea, pero Visual Studio Standard es necesario para depurar el programa.

Esta dependencia de software externo hace que PSoC sea algo menos accesible o amigable para el usuario objetivo de este proyecto. Por suerte, su uso no es necesario para realizar desarrollos completos usando la plataforma, y estas tareas pueden ser parcialmente realizadas con alternativas gratuitas.

D.2.3 Documentación

Hay mucha documentación, trainings y ejemplos en el sitio web de Cypress [25]. Todos los componentes tienen datasheets muy detallados, y existen *application notes* contemplando varios de ellos en diferentes condiciones de uso.

Además, hay algunos blogs editados por empleados de Cypress con ejemplos de aplicaciones, tutoriales y más información alrededor de PSoC. Así como un foro oficial donde los mismos empleados también participan. Yendo más lejos, hay varios sitios web independientes sobre PSoC donde pueden encontrarse manuales y ayuda.

Para aplicaciones avanzadas el usuario puede sentirse algo solo. PSoC3 y PSoC5LP se basan en una arquitectura relativamente reciente y nueva documentación es constantemente publicada. Puede observarse que los temas más avanzados van siendo cubiertos con las últimas publicaciones.

D.3 Trabajo futuro

Para construir el sistema aún hay que trabajar profundamente en el servidor, la puerta de enlace y los mismos nodos.

En el lado del servidor, un estudio exhaustivo de las posibilidades de Zabbix y la posibilidad de adecuarlo a las particularidades de una red de sensores debe ser realizado antes de tener un sistema completamente funcional. Para el prototipado, una instalación mínima ha sido realizada y Zabbix funciona para este fin sin más manipulación, pero quedan algunas características interesantes por ser estudiadas o desarrolladas. Como la habilidad de enviar comandos remotos a los nodos, posible a traves de los scripts propios que Zabbix es capaz de lanzar.

En la puerta de enlace, un canal de comunicación unidireccional ha sido establecido usando en script en Python que controla un XBee y traduce sus tramas al protocolo de Zabbix. Más trabajo podría ser realizado para conseguir un canal bidireccional de comunicación, o incluso más allá para conseguir abstraer las distintas partes implicadas y no condicionar al uso de XBee.

Respecto al nodo, queda mucho trabajo por hacer:

1. Se debe diseñar, construir y verificar un placa. Algunas recomendaciones han sido expuestas en el capítulo 7.

2. Estudiar sensores, interfaces de comunicación y fuentes de energía del estado del arte. Con análisis de sus aplicaces reales y las necesidades del mundo real.

3. Construcción de *daughter cards* e implementar los componentes propios y firmware necesarios para operarlas.

Y, para conseguir un sistema usable por el público, una biblioteca de documentación contemplando:

1. Datasheets de los componentes desarrollados.

2. Datasheets del nodo y las *daughter cards* desarrolladas.

3. Applicacion notes.

4. Ejemplos de aplicación.

5. Análisis de rendimiento.

6. Manuales técnicos.

7. Imágenes software del sistema.

E

Pliego de condiciones

Pliego de condiciones

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de una *red de sensores basada en PSoC*. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales.

1. La modalidad de contratación sera el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si éste se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el numero de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algun trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Tecnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 dias naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares.

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.

F

Presupuesto

1) **Ejecución Material**

- Ordenador (Software incluido) 500 €
- Servidor (Software incluido) 400 €
- Osciloscopio 330 €
- Raspberry Pi 45 €
- Kits de desarrollo PSoC 500 €
- Componentes electrónicos 100 €
- Impresora láser 120 €
- Material de oficina 5 €
- Total de ejecución material 2.000 €

2) **Gastos generales**

- sobre Ejecucion Material 260 €

3) **Beneficio Industrial**

- sobre Ejecucion Material 120 €

4) **Honorarios Proyecto**

- 800 horas a 15 €/ hora 12.000 €

5) **Material fungible**

- Gastos de impresión 5 €
- Encuadernacion 10 €

6) **Subtotal del presupuesto**

- Subtotal Presupuesto 14.395 €

7) I.V.A. aplicable

- 21% Subtotal Presupuesto 3.022,95 €

8) Total presupuesto

- Total Presupuesto 17.417,95 €

Madrid, Febrero, 2013

El Ingeniero Jefe de Proyecto

Fdo.: Pablo Bacho Manzorro

Ingeniero Superior de Telecomunicación