2006

A Low-Power Interface Design for Intelligent Sensor Nodes Utilized in Wireless Sensor Networks

Shruti Lakdawala

A Low-Power Interface Design for Intelligent Sensor Nodes Utilized in Wireless Sensor Networks

By

Shruti Lakdawala

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Supervised by

Dr. Fei Hu
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
February 2006

Approved By:

Fei Hu

Dr. Fei Hu, Assistant Professor
Department of Computer Engineering, RIT

Daniel Phillips

Dr. Daniel Phillips, Assistant Professor
Department of Electrical Engineering, RIT

Pratapa Reddy

Dr. Pratapa Reddy, Professor
Department of Computer Engineering, RIT

Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

A Low-Power Interface Design for Intelligent Sensor Nodes Utilized in Wireless Sensor Networks

I, Shruti Lakdawala, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

S.L

Shruti Lakdawala

03/02/2006

Date

Acknowledgements

My sincere gratitude to Dr. Hu and Dr. Phillips, who made it possible for me to do this research by providing me with the necessary resources and their valuable guidance from time to time. Also, Dr. Reddy for his valuable comments towards improving my research. I am extremely grateful to all of them for being understanding and encouraging throughout the course of my thesis.

I wish to thank the members of Tinyos-Help forum that helped me resolve my queries in TinyOS as well as NesC. Also, Cypress Technical Support as well as the PSoC Developers forum, for making all the PSoC technical resource available and for their quick response to my PSoC queries.

I am very grateful to my family for being patient, understanding and supportive throughout the course of my Masters curriculum. I would also like to specially thank Jyoti and Kush Lakdawala for their generosity and hospitality. Finally, my friends for their help and motivation when I most needed it.

Abstract

A Low-Power Interface Design for Intelligent Sensor Nodes utilized in Wireless Sensor Networks

This thesis describes a means for performing complex event detection at a single sensor node of a wireless sensor network (WSN) by interfacing a low-power mixed signal Programmable System on Chip (PSoC) to a MICA2 wireless sensor node. The proposed system helps to reduce the overall power consumption of the node, by lending it the advance computational capability to process a significant amount of data at the node rather than transmitting it. This allows the node to "intelligently" monitor a signal for impending events instead of transmitting the "raw" signal to the base constantly. Previous work by others has indicated that lowering the transmission data rate lowers the high cost of transmission power [41], [42] in a node thereby lengthening the node life and, ultimately, increasing the reliability of the network [43].

This work implements a threshold technique which controls the data transmission rate depending on the value of the monitored signal and a cardiac monitoring system that performs complex computations at the node for the detection of either a skipped heart beat or a reduced Heart Rate Variability (HRV) whereupon the relevant unprocessed signal is transmitted to the base station for direct observation. A performance analysis of the system demonstrates that there is a reduction in the power consumption of the overall sensor node and a significant reduction in data transmission rate which also results in a reduction of the overall network traffic and congestion.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Introduction:

Wireless Sensor Networks (WSN) are formed by a number of sensor nodes that are deployed over a region for monitoring data wirelessly. Each node is normally comprised of a microprocessor, transceiver and a sensor. Ongoing developments in wireless communication, integrated circuit design and Micro-electro-mechanical systems (MEMS) have resulted in small, low-power, low-cost sensors and electronic devices, which makes establishment of a WSN a reality. A single wireless node relies on its limited local resources and has minimal capabilities, however when a number of this type of sensor node are distributed over a region they are able to extend the overall capabilities by using advanced networking protocols to transfer data autonomously from one node to another and arbitrary remote destinations. As a result, the wireless network comprised of these nodes can possess substantial processing capability [49].

WSNs are becoming increasingly versatile with the miniaturization of the nodes and can be used for a wide range of applications [1]. They can be used for environmental applications: the air, water, soil conditions which affect the crops and livestock could be monitored continuously; habitat monitoring: the seasonal migration of animals and birds could be tracked by tagging them with a sensor node; disaster management: forest fires can be detected, alerting the authorities before they spread uncontrollably; indoor security and maintenance: In homes, offices and hospitals they could be used to monitor and help maintain temperature and humidity and trigger alarms in case of any security breach;

military applications: surveillance, tracking soldiers, damage assessment; structural monitoring; physiological monitoring: whereby the patients could be tagged with wireless sensor node that would makes patient mobility possible for patients that need to be monitored over an extended period of time.

In particular, WSNs have enormous potential benefits in the area of physiological monitoring or biomedical applications [38]. It affords patient mobility that can facilitate physiological study of the patients or subjects during their normal day to day activities. Remote physiological monitoring for vital signs such as blood pressure, electrocardiogram, pulse and body temperature enables patients to have a better quality of provided care and help maintain their health. It is useful for diagnostic and research purposes whereby, a collection of data from subjects is required over a long period of time. A good example is the study of Heart Rate Variability (HRV) [3] [20] of subjects over a period of 24 hours for the study of the risk of heart attacks.

1.2 Problem and Motivation:

The general architecture of a wireless sensor node generally includes sensor, control and communication modules along with a power subsystem. The sensor module is responsible for data acquisition and any necessary signal conditioning. The control module provides the network protocols and controls the sensor and communication modules. The communication module handles the wireless transmission and reception of data/sensor readings. The power subsystem is responsible for providing power to all the three modules.

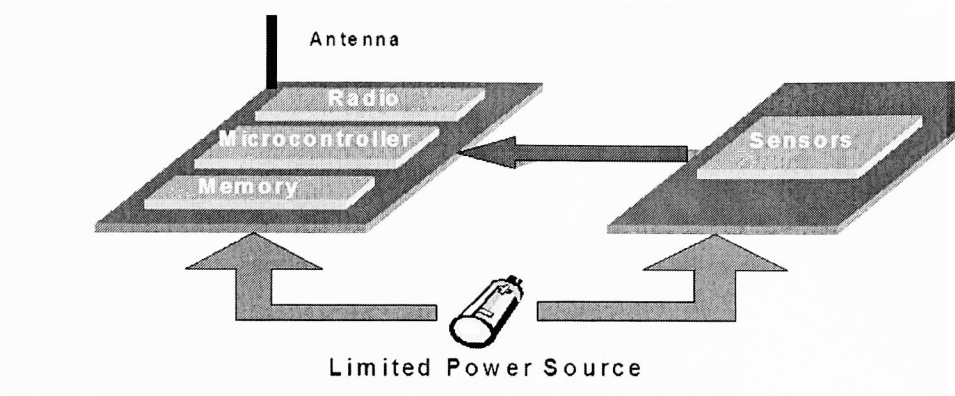


Figure 1.1: General Block Diagram of a Sensor Node

The finite power budget of an individual node is a primary design constraint for applications requiring detailed monitoring over a long period of time using wireless sensor nodes [36]. A single node depends on a relatively limited power source, which may be inconvenient, expensive or impractical to replenish. Hence, increasing the power efficiency of individual sensor node is important for a WSN node design.

The key to reducing power consumption in individual nodes is to reduce the amount of data transmitted wirelessly because radio transmission typically consumes a major percentage of total power consumption. Based on this premise, Table 1.1 shows the power consumption in a MICA2 sensor node utilized in this work. It can be observed that the transmission power is higher than the power consumption of the rest of the node.

POWER SPECIFICATION	
Processor	**Power**
Full Operation	28.8 mW
Sleep Mode	28.8 μW
Radio	
Receive Mode	28.8 mW
Transmit Mode	43.2 mW
Sleep Mode	7.2 μW
Sensor Board	
Full Operation	18.0 mW
Sleep Mode	18.0 μW

Table 1.1: Approximate Power Consumption of MOCA2 [47].

It is well known that extensive research is being carried out in the reduction of transmission overhead including power associated with data communication [11][13]. As mentioned by Pottie and Kaiser in their discussion of wireless networks [2], the computation cost can be several orders lower than the communication cost at an individual node; therefore local processing of the data points and transmitting fewer data bits over the network, especially over a long distance, would save a substantial amount of power. Constraints on this line of thought consist of the computational capabilities within an individual node, limited memory resources and any external custom amplification or filtering circuitry necessary for a particular sensing method or technology.

1.3 Proposed Design:

This thesis proposes interfacing a highly versatile PSoC Mixed-Signal array with a wireless sensor node MICA2, to extend the computational capabilities of the node, while keeping its power consumption in check. A PSoC Mixed-Signal Array is a low-power programmable-systems-on-chip that allows programming of analog and digital (mixed-

signal) components that are typically used in embedded systems. It also has a built in microcontroller which integrates and controls all of the programmed components. It has a unique capability of routing analog as well as digital signals in a manner such that they can be used as trigger to different analog/digital blocks as per the designer's requirements. This flexible feature of having configurable mixed-signal blocks, whose signals can be routed as desired, makes the PSoC an extremely versatile device to incorporate in the design of a system for the detection of complex events. Efficiencies, compared to the design of a similar circuit using separate analog and/or digital components, are potentially realizable in terms of simplicity, energy and overall circuit real-estate. Because of extended capabilities, that can be achieved by the interfacing of this mixed signal array with programming functionality, complex filtering or triggering functions as well as application specific data compression or suppression algorithms can be implemented at an individual node. This can facilitate increased reduction of the volume of data to be transmitted over the network resulting in reduced transmission time and significant reduction in network traffic.

The overall design of the wireless sensor node utilized in this work is shown in Figure 1.2 and consists of a standard MICA2 Wireless node (Crossbow Technology Inc, San Jose, CA) that communicates with a PSoC (Part number: CY8C27743) (Cypress Semiconductors, San Jose, CA) based sensor node over a standard, UART-based RS-232 data link.

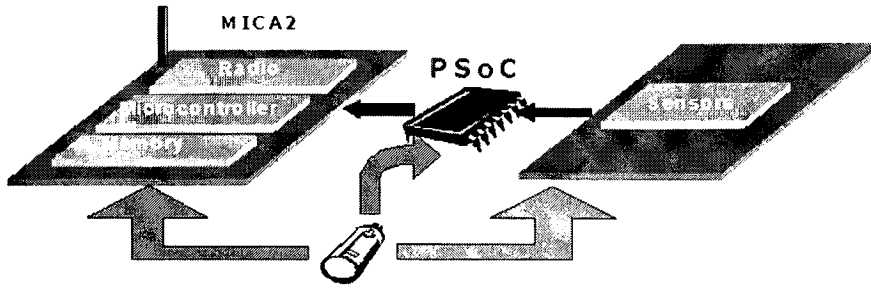


Figure 1.2: Block Diagram of the Proposed System, PSoc Mixed Signal Array interfaced to MICA2 platform

In this thesis, the aforementioned concept of reduction of transmission data due to extended computation capabilities of the PSoC is demonstrated using a technique where the raw sensor readings are compared to a threshold value. If the signal property of interest is above a specified threshold, it is transmitted to the base station immediately; however if the computed signal property is below the threshold, it is transmitted after ten counts, during which, the average of ten consecutive sensor readings, lying below the threshold, is computed. The transmission of the computed values of the sensor data which are below the specified threshold are thereby suppressed, reducing transmission data, which thus saves power consumption in communications.

In addition, the application of this PSoC enhanced sensor node in physiological monitoring is developed and analyzed, which further demonstrates that the low-power PSoC's computational capabilities is capable of facilitating reduced transmission data and network traffic substantially. The Heart Rate Variability (HRV) derived from a simulated electrocardiogram (ECG) signal is calculated using statistical methods. HRV refers to the alterations in the beat-to-beat heart rate [3]. Reduced HRV is used as a marker of unhealthy cardiac function. The PSoC continuously processes the cardiac signal and computes the HRV as well as looks for skipped heart beats. In our implementation, the

sampled cardiac signal transmitted to the base station only on the occurrence of a skipped heart beat or reduced HRV. This could also be altered to send the calculated HRV or skipped heart beat notification. This reduces continuous transmission of the sampled cardiac signal over the network.

2. Background

2.1 Power-efficient Sensor Networks:

In WSN, the lifetime of individual sensor nodes is strongly dependent on its battery lifetime. It is difficult to replenish the batteries of individual nodes in most of the applications. The available energy resource of the nodes therefore limits the overall operation of the entire network, because failure of individual nodes can cause substantial topological changes to the network. Hence, power management and power conservation is of utmost importance within the network as well as in individual nodes.

To design a power-aware system, the energy consumption at all levels of system hierarchy must be kept at a minimum. The integrated circuit design of the node components, the data acquisition system, operating system of the node processor, data processing algorithm at application level, the network protocol as well as the communication medium all must strive for overall minimal power consumption. Active research is being conducted in all these areas for reduction in energy consumption so as to increase the node lifetime and thereby the quality and reliability of WSN.

A low-power microprocessor, transceiver, data acquisition system and external memory are the basic building blocks of the node architecture. Innovative transistor technologies and special-purpose microprocessor architectures are being researched and designed for small size, low-voltage, low-power and better computational performance. Similar research and development effort is being made in integrated RF communication system

with main focus in low power RF integrated circuits resulting in improved, low-cost RF transceiver IC's.

Different hardware platforms, using alternative components for the basic blocks have been developed and analyzed for power consumption, flexibility, robustness as well as communication and computational capabilities.[4][5][6][7] Table 2.1 shows few wireless nodes called "Motes" that are made available by Crossbow Technology Inc.

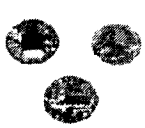
Photo	Crossbow Part ID	Commonly Used Name	Frequency Range	Processor	Radio Transceiver	Nonvolatile Memory
	MPR400 (discontinued)	MICA (sometimes referred to as MICA1)	902 to 928 MHz	Atmel ATmega128L	RFM TR1000	Atmel AT45DB041B (512 kB)
	MPR410 (discontinued)		433.1 to 434.8 MHz			
	MPR400	MICA2	868 to 870, 902 to 928 MHz	Atmel ATMega128L	Chipcon CC1000	Atmel AT45DB041B (512 kB)
	MPR410		433.1 to 434.8 MHz			
	MPR420		313.9 to 315.1 MHz			
	MPR500	MICA2DOT	868 to 870, 902 to 928 MHz	Atmel ATMega128L	Chipcon CC1000	Atmel AT45DB041B (512 kB)
	MPR510		433.1 to 434.8 MHz			
	MPR520		313.9 to 315.1 MHz			
	MPR2400	MICAz	2400 to 2483.5 MHz	Atmel ATMega128L	Chipcon CC2420 (802.15.4)	Atmel AT45DB041B (512 kB)

Table 2.1: Hardware platforms ("Motes") from Crossbow Technology Inc. [40]

The operating system is a software framework for the hardware and the network. Due to the memory constraint on individual nodes, operating systems designed explicitly for wireless sensor network application is needed. Performance of the operating system is a key to further reduction in power consumption. Different operating systems such as TinyOS [8], SOS [9], Contiki [10] etc. are being researched and developed.

Design for energy efficient network protocols are also being researched: MAC protocols [11] [12], investigate reducing energy consumption in periodically putting the nodes in sleep state instead of the node being idle; Ad-hoc Routing protocols [13], investigate methods for the transmitted data to take the shortest possible path to travel from the source node to its destination; Clustering Algorithm [14], analyze and propose energy efficient algorithms for clustering of nodes in groups; Data Aggregation Methods[15], that analyze and investigate novel approach for path sharing to save energy consumption; Data processing algorithms similar to those in [37].

New protocols for RF communication are being researched. Wireless protocols such as Zigbee, Bluetooth and WiFi are being analyzed for optimum data rate, range, reliability, cost effectiveness and low-power. Improvements in battery technology are also being researched to lengthen the battery life.

2.2 PSoC Mixed Signal Array:

The PSoC (programmable-system-on-chip) Mixed Signal Array has been developed by Cypress. It is an effort towards integrating the commonly used embedded system components in one chip replacing the need to have separate IC's consequently reducing board space, power consumption and cost [17]. It is extremely versatile and Cypress was even awarded the "EDN Innovation Award" in 2001 for the 8/16 bit microprocessors category [16] for the PSoC(TM) product. EDN magazine is a leading electronics and the awards are selected annually by EDN readers.

Due to its flexibility to program different analog and digital components, it has been used in various applications including physiological monitoring because of its capability to perform complex analog as well as digital functions. The PSoC has been used in building an ECG monitoring. One such project has been described in "*ECG Meter using PSoC*" by Dr. Altenburg. It is designed by programming Delta Sigma ADC, Low pass filter, counters as well as a serial communication module on the chip. It transmits the data to the PC where the cardiac signal is displayed.

2.3 Related Research in Physiological Applications:

Wearable, unobtrusive devices that monitor the patients for vital signs are currently being researched extensively. Wireless sensor network has an enormous potential in this area. Harvard University is currently working on project "Codeblue" [18], which is developing hardware as well as software platforms using for medical sensor networks using motes supplied by Crossbow Inc. They have also developed medical sensors such as EKG[19], pulse-oximeter and motion-activity sensor based on the popular MicaZ and Telos mote designs. It performs real time data collection on the patients' physiological status, which is monitored and/or stored at the base station. They are also studying the issues related to wireless medical-sensor networking, such as node mobility, patterns of packet loss, variations in data rate etc.

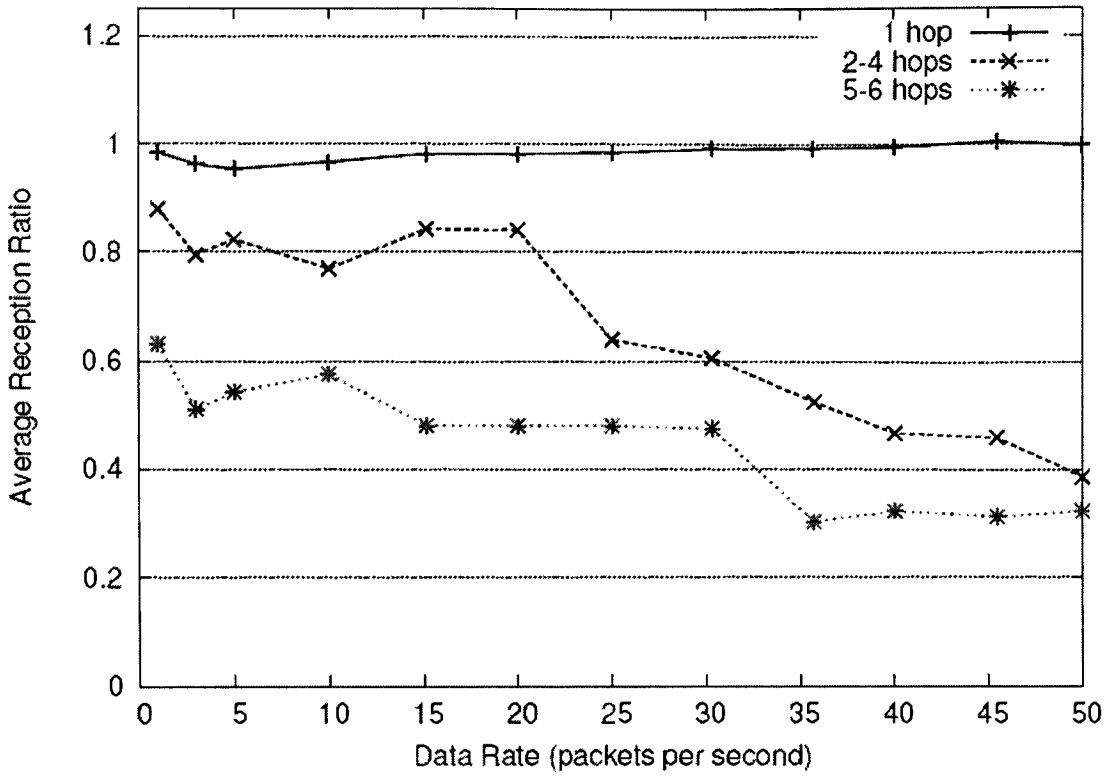


Figure 2.1: The effect of increasing data rate and hop count on reception ratio: Increasing the transmission rate leads to degradation in reception rate due to dropped packets. [18]

During their study, it was observed that the packet reception ratio i.e. the number of received packets divided by the number of transmitted packets [18] decreased with the increase in the data rate, with an exception of a single hop where the reception ratio remains pretty good up to a data rate of 50 packets per second. A Similar trend is observed with multiple senders with a single receiver (figure 2.2). This is because of limitation of the available bandwidth for each node. Also, increased data rate result in packet queues on each node and might eventually force packets to be dropped.

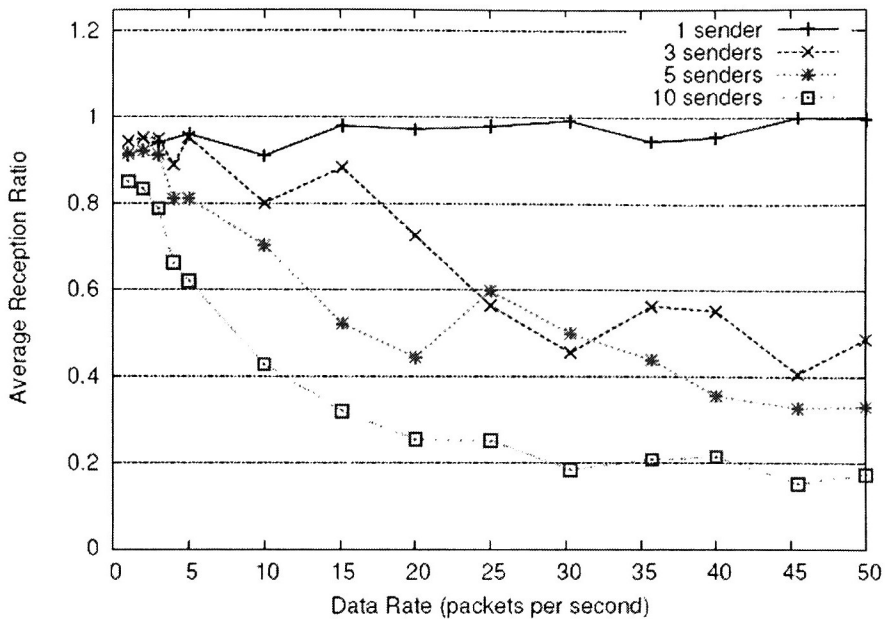


Figure 2.2: Effect of increasing data rate and number of senders with 1 receiver[18]

From figure 2.1and 2.2 it is observed that for a high reception ratio, a low data rate is encouraged, of about 5 packets per second or less for 10 senders. This could be acceptable incase of blood pressure or pulse oximetry sensors, but not for an ECG monitoring system, which requires sending constant packets of the sampled cardiac signal.

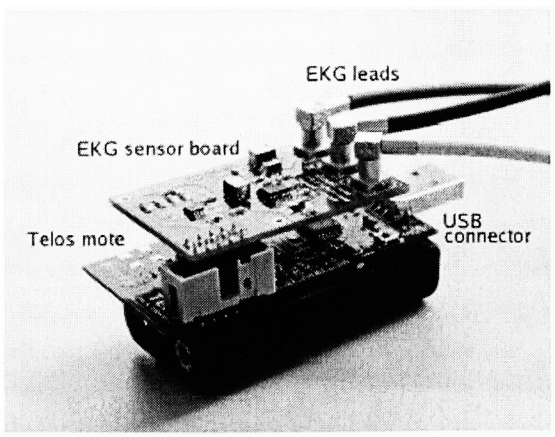


Figure 2.3: Wireless EKG developed by Codeblue [18]

In this thesis we demonstrate that by interfacing a PSoC to MICA2, we are able to develop a wireless ECG monitoring system that sends out data only when the cardiac signal might seem abnormal or send computations of Heart Rate Variability, which is an important marker of unusual activity of the cardiac signal. This reduces the network traffic by reducing the data rate of an ECG monitoring device.

2.4 Cardiac Monitoring Systems:

2.4.1 Heart Rate Variability

Heart Rate Variability (HRV) defined by [3] is "the beat-to-beat alterations in the heart rate" or RR-Interval (figure 2.4) fluctuations. Incase of a healthy person, the beat to beat interval varies slightly as a person inhales and exhales. Inhalation causes cardio acceleration and exhalation causes cardio deceleration. Hence, there should be a periodic variation in the intervals (figure 2.5).

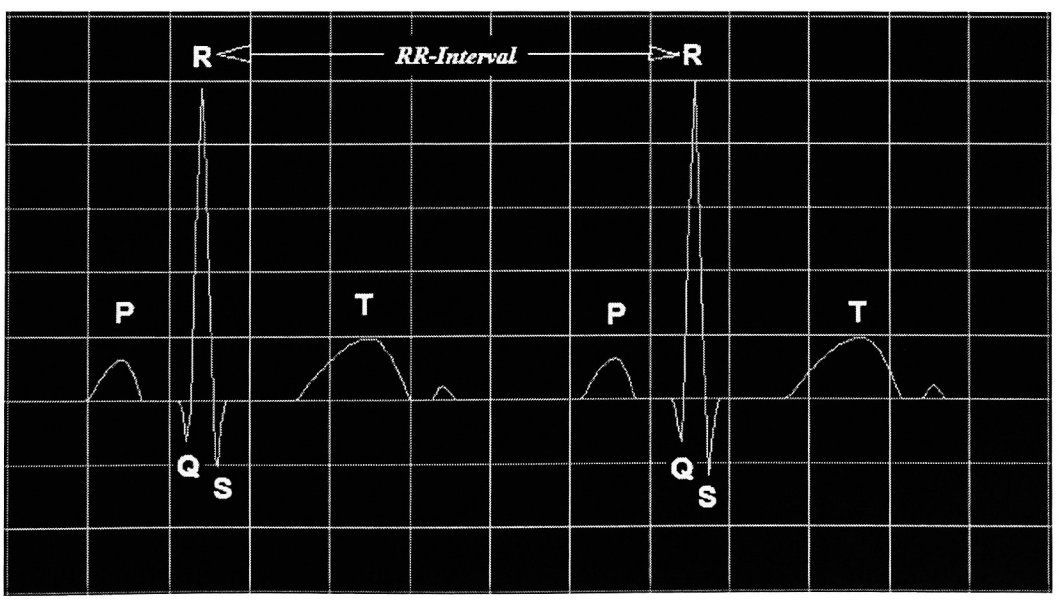


Figure 2.4: Cardiac Signal, Illustrating the RR Interval.

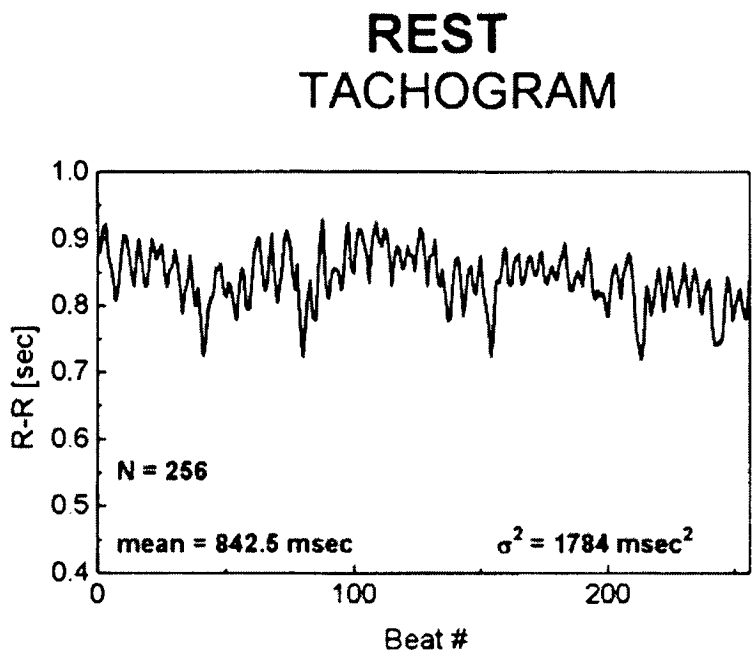


Figure 2.5: Tachogram of 256 consecutive RR values in a normal/ healthy subject at supine rest [20]. Notice the periodic increase and decrease in the RR-Interval. The calculated standard deviation (in msec^2) is 1784

HRV analysis is being actively researched for risk assessment. Studies have linked HRV to stress, inner emotional states, such as anxiety, anger etc. as well as the ability to predict survival after heart attack. Reduced HRV has been linked to the prediction of an impending heart attack for patients with MI. (MI: abbreviation for myocardial infraction in other words a heart attack)

2.5.2 Measuring HRV

HRV is assessed using different techniques. One common statistical technique is by measuring the RR-Interval of the cardiac signal and computing the standard deviation of the RR-intervals over a short term period of 5 minutes or over 24 hours. A low standard deviation value indicates a low HRV.

Among some of the other methods of analyzing the HRV [3] are:

- pNN50 index: The number of times in a period of about 60 minutes two consecutive RR-Intervals vary over 50msec.
- rMSSD index: Computing the root-mean-square of the difference between the consecutive RR-intervals.
- MAX-MIN or peak-valley quatification : The diffence between the shortest and longest RR-intervals during inspiration and expiration respectively .

3. MICA2 Hardware/Software

3.1 Hardware Platform

As mentioned in chapter 2, several different hardware platforms have been developed for power consumption, flexibility, robustness as well as communication and computational capabilities.[4] The general architecture of the MICA2 platform, supplied by Crossbow Inc. is shown in figure 3.1 and the snapshot of the mote without the sensor board is shown in figure 3.2. It uses an Atmel mega 128Lmicroprocessor as its central processor, a Chipcon CC1000 tranceiver, an external flash memory of 512KB. The pins of the microcontroller are brought out to a 51 pin expansion connector, of which some of them are analog I/O's that are inputs to an A/D converter (one of the peripheral feature of the microcontroller). The sensor board can hence directly connect the analog equivalent of the sensed element to the A/D converter (of the microcontroller) for it to be sampled. It works on 2AA batteries.

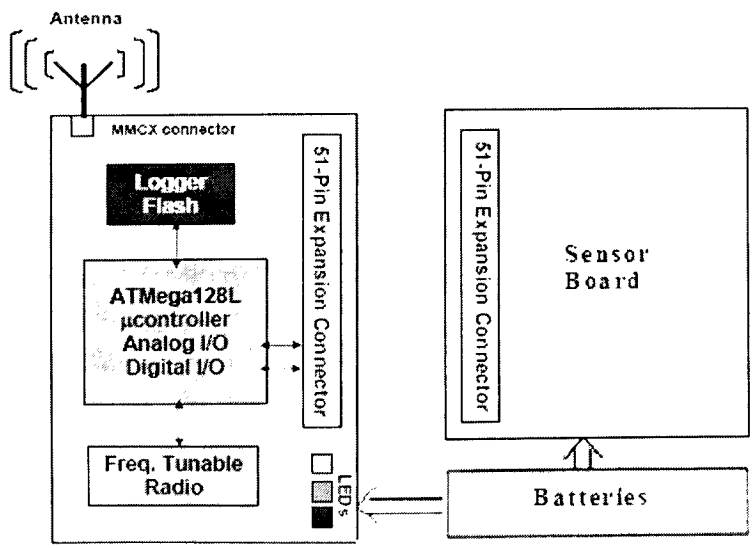


Figure 3.1: MICA2 Platform [40] with interfaces to the sensor board and the batteries.

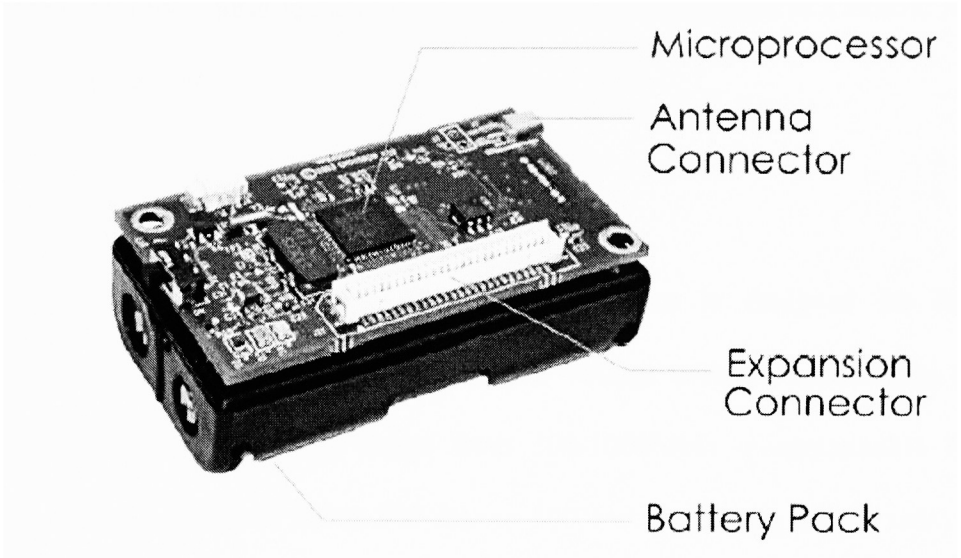


Figure 3.2: Snapshot of MICA2 mote supplied by Crossbow Incorporated. [40]

Microcontroller:

The ATmega128L [21] is a low power, 8-bit microcontroller that uses CMOS technology. It is based on AVR enhanced RISC architecture, that achieves high computational performance, by executing powerful instructions in a single clock cycle. The architectural design achieves optimum power consumption versus processor speed. It has useful peripheral features

It has Harvard architecture –separate memory for data as well as program code. It has 128KB of reprogrammable flash memory for program memory, 4KB EEPROM, 4KB (of SRAM) data memory which can be extended up to 64KB using an external memory. It operates at speeds of 0-8 MHz and voltage between 2.7 to 5.5V.

It has peripheral features such as timer/counters, ADC's, serial communication blocks that support USART, I2C protocols and master/ slave serial interface. It has different sleep modes such as idle, power-save, power-down, standby, ADC noise reduction and

extended standby. These features make the AT128L very useful as a central processor in a wireless sensor node.

Transceiver:

CC1000 [22] is a single-chip, RF transceiver that is designed for low current consumption, specifically for low power, low-voltage wireless applications in mind. It has programmable frequency range from 300-1000MHz, programmable in steps of 250Hz. It has a range of about 300 feet to 500 feet depending on the radio band. The CC1000 can be programmed serially, which makes it easy to be programmed via the microcontroller. It has a small size, low operating voltage of about 2.1 to 3.6V and requires very few external components.

Flash Memory:

MICA2 uses AT45DB041 [23], which is a 512 KB, 2.5-volt serial flash memory. Due to the simple serial interface design, the reliability increases, the switching noise decreases and the package size as well as the pin count decreases. It is designed for low power dissipation, with a current requirement of only 4mA during read and 2uA CMOS standby current

Power analysis:

The power consumption of the MICA2's microcontroller and the RF chip has been listed below in table 3.1 and active power has been enumerated in Chart 3.1

Mote Type	*Mica 2*
Microcontroller	
Type	ATmega128
Program memory (KB)	128
RAM (KB)	4
Active Power (mW)	33
Sleep Power (μW)	75
Wakeup Time (μs)	180
Communication	
Radio	CC1000
Data rate (kbps)	38.4
Modulation type	FSK
Receive Power (mW)	29
Transmit Power at 0dBm (mW)	42
Power Consumption	
Minimum Operation (V)	
Total Active Power (mW)	89

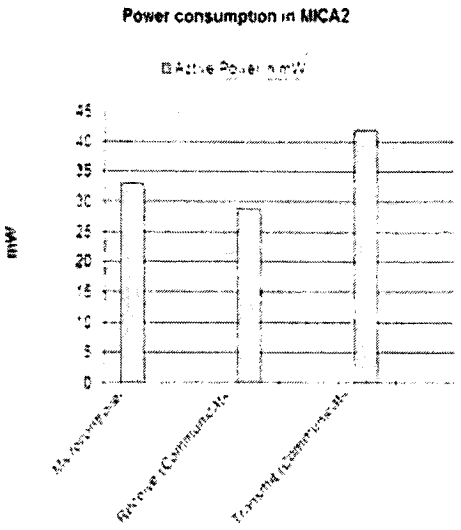


Table 3.1: Power consumption table for MICA2. [44], Illustrated with Chart.

As shown in Table 1, the transmission power of the radio is 42mW, which is a major percentage of the total power consumption (89mW) of the node. Hence, reducing the wireless transmission time would result in a considerable amount of reduced power consumption.

3.2 Mica2 Software Platform:

3.2.1 TinyOS:

TinyOS [8] is an Operating System designed specifically for WSN. The 4 KB of RAM space, which is shared among the stack, global variables and all the static variables, used in the TinyOS components puts a very tight memory constraint on the system. Therefore, it is designed such that it minimizes the code size and facilitates faster implementation using a modular architecture. It wires together several different 'components' in a 'configuration file' that create a mote application. For example, figure 3.3 illustrates the

wiring together of components, BlinkM, LedsC and SingleTimer with Main for building an application that toggles the Leds every clock interrupt using the Timer. This application has been built by the TinyOS developers and is available for TinyOS users.

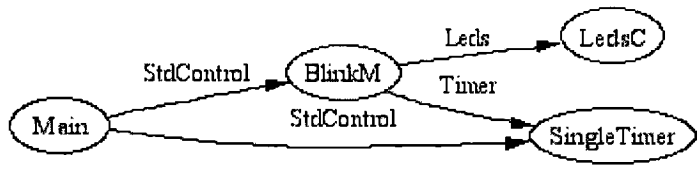


Figure 3.3: Components wired together to build the Blink application [25]

TinyOS implements an event-driven execution model, which enables fine-grained power management and at the same time allows the necessary scheduling flexibility required in WSN application. [tinyos.net] Long procedures are scheduled to be executed in series and are called tasks. Interrupts (or events) can preempt the scheduled tasks. Lengthy operations must be spread over multiple tasks, such that each task is kept short enough to ensure low task latency. It performs split-phase operations [26] allow for concurrency with low memory overheads in contrast to a thread-based concurrency model. Split-phase operation: Uses 'commands' to request the execution of a certain operation, and returns immediately to execute the scheduled sequential tasks. The completion of the requested operation will then be signaled by an 'event'. Such a no-blocking operation results in low-memory resource overhead. Example: 'send' command requests the sending of packets. Once the packets have been transmitted, the 'sendDone' event is executed.

3.2.2 NesC:

TinyOS as well as customized application on MICA2 are written in NesC programming language. It supports a programming model that allows component-based programming, event-driven execution thereby supporting a flexible concurrency model. With nesC, performing program optimization and compile-time data race detection, it reduces the memory utilization and also helps detect potential bugs early on [26].

NesC is like an extension to C. The programming of the low level abstraction is done in C, but nesC has additional features – the low level abstractions (written in C) are organized as functions, which are then organized as components and allow concurrent operation.

There are two types of components:

- Modules, which describe the components using code, in three program blocks: events, commands and tasks and
- Configuration, that wires the different components, to build an application.

However, the split-phase operation of TinyOS makes it difficult to program in nesC [26]. The TinyOS developers have provided the users with many programmed low-level components. These could be used to build custom applications, by simply wiring the components together.

4. Programmable System-on-Chip (PSoC) ™

4.1 PSoc Hardware Framework:

The PSoC Family supplied by Cypress, consists of an array of configurable analog and digital blocks (highlighted in yellow in figure 4.1) with a central micro-processor. It is very useful in building customized designs on one chip. Figure 4.1 shows the block diagram of a PSoC device, part-number: CY8C274x3.

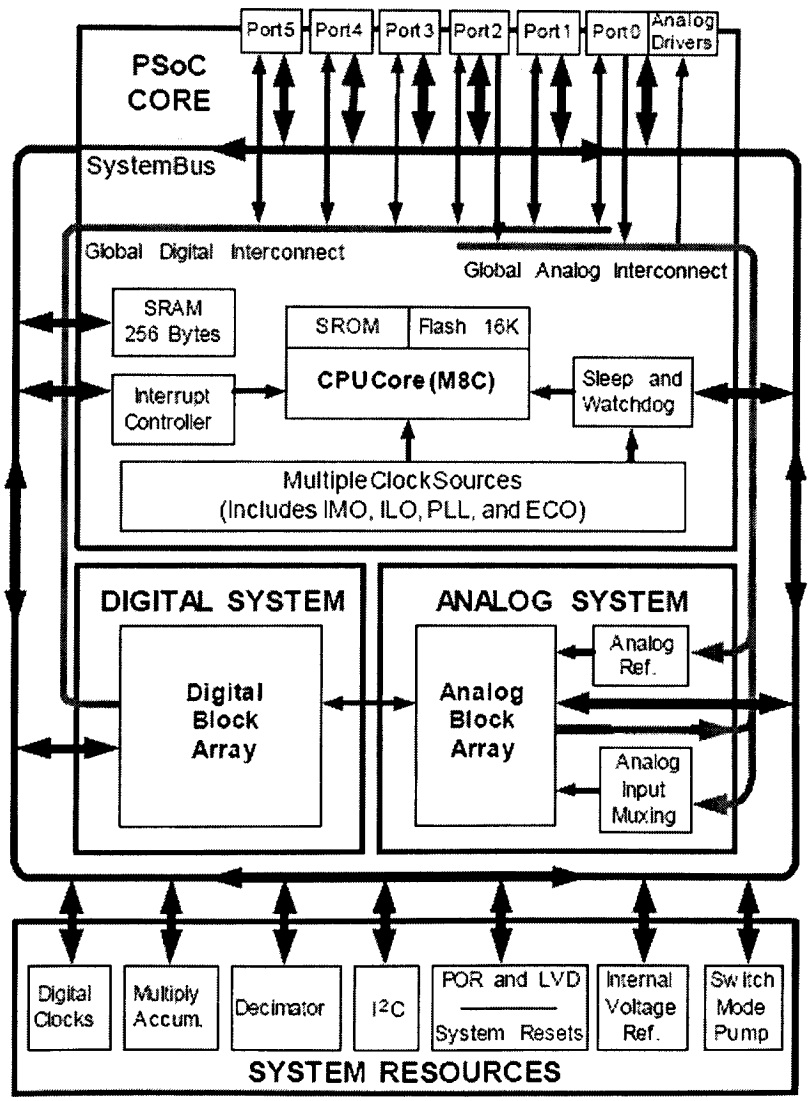


Figure 4.1: PSoC Block Diagram of CY8C274x3 [27]

As can be seen from the figure 4.1, PSoC architecture has a central processor, Data SRAM, Flash program memory, configurable analog and digital blocks, programmable I/O ports, clock generator and some other system resources.

The M8C microprocessor is a powerful processor with an 8-bit Harvard architecture and can operate on speeds as high as 24MHz. It utilizes an Interrupt Controller that provides interrupt management for 17 interrupt vectors. Sleep and Watchdog timer is also utilized to provide timed and protected program execution.

PSoC devices have multiple clock sources, clock dividers and multipliers that provide flexible clock generation to the microcontroller as well as the analog and digital blocks. It has a 24MHz Internal Main Oscillator (IMO) and a 32KHz Internal Low Speed Oscillator (ILO). Also, a 32.768KHz external crystal oscillator along with a PLL (Phased Lock Loop) can be used to generate crystal-accurate clock.

The general purpose input/output pins (GPIO) allow great flexibility for external interfacing, because each pin's drive mode can be selected from eight options and it also has an option to generate an interrupt incase of predefined events at a pin.

The PSoC family has different device groups based on the digital and analog pin count, the program memory, data memory and the number of configurable digital and analog blocks. Table 4.1 shows the available device groups.

PSoC Device Group	Digital IO (max)	Digital Rows	Digital Blocks	Analog Inputs	Analog Outputs	Analog Columns	Analog Blocks	Amount of SRAM	Amount of Flash
CY8C29x66	64	4	16	12	4	4	12	2 KB	32 KB
CY8C27x43	44	2	8	12	4	4	12	256 Bytes	16 KB
CY8C24x94	50	1	4	48	2	2	6	1 KB	16 KB
CY8C24x23	24	1	4	12	2	2	6	256 Bytes	4 KB
CY8C24x23A	24	1	4	12	2	2	6	256 Bytes	4 KB
CY8C22x13	16	1	4	8	1	1	3	256 Bytes	2 KB
CY8C21x34	28	1	4	28	0	2	4*	512 Bytes	8 KB
CY8C21x23	16	1	4	8	0	2	4*	256 Bytes	4 KB

* Limited analog functionality.

Table 4.1: Available PSoC Device groups. [27]

These device groups are further divided into different parts depending on the package type and pin count. In our work we use PSoC chip CY8C27443, 28 pin PDIP package, it has 8 configurable digital blocks and 12 configurable analog blocks. It has 256 bytes of RAM and 16KB of program memory. The range of supply voltage for PSoC varies from 3.0V to 5.25V. The digital and analog blocks within the PSoC device has the same supply voltage range.

4.1.1 Digital Blocks:

The block diagram of the digital block is shown in Figure 4.2 Each block has a 8-bit resource, therefore it can be combined with the other blocks to build 16, 24 or 32 bit peripheral block. For example, a 16, 24, 32 bit timer or a counter. The different peripheral blocks that can be configured in these digital blocks are the

- 16-32 bit Pulse Width Modulators (with and without dead bands)
- 8, 16, 24 or 32 bit Counter
- 8, 16, 24 or 32 bit Timers
- Serial Communication blocks such as UART, I2C, SPI
- A 8 to 32 bit CRC (cyclic redundancy check) checker/generator
- Or a pseudo-random sequence generator

These blocks can be placed by the user in the desired blocks, the signals from the blocks can also be routed to the GPIO.

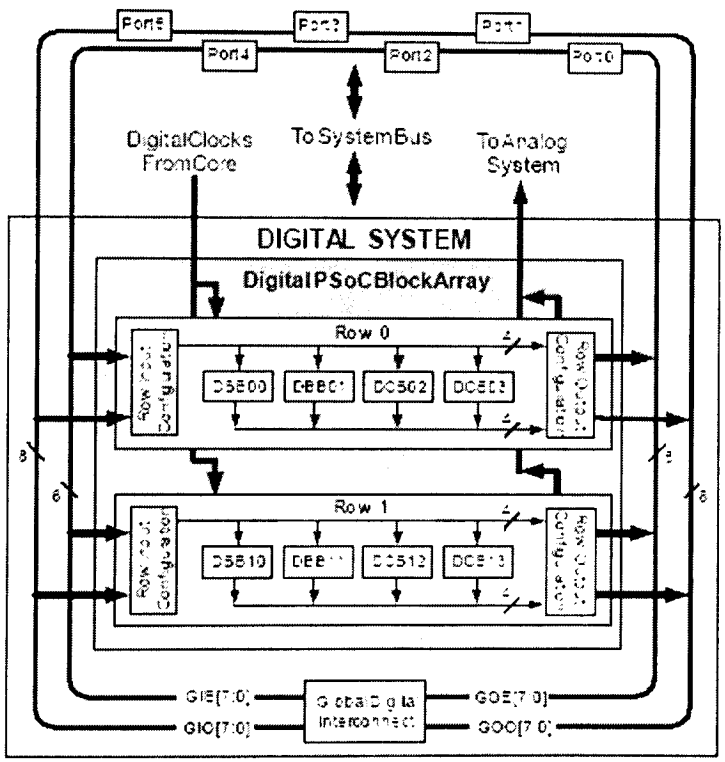


Figure 4.2: Block diagram of the configurable digital block [27]

4.1.2 Analog Blocks:

The Analog blocks are what makes the PSoC very useful in out application. CY8C27443 has 12 configurable analog blocks. The flexibility of placing and routing the blocks

internally are specifically designed keeping the common embedded systems' design requirements. These analog blocks can be applied to design customized functions. The block diagram of the configurable analog blocks is shown in figure 4.3.

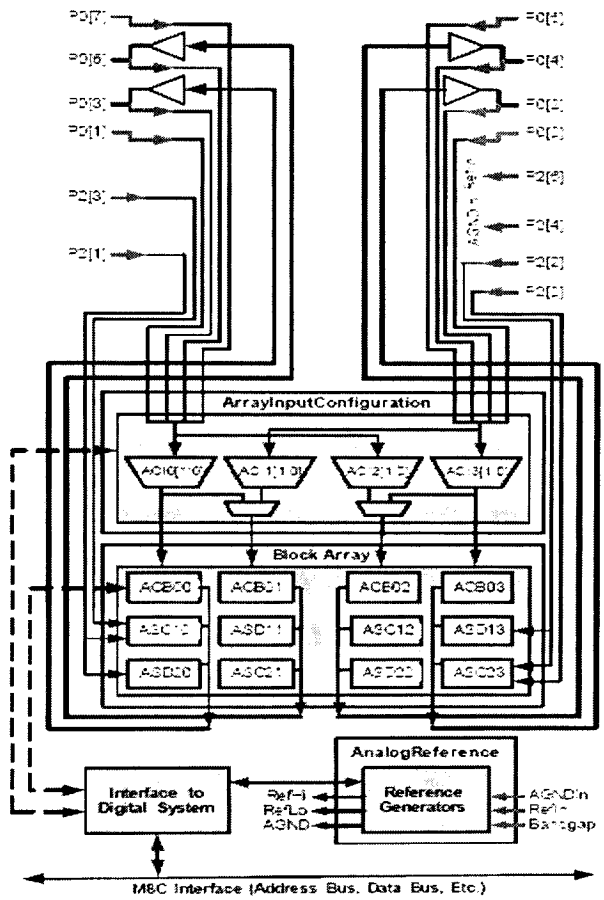


Figure 4.3: Block diagram of the configurable analog system [27]

It sometimes needs a little bit of experience and some creativity to be able to use the provided functions to build a required customized application. Some of the common PSoC functions that can be build using the analog blocks are listed below:

- Variety of Analog to Digital Converters, such as Delta-Sigma, Incremental, Successive Approximation etc with a selectable resolution of 6 to 14 bits

- Analog Filters: band-pass, low-pass, notch
- Programmable Gain Amplifier that can be programmed up to 48 times
- Instrumentation amplifier with a selectable gain up to 93 times.
- Comparators
- Digital to Analog Converters
- Modulators

4.2 Programming Environment:

The PSoC design software needs to be different from those used for programming traditional fixed function microprocessor, because of the configurable analog and digital hardware blocks. The PSoC Designer Integrated Development Environment (IDE) developed by Cypress Microsystems is an innovative software development environment for PSoC. It is a GUI-based design suite that simplifies the designing of the configurable blocks, provides the libraries for developing customized application code, provides advanced debugging tools using an emulator and supports the programming of the PSoC chip.

Figure A.1 (Appendix) shows the block level diagram of the programmable analog as well as digital blocks along with interconnects, which can be programmed as per the desired routing of the signals.

The PSoC application code is developed in three main stages which is shown in figure 4.4.

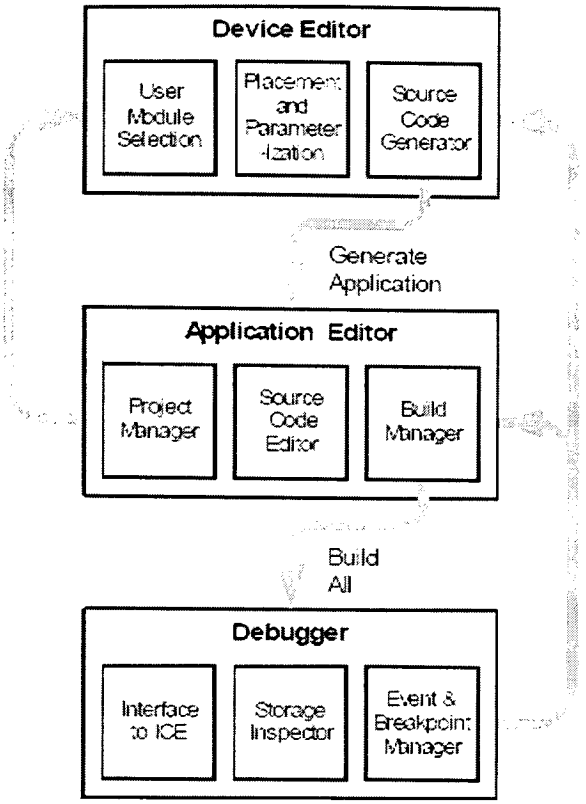


Figure 4.4: User Module and Source code development flow in PSoC Designer [27]

The Device Editor allows the user to choose the functional component (e.g. ADC/ filter/ timer/ counter etc) for each of the analog or digital blocks (shown in figure A.1 in appendix). The user has an option to place these components within the permitted blocks and route the signals by making or breaking the programmable interconnects or by programming the multiplexers as per the requirement of the design. The device editor also allows the user to set the parameters (user-defined-registers) of the analog and digital component. In PSoC Designer IDE, the standard components have libraries of pre-built and pre-tested hardware functions. The corresponding libraries of the selected components are generated and added to the projects in the device editor before moving forward to the application editor. The design must pass the Design Rule Check and then proceed.

Application Editor

The application editor allows the user to write the code either in C language or assembly language. The analog and digital blocks must be initialized in the code using the generated component libraries. It also gives access to the interrupt service routine. The code is then assembled/ compiled, linked and built without any errors before moving to the debugger to evaluate the design on an emulator.

Debugger

Debugging is the last step in the development process of PSoC. PSoC designer IDE provides an In-Circuit-Emulator (ICE), which allows debugging at the full operating speed of PSoC. A HEX file is downloaded onto the ICE and it has advanced debugging features in addition to the single-step, breakpoints and watch-variables. It allows the user to define complex breakpoint events and also a large trace buffers that allow the user to monitor registers, memory locations etc.

5. Interface Design

This chapter discusses the implementation of the interface between MICA2 and PSoC in detail. It focuses on the hardware setup and programming of the PSoC and MICA2 required for the implementation of an interface between them.

MICA2 and PSoC, both support Universal Asynchronous Receive/Transmit (UART) serial communication. MICA2 supports a baud rate of 57600 bits per second. The UART pins in the MICA2 mote is brought to the 51 pin connector. However, since this is a development phase, direct soldering on the machine soldered components of MICA2 sensor node (called 'mote') is avoided. Instead a breakout board is used, which brings out the signals at the 51 pin connector. The breakout board, called the Serial Gateway MIB510CA (figure 5.1) is made available by Crossbow Technology Inc. The serial gateway provides a RS232 interface, converting the UART signals to RS232 using MAX-3223I chip. It has a 9-pin RS232 connector. This board is generally used for interfacing the mote with the computer at the base station or for programming the motes, using its on-board processor.

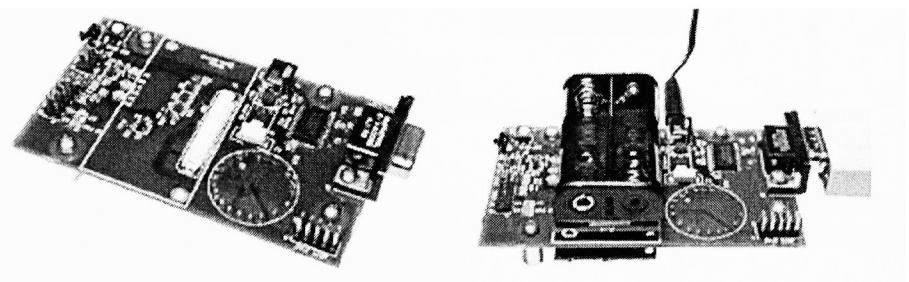


Figure 5.1 (a) Figure 5.1 (b)

Figure 5.1: (a) MIB510CA serial gateway, made available by Crossbow Technology Inc. (b): with attached MICA2 mote, serial connector and external power supply. [29]

5.1 Programming PSoC to support UART serial communication.

For the PSoC to be interfaced with MICA2, its digital blocks need to be programmed to support UART communication at a baud rate of 57600 bps. This section describes in detail the programming of digital as well as analog blocks of PSoC for sampling the sensor signal and then transmitting it serially to MICA2. Figure 5.2 illustrates the peripheral blocks required to be programmed in PSoC.

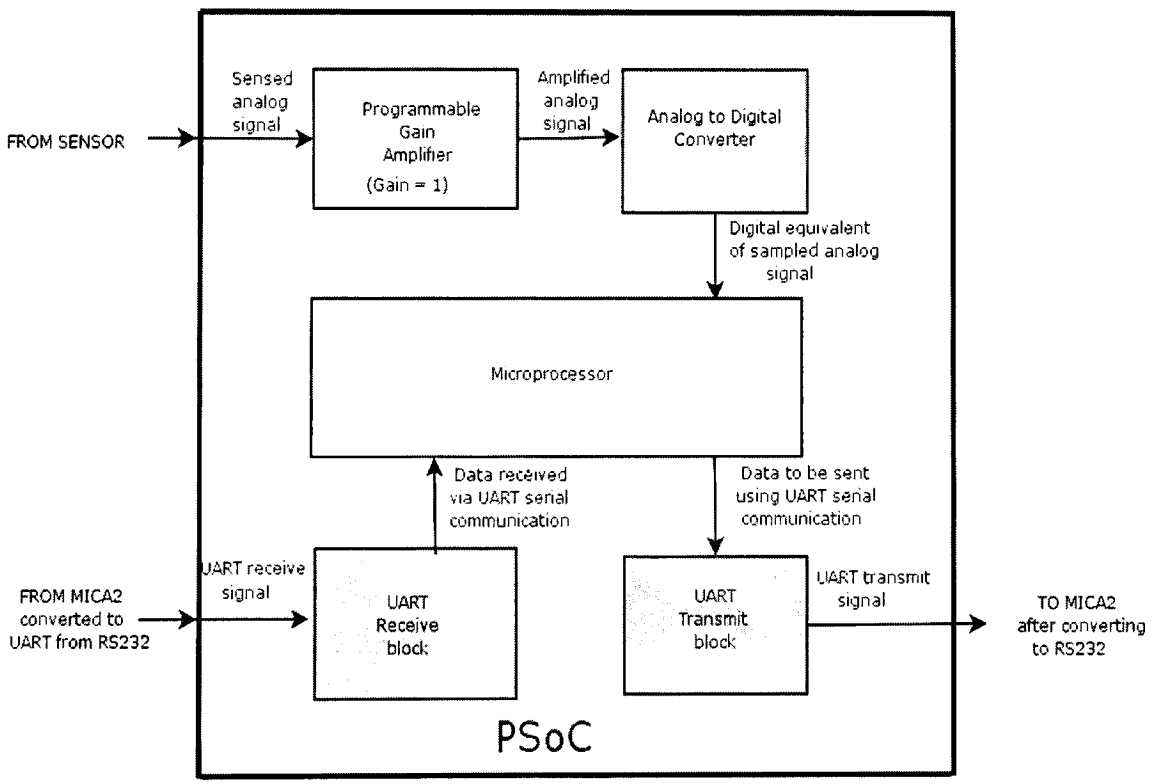


Figure 5.2: Block diagram showing the digital (blue/ below the MCU) and analog (green/ above the MCU) peripheral blocks required to be programmed in the PSoC for interfacing with MICA2.

5.1. 1: Amplifier.

The Programmable Gain Amplifier (PGA) uses only 1 peripheral (analog) block. It is used to provide high input impedance to the transducer. Figure 5.3 shows the PGA block diagram in the PSoC.

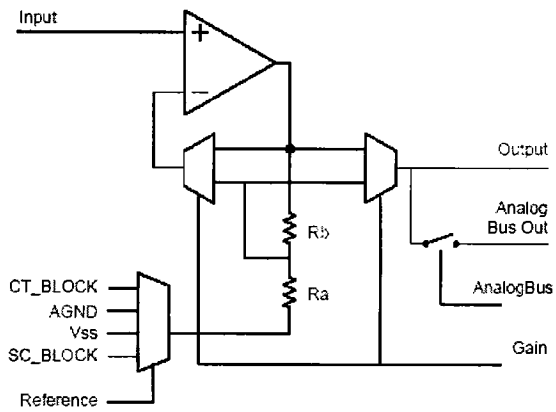


Figure 5.3: PGA Block diagram in PSoC [30]

In our design the PGA is programmed for unity gain, the reference is selected as AGND and the analog bus is disabled. For a unity gain, the output MUX is connected to the top of the resistor string and feedback to the inverting input is connected to the resistor tap. The transfer function of the PGA is therefore as follows:

$$V_O = (V_{IN} - V_{GND}) \cdot \left(1 + \frac{R_b}{R_a}\right) + V_{GND}$$

The Device Editor of the PSoC automatically programs the appropriate resistor values depending on the user-defined gain value.

5.1.2: Analog to Digital Converter (ADC)

The PSoC has a number of different ADC's to select from depending on the required sampling rate, resolution, Signal to Noise Ratio (SNR) and depending on the number of

peripheral analog and digital blocks available. It offers Incremental Type (algorithmically equivalent to a Dual Slope type), Successive Approximation as well as Delta-Sigma ADC's.

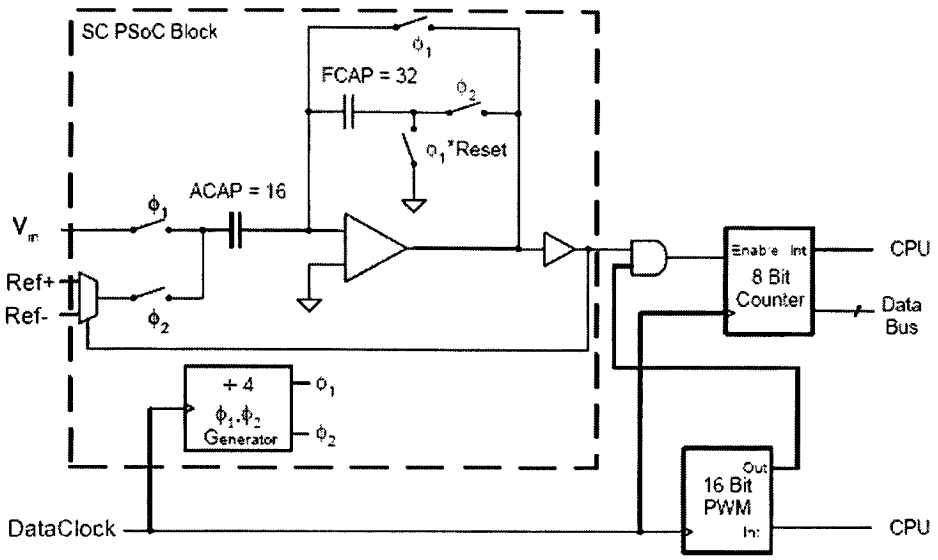


Figure 5.4: Simplified Block diagram of the Incremental ADC in PSoC [31]

In our design a Variable Resolution Incremental ADC has been selected, because of its flexibility to finely tune the sample rate and change the resolution as per the application requirement during the development stage. It uses integrator, comparator, reference signals, counter and a pulse width modulator (PWM). In our design the ADC has been programmed for a resolution of 11 bits and the data clock 2.4MHz. It uses 1 analog peripheral block and 3 digital peripheral blocks for 8 bit counter and 16 bit PWM.

The algorithm is similar to that of a dual-slope ADC, whereby an unknown voltage is integrated over a fixed amount of time and then a known reference signal is 'de-integrated', for a variable period of time. The measured variable time helps estimate the unknown voltage. The variable period in this case is kept track by using the counter and

the integrate time is kept track using the PWM. The PWM also gates the clock to the counter.

5.1.3: UART Receive/ Transmit Blocks:

UART Receive/ Transmit blocks use 2 digital peripheral blocks, one for receiving and the other for transmitting. It is required to be programmed for a baud rate of 57600 for interfacing with MICA2. The input clock to the UART blocks needs to be 8 times the required baud rate. Hence the clock is 460.8 KHz. The transmit interrupt request is enabled, therefore once a transmit register is empty the microcontroller is interrupted to send more data.

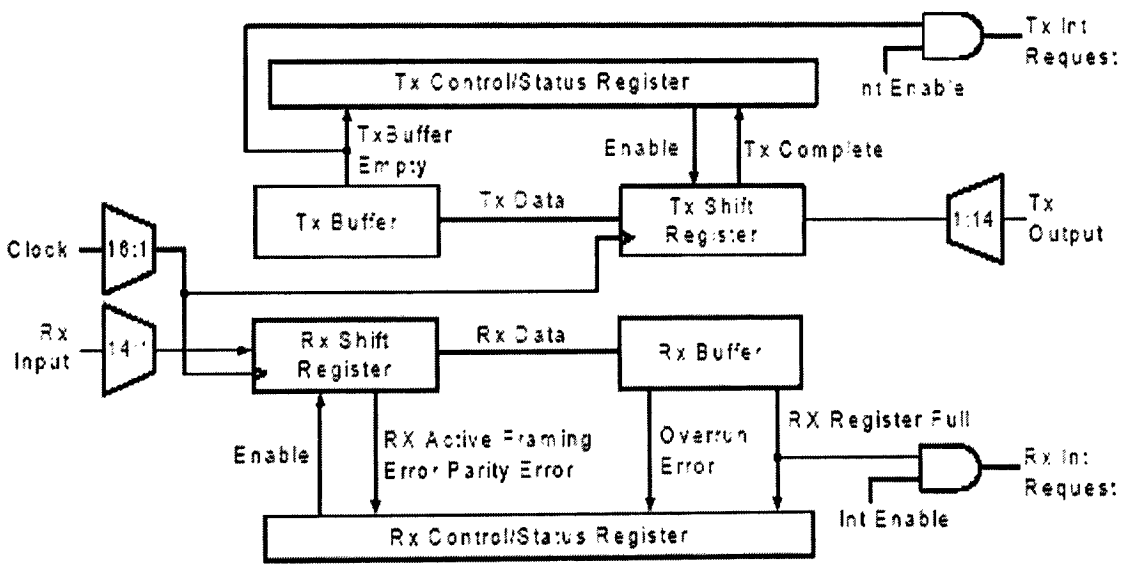


Figure 5.5: Internal Block Diagram of the UART Receive and Transmit blocks.[32]

As seen from figure 5.5, the transmit and receive, control and status registers are different. Since the data comes in as a bit stream, the data is accumulated in a shift register, before sending it out or before reading it

5.1.4: The Device Editor Placing and Routing:

Figure 5.6 shows the Device Editor (DE) after programming the peripheral blocks on the PSoC. The UART blocks, PGA and the ADC are placed in the DE. The interconnects are programmed to route the signals correctly. The global resources and the user-parameters for each of the programmed user modules are also defined in the DE. In figure 5.6 PGA block (is placed in green) receives the analog signal from the sensor. The output of the PGA is routed the input of the ADC blocks (in red). The UART receive/transmit signals from the UART blocks (in blue) are routed to the port pins to transmit to MCU in the node. More description on these can be found in [24].

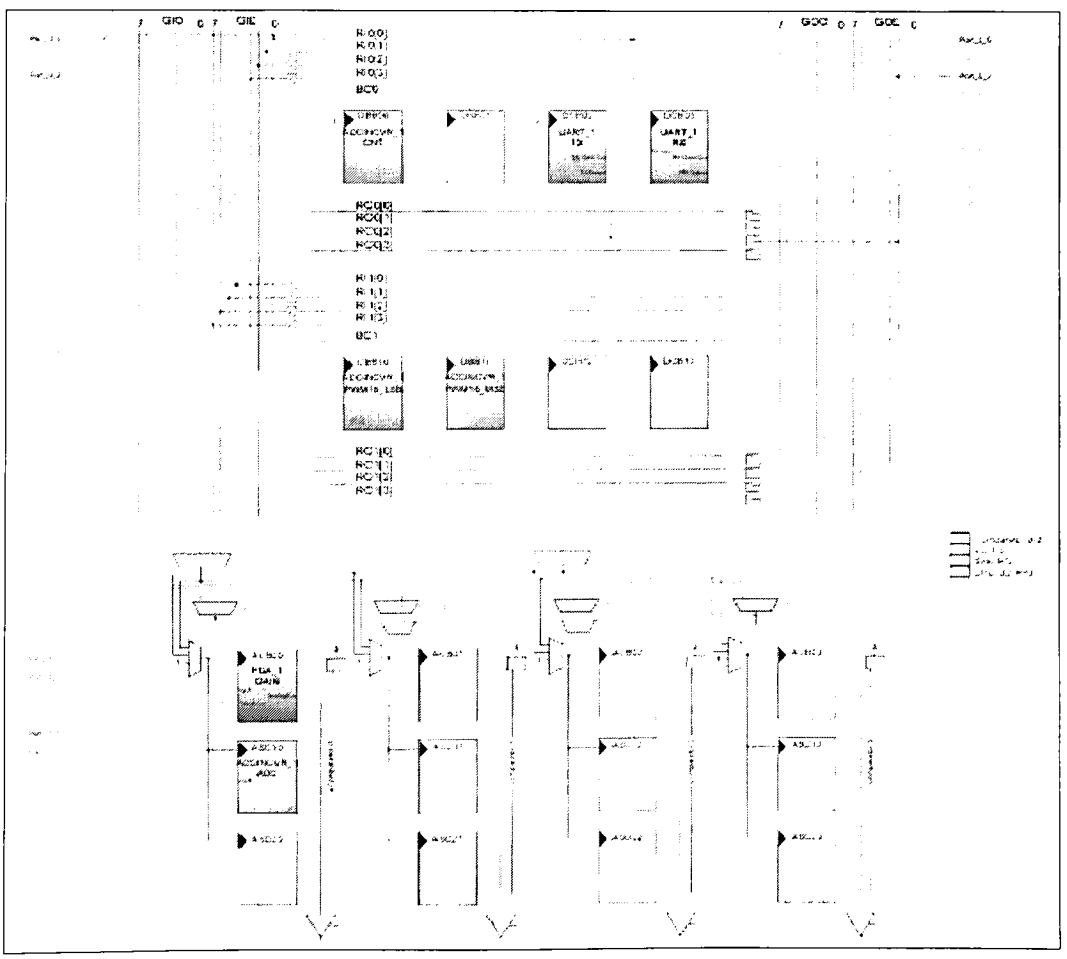


Figure 5.6: Zoomed in view, of the placed peripheral components and the internal routing of the signals.

5.1.5: Application Editor (AE):

The libraries of the peripheral components placed in the DE are generated and available to view and edit in the AE. Each of the user programmed peripheral components must be initialized in the application code.

Using the library functions, the ADC sampling is set such that it will continuously sample the incoming signal at its pre-defined (in DE) sampling rate. Each sample is read into an integer (2 bytes) variable, since the ADC resolution is 11 bits. This variable value is then transmitted serially using the UART library functions.

5.1.6: The Pin Diagram of the PSoC:

After programming of the peripheral blocks, routing the signals to the port pins and initializing the blocks in the AE, the pin diagram of the PSoC is as shown in figure 5.7

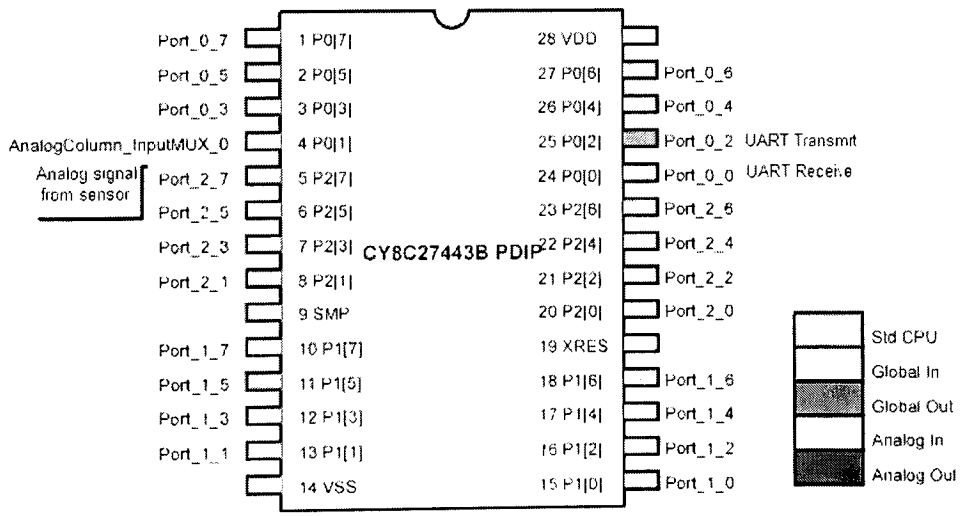


Figure 5.7: Pin Diagram of PSoC after programming the peripheral blocks.

5.2. UART to RS232 Conversion at PSoC end

The UART voltage signals from the PSoC chip vary between 0 to 5 volts as per the TTL/CMOS voltage levels. This needs to be converted to RS232 levels that vary between

12V to -12V. RS 232 is a more reliable serial communication protocol; because it has a greater noise margin.

Figure 5.8 shows the schematic diagram for the conversion of UART signals at the PSoC to RS232 required for interfacing with MICA2. It uses MAX233 IC [33], which performs the level conversion. Figure 5.9 shows a snapshot of MICA2 interfaced with PSoC

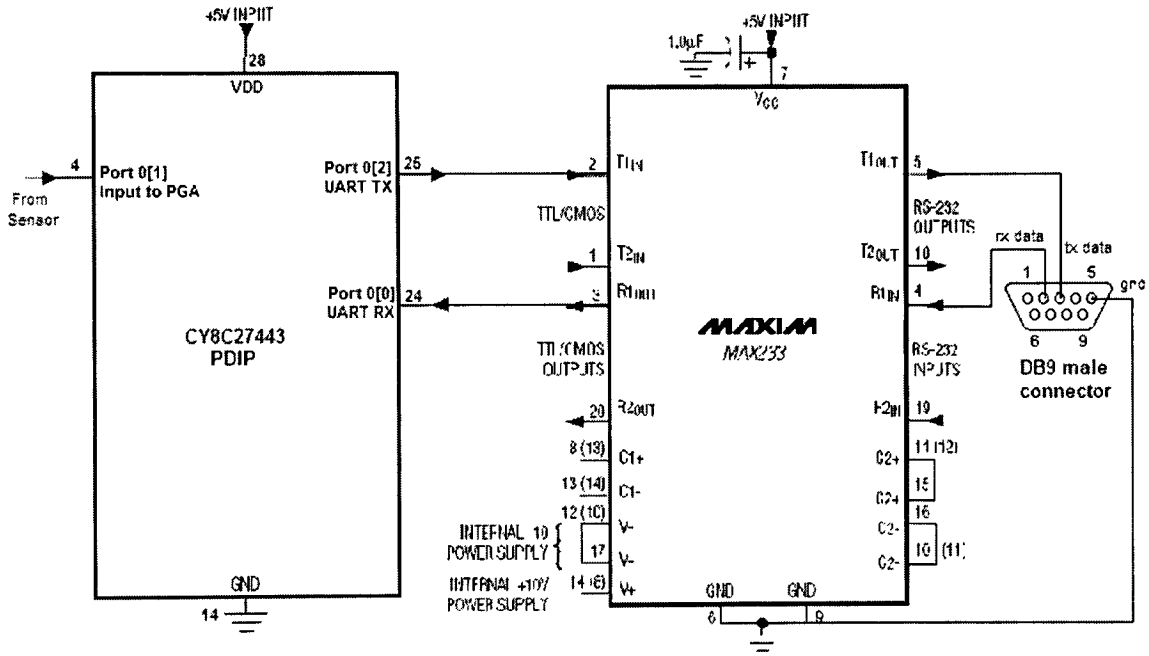


Figure 5.8: Schematic diagram for converting UART signal to RS232 signals.

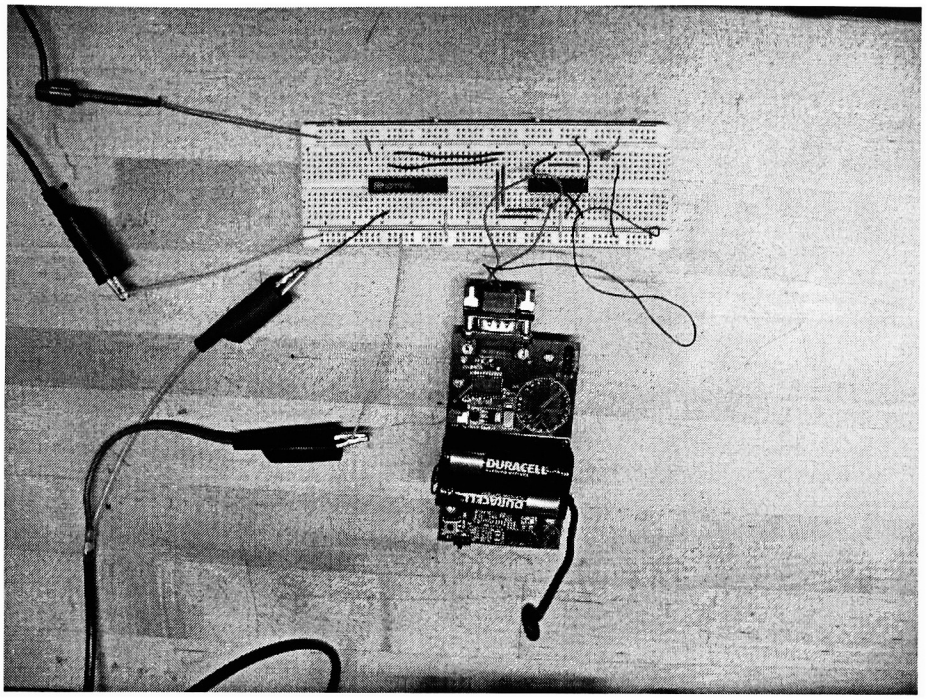


Figure 5.9: Snapshot of MICA2 interfaced with PSoC.

5.3 TOS Message Structure:

TinyOS has a message structure that must be followed while communicating wirelessly or serially with the motes. Just sending a random byte of data will not be understood by MICA2 and will therefore be rejected. Hence, in order to interface PSoC with MICA2 serially, it is important to understand the TinyOS message structure and thereby transmit the data from the PSoC in accordance with it.

5.3.1 Packets:

TinyOS messages are sent in 'packets'. A raw packet has 2 synchronization bytes, the packet type and the TOS message. The contents of a raw packet are illustrated in table 5.1. The frame synchronization byte (0x7E) marks the start and end of the packet; therefore every packet must start and end with '0x7E'. Packet type indicates whether or

not an 'acknowledgement is required' to be sent back after having received the packet correctly or whether the packet is an 'acknowledgement' being sent back to the sender of the original packet. For our design, the packet type sent from the PSoC to MICA2 is a user packet, with no acknowledge required. It has been highlighted in table 5.1. The payload data contains the TinyOS message; it is in this message that the sampled data will be transmitted. The TinyOS message structure has been discussed in the next section.

SYNC_BYTE	Packet Type	Payload Data	SYNC_BYTE
0	1	2...n-1	n

Byte #	Field	Description
0	Packet frame synch byte	Always **0x7E**
1	Packet Type	There are 5 known packet types: • P_PACKET_NO_ACK (0x42) - User packet with no ACK required. • P_PACKET_ACK (0x41) – User packet. ACK required. Includes a prefix byte. Receiver must send a P_ACK response with prefix byte as contents. • P_ACK (0x40) – The ACK response to a P_PACKET_ACK packet. Includes the prefix byte as its contents. • P_UNKNOWN (0xFF) – An unknown packet type.
2...n-1	Payload Data	In most cases will be a TinyOS Message of varying length, which is described below.
n	SYNC_BYTE	Always 0x7E

Table 5.1: Packet Structure in TOS Message Structure [34]

5.3.2 TOS Message Structure:

The TOS message structure, illustrated in table 5.2, has 5 bytes of header, followed by data bytes (also called payload data) and ends with the 2 bytes of CRC. The number of data bytes can be variable and must be indicated in the header. A maximum of 26 data bytes can be sent in a single packet.

The header consists of 5 bytes: the first 2 bytes have the destination address - in our case it will be the UART address. The UART address has been set to 0x007E. The next byte indicates the message type – since our message is being transmitted via UART, the message type is AMTYPE_XUART. Then follows the Group ID for motes of one group to identify each other in an environment with multiple sensor nodes of different networks are present. Since we do not have multiple motes or networks present; we use the default group id, 0x7d. Next byte, the data length indicates the number of data bytes that follow in the payload data.

Address		Message Type	Group ID	Data Length	Data	CRC	
0	1	2	3	4	5...n-2	n-1	n

Byte #	Field	Description
0 - 1	Message Address	One of 3 possible value types: • Broadcast Address (0xFFFF) – message to all nodes. • UART Address (0x007e)– message from a node to the gateway serial port. All incoming messages will have this address. • Node Address – the unique ID of a node to receive message.
2	Message Type	Active Message (AM) unique identifier for the type of message it is. Typically each application will have its own message type. Examples include: • AMTYPE_XUART = 0x0A • AMTYPE_MHOP_DEBUG = 0x03 • AMTYPE_SURGE_MSG = 0x11 • AMTYPE_XSENSOR = 0x32 • AMTYPE_XMULTIHOP = 0x33 • AMTYPE_MHOP_MSG = 0xFA
3	Group ID	Unique identified for the group of motes participating in the network. The default value is 125 (0x7d). Only motes with the same group id will talk to each other.
4	Data Length	The length (*l*) in bytes of the data payload. This does not include the CRC or frame synch bytes.
5...n-2	Payload data	The actual message content. The data resides at byte 5 through byte 5 plus the length of the data (*l* from above). The data will be specific to the message type. Specific message types are discussed in the next section.
n-1, n	CRC	Two byte code that ensures the integrity of the message. The CRC includes the Packet Type plus the entire unescaped TinyOS message. A discussion on how the CRC is computed is included in the Appendix.

Table 5.2: Field description in packets of TOS Message Structure. Source: [34].

The Payload data can vary between 1 and 26 bytes. In our design we keep the payload data constant at 26. The payload is where our sampled data is added. The last 2 bytes of the TOS message structure are the CRC bytes, used for checking the integrity of the received signal at the receiver. The C-code for calculating the CRC of the packet is found in the [octave-tech] document, which explains the TOS message structure in further details.

5.3.3 PSoC and TinyOS Message Structure:

The raw data packets to be transmitted from PSoC to MICA2 using UART look as shown in figure 5.10. It show three raw data packets serially transmitted from the PSoC to MICA2. The data has to be sent in a little-endian format. The synchronization byte (0x7E), packet type (0x42), UART address (0x007E = 7D 5E 00, because of escaping byte), Message type -UART (0A) and Group ID (7D) are highlighted in yellow. The payload data highlighted in blue, green, pink and grey are arranged in an Oscope message structure. This structure is used by Oscilloscope, which is a java-based program that is used to display the sampled data at the base station. The Source mote indicates the motes ID, incase there exists more than 1 mote transmitting data in the n/w. The sample numbers are kept track of and the last sample number is sent out with each packet. Channel indicates the ADC channel, incase of more than 1 element is being sensed at a mote. The dark-green highlighted bytes are the CRC bytes.

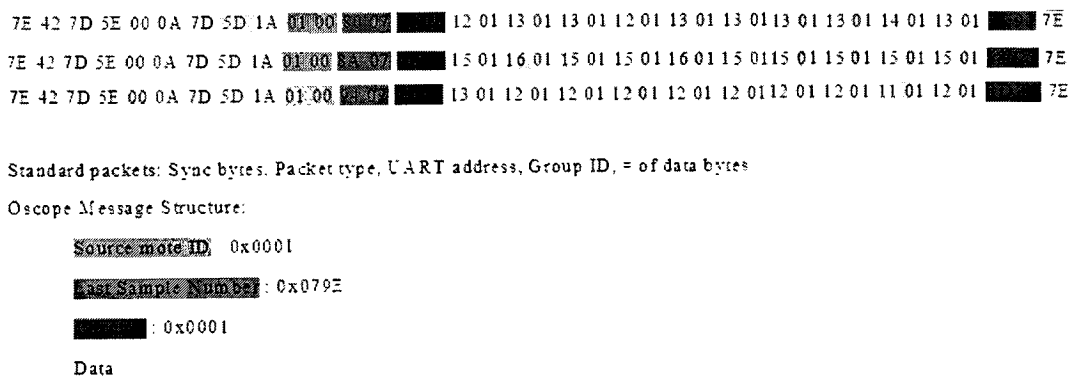


Figure 5.10: Example Raw Data Packets to be sent to MICA2

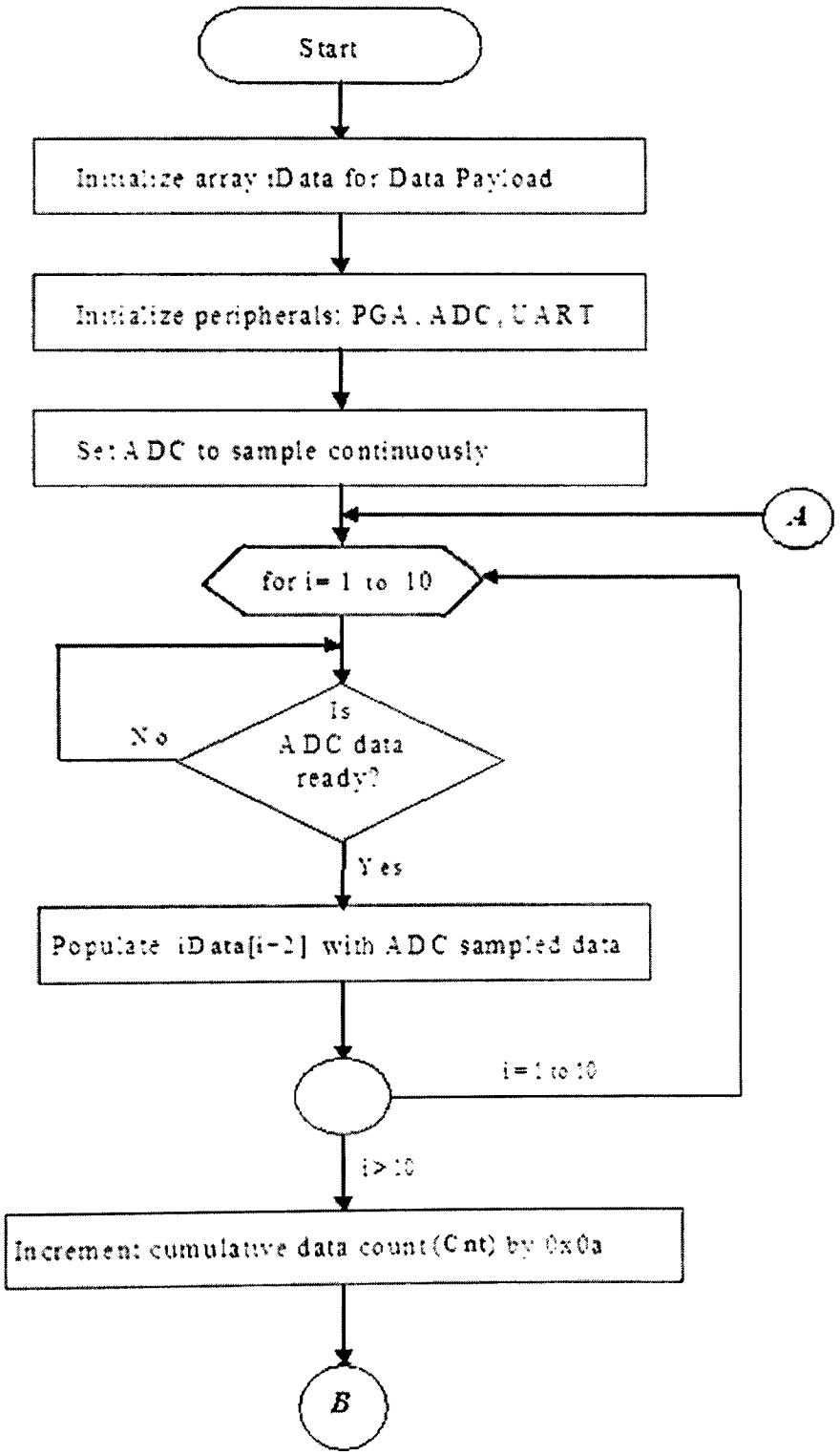
Figure 5.11 describes the program flow of PSoC in order to send the data in the TOS message structure described above. Since the data payload (last sampled number and sampled data) and the CRC are changing with each packet (in figure 5.10), a 13 integer

array, iData, containing only the data payload is initialized with the constant source mote as 0x0001, cumulative data count as 0x0000, ADC channel number (remains constant) as 0x0001 and the sampled data as all zeros.

Ten sampled values are then added to the iData array. A cumulative count variable is used to keep track of the last sample number transmitted. This count loaded into the iData array after the samples have been loaded. The iData array is now required to be converted to a 26 character array because all the sampled data is to be transmitted in the little-endian format (i.e. lsb first and then the msb)

Once the new character array is formed the CRC is calculated. The CRC is initialized with a pre-calculated value of the standard bytes (highlighted in yellow in figure 5.10) because they remain the same with every packet. Therefore only the CRC of the data payload needs to be calculated.

After having computed the CRC, the packet is ready to be sent. The PC now jumps to the UART send function that initially sends out the standard bytes (highlighted in yellow in figure 5.10) then the character array containing the data payload followed by the CRC and the last sync byte.


Start
Initialize array iData for Data Payload
Initialize peripherals: PGA, ADC, UART
Set ADC to sample continuously
A
for i= 1 to 10
No
Is ADC data ready?
Yes
Populate iData[i-2] with ADC sampled data
i= 1 to 10
i> 10
Increment cumulative data count (Cnt) by 0x0a
B

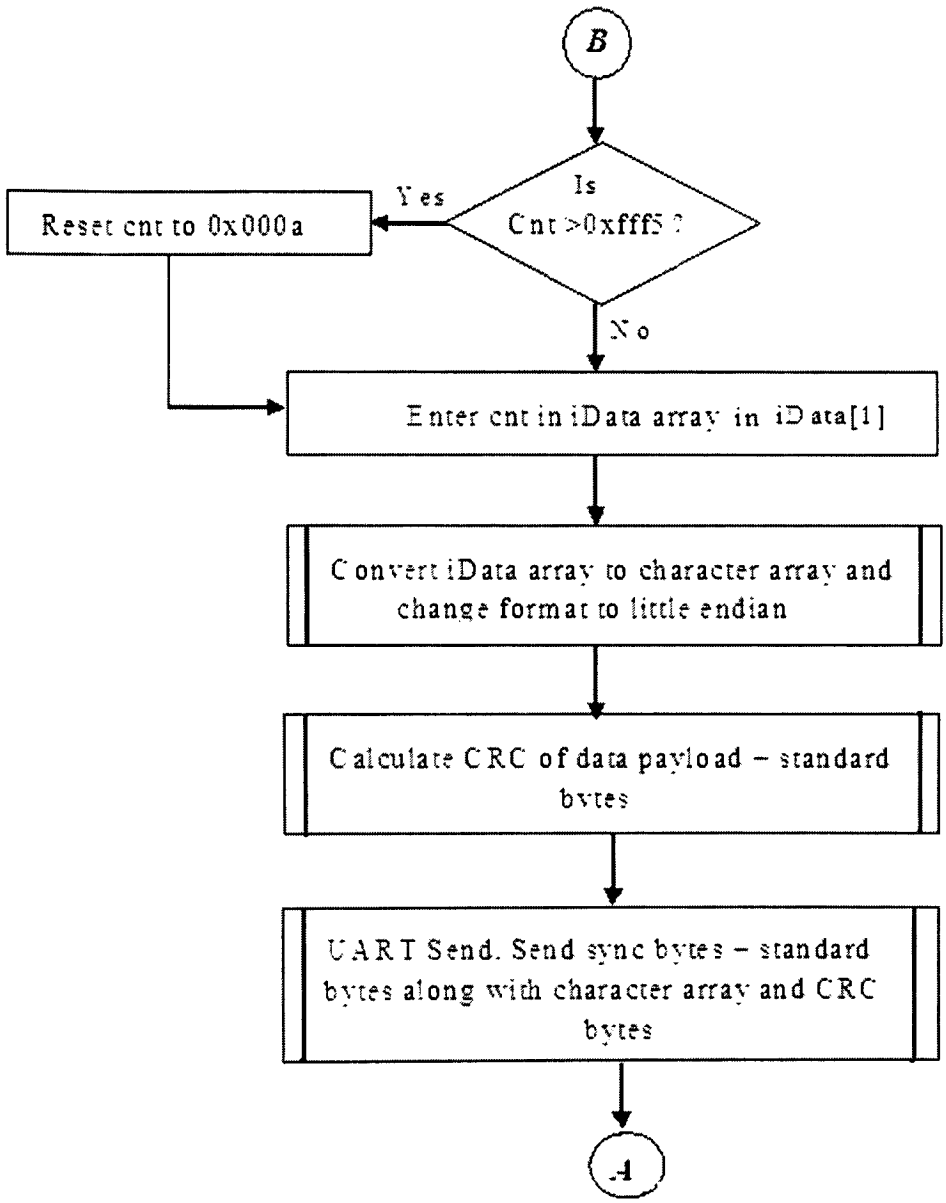


Figure 5.11 : Flowchart describing the PSoC sending sampled data in accordance to TOS Message

5.4 MICA2 programming:

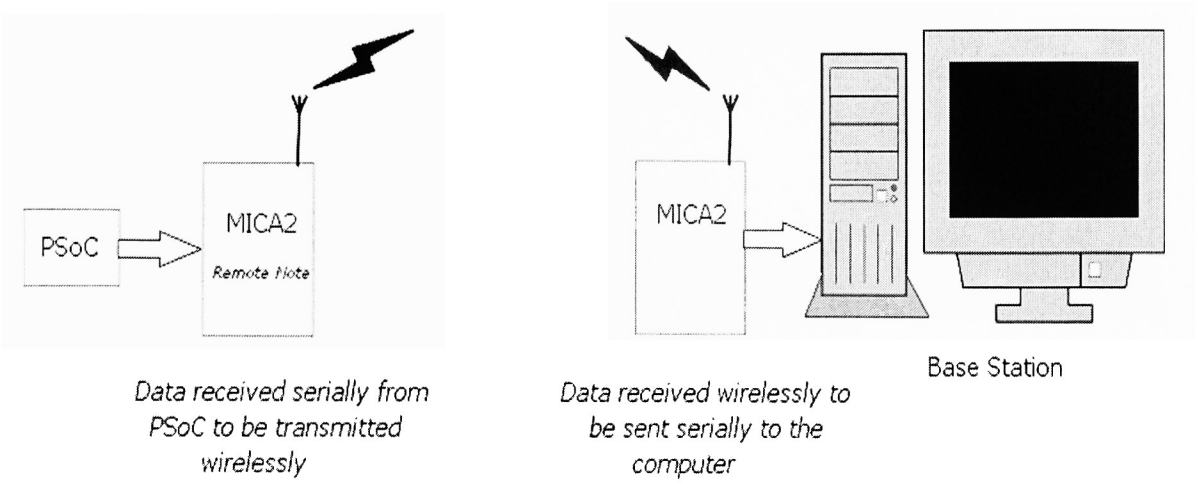


Figure 5.12: Block Diagram of the implemented system

The Remote mote must be programmed such that it is able to receive the data serially from the PSoC and transmit this data wirelessly to the base mote. On the other hand, the base mote must be programmed such that it is able to receive data being transmitted by the base mote and transmit the same to the computer.

Such an application has already been implemented by the TinyOS researchers and is available to TinyOS users. This application code is called TOSBase. The application bridges the serial and wireless channel – whenever it receives a packet serially through UART it transmits it wirelessly to the other mote or when it receives a radio packet wirelessly it transmits it serially, in most applications to a computer at the base station. When a packet is received serially the red led on the mote blinks and when a packet is received wirelessly a green led blinks. Hence, at the remote mote we notice the red led blinking whereas at the base mote we notice the green led blinking. The wiring diagram of the component TOSBase is shown in figure 5.13. It wires together components UART

and RadioCRCPacket which control the low-level UART hardware and CC1000 transceiver respectively. Details on each of the components are found in TinyOS documentation.

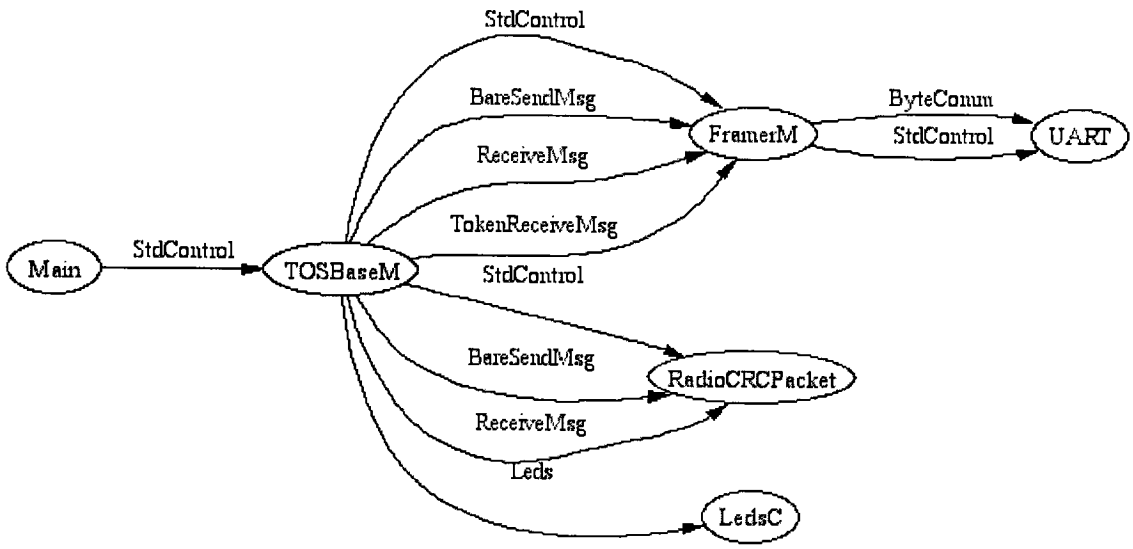


Figure 5.13: shows the wiring diagram of component TOSBase. [25]

5.5 Setting up the Base Station:

At the base station to which the base mote is connected serially, the computer must continuously listen to the serial port for the received packets. It must be able to extract the sampled data from the packets.

TinyOS developers have provided with a java based program of SerialForwarder[35] that listens to the specified serial port at a user-defined baud rate. The SerialForwarder forwards the packets received, onto the internet for connection to with other computers, running the same program.

Another java based program Oscope [35] is provided to the users. This program is run in conjunction with the SerialForwarder. It retrieves the packets from the SerialForwarder and is able to correctly decipher the TOS Message structure. It then displays the sampled data in a graphical display.

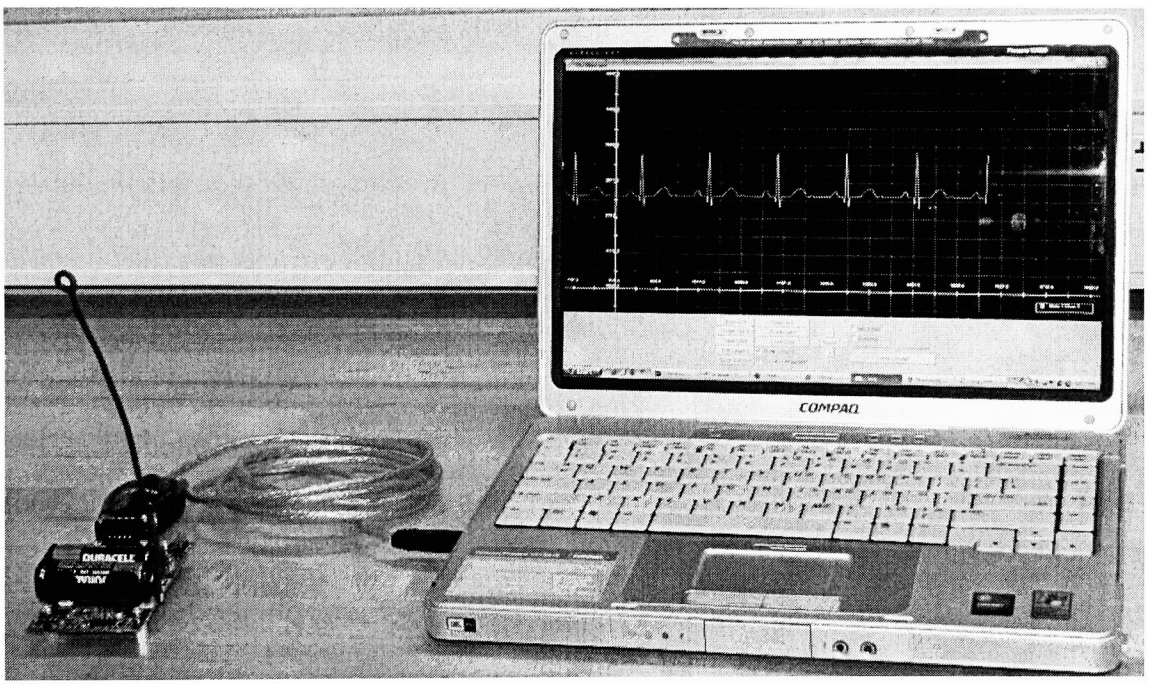


Figure 5.14: Base Station set-up

In our base-station setup we program the SerialForwarder to listen the serial port at a baud rate of 57600, since that is the serial baud rate of MICA2. The Oscope program is run along with it as shown in figure 5.14.

6. Thresholding and Physiological Application

After having discussed the interface design in the last chapter, we discuss the application and usefulness of a PSoC interfaced MICA2 in this chapter. Two applications have been implemented - a thresholding principle and a cardiac monitoring system.

6.1 Thresholding:

Thresholding is used to reduce the transmission data by suppressing the data below a predetermined threshold value. In the implemented design, 10 samples are saved in an array. These samples are then compared to the threshold value, if all the samples are below the threshold; their average is computed and added to the data payload. Once the entire payload is filled, the packet is ready to be sent. However, if any sample in the array is above the threshold the entire array of all 10 samples are sent immediately to the base station. It does not require programming of extra peripheral blocks. Such a system, where the transmission rate is controlled by the value of the element being monitored, is useful in the implementation of an auto-controlled indoor environment where a quick action is required to be taken to bring the sensed element back below the threshold.

The Sample Rate of the implemented system is given by the following equation 6.1:

$$SampleRate = \frac{DataClock}{2^{Bits+2} + CalcTime} \quad ; \textit{Equation 6.1}$$

where, $DataClock = 2.4MHz$

$Bits = 11\ bits$

$CalcTime = 114$

Hence, the SampleRate is about 289 Hz. While above the threshold, at least 28 packets require to be transmitted per second, because each packet contains 10 samples; however below the threshold only 3 packets are required to be sent per second, because for every 10 samples, one average value is computed; therefore a packet containing 10 average values carries the information of 100 samples below the threshold. Such a system reduces the number of bits transmitted by a good amount when the samples are below the threshold. Figure 6.1 shows the samples displayed in the Oscope GUI. Figure 6.1(a) shows the results before implementation and figure 6.1(b) shows the results after implementation of the thresholding principle. It is evident from the illustration that the packets sent below the threshold are reduced by a considerable amount.

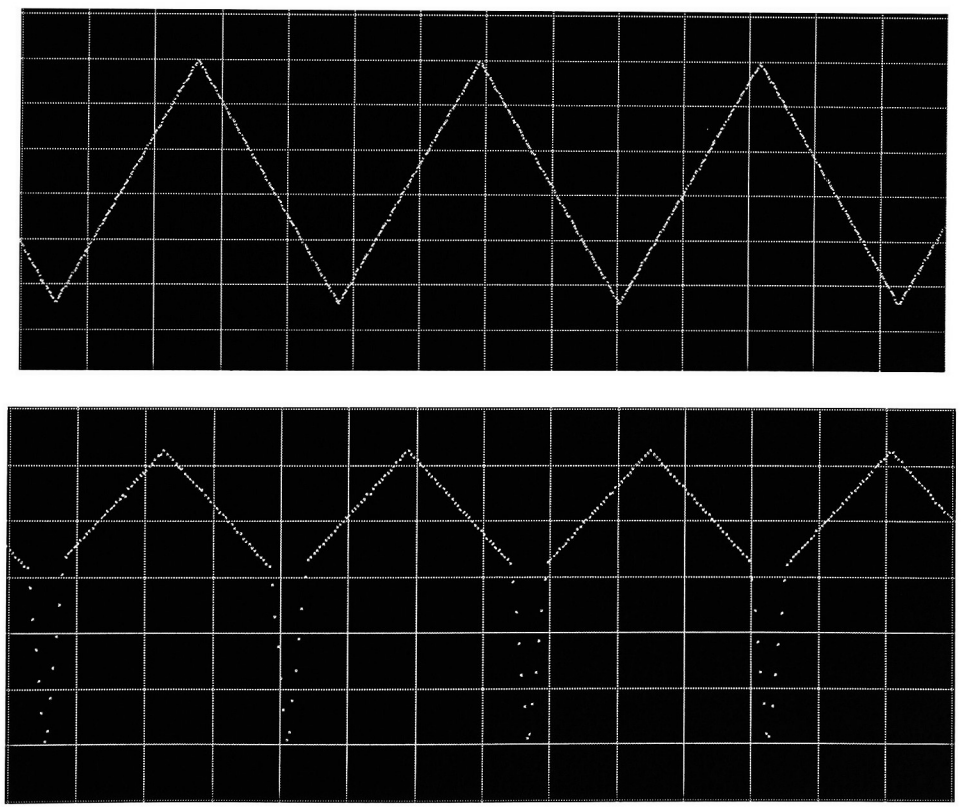


Figure 6.1: Received waveform data displayed at base station using Oscope GUI. (a), above: before thresholding (b), below: after implementing thresholding. Notice the number of samples below the threshold have reduced by 10%

6.2 Cardiac Monitoring System:

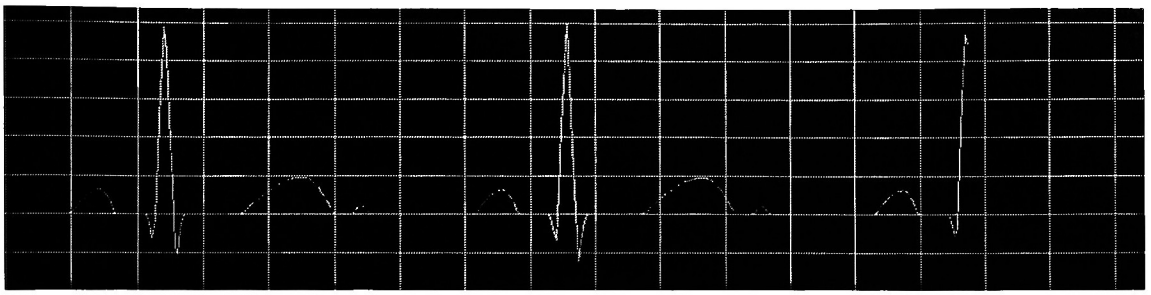


Figure 6.2: Sampled cardiac waveform being displayed at the base station.

6.2.1 Concept of monitoring the cardiac signal within the PSoC:

Incase of a wireless cardiac monitoring system, the sampled cardiac signal would need to be transmitted to the base station at all time at a minimum rate of 10 packets in 1 second (for a sampling rate of about 100Hz). If there is more than 1 patient monitored at the same time then it would result in network congestion and the average reception ratio would begin to drop as shown in figure 2.2 and the cardiac signal would appear distorted at the base station making it difficult to monitor the signal precisely.

Hence, a cardiac monitoring system was conceptualized using the PSoC for continuous monitoring of either a skipped heart beat or for a drop in the HRV. Only after the occurrence of either event is the sampled cardiac signal transmitted to the base station to be monitored and recorded. Such a system makes use of the analog front end capabilities of the PSoC along with its advanced low-power computational capabilities. Implementation of such a system would not have been possible without the interfacing of PSoC to MICA2.

The HRV of a cardiac signal is measured by measuring the standard deviation of the RR-Interval as described in section 2.4 As shown in figure 2.5, the RR-Intervals vary periodically and therefore its standard deviation must not be too low. If the standard deviation reduces, then it is used as a marker for unhealthy activity or a possible indication of an impending heart attack. The formula for calculating the standard deviation is given in equation 6.2.

$$\text{Standard Deviation} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^2} \qquad ; \textit{Equation 6.2}$$

where, n : Number of samples

x_i : Value of each sample (or RR-Interval)

$\overline{x}$: Average of all the samples (i.e. average of all the RR-Intervals)

6.2.2 Measuring the RR-Interval:

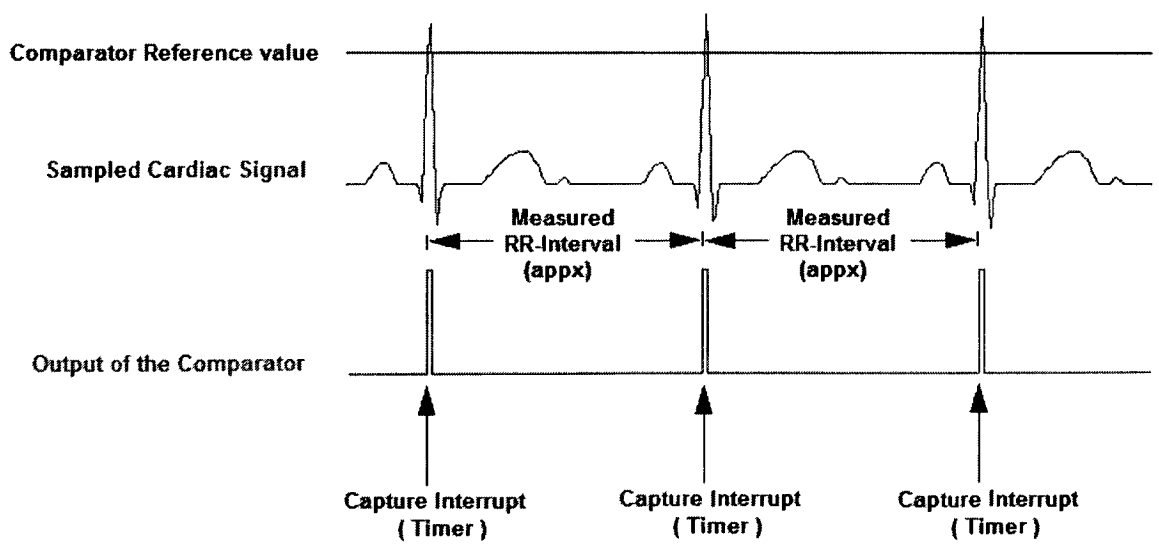


Figure 6.3: Measuring the RR-Interval using PSoC

In our design, the R-wave is detected using an analog comparator. The reference value of the comparator is set such that it detects the positive edge of the R-wave (figure 6.3). The output of the comparator is fed to the capture input of a 16-bit timer. The timer is set to a Capture Interrupt, which copies the value of the count-register to the compare-register of the timer when it encounters a positive edge at the capture input and it also interrupts the microcontroller to read it. The timer continues to run, hence when the next capture interrupt occurs, the user only needs to subtract the present captured value from the previous one to measure the elapsed time between the two interrupts, which is the RR-Interval. For implementing such a system, peripheral blocks such as an analog comparator and a 16 bit timer need to be programmed. This will be described in the following sections.

6.2.3 Measuring a Skipped beat:

For detecting a skipped heart beat, a counter is used which resets itself each time an R-wave is occurs. If an R-wave does not occur for a interval of 1.92 seconds, a variable is flagged, to indicate a skipped heart beat and the sampled cardiac signal is immediately transmitted to the base station for recording/ monitoring.

However, to measure an interval of 1.92 seconds, using the available resources a different approach is utilized. Only 1 digital block is available after programming the other components, therefore an 8-bit counter is programmed. The counter is set to interrupt approximately every 3.88ms, using a clock of 32.967 KHz (derived from the system clock). A timeout variable is utilized to measure the interval of 1.92 seconds. It is decremented every 3.88msec in the counter-ISR from an initial value of 500. When the

timeout variable reaches zero, about 1.92 seconds would've elapsed. The timeout variable is set back to 500 within the GPIO ISR. The GPIO interrupt occurs during the positive edge of the comparator output (figure 6.3). The GPIO interrupt is generated by routing the comparator output to one of the PSoC pins and enabling an (positive) edge triggered interrupt at the pin. This way the timeout variable will reach zero only when an R-wave fails to occur for a period of 1.92 seconds. Programming of the counter and the GPIO interrupt will be discussed in the following sections.

6.2.4 Placing and Routing in the Device Editor:

To implement the described cardiac monitor, an analog comparator, a 16-bit timer and an 8-bit counter are placed as shown in figure 6.4. The output of the PGA (dark-green) is routed to the input of the comparator (purple), which detects the positive edge of the R-wave. The output of the analog comparator is routed to the capture input of the 16 bit timer (light blue) and the GPIO Port 0[6]. The GPIO interrupt is enabled for a positive edge on Port 0[6]. An 8-bit counter (light-green) is placed in the last available digital peripheral block.

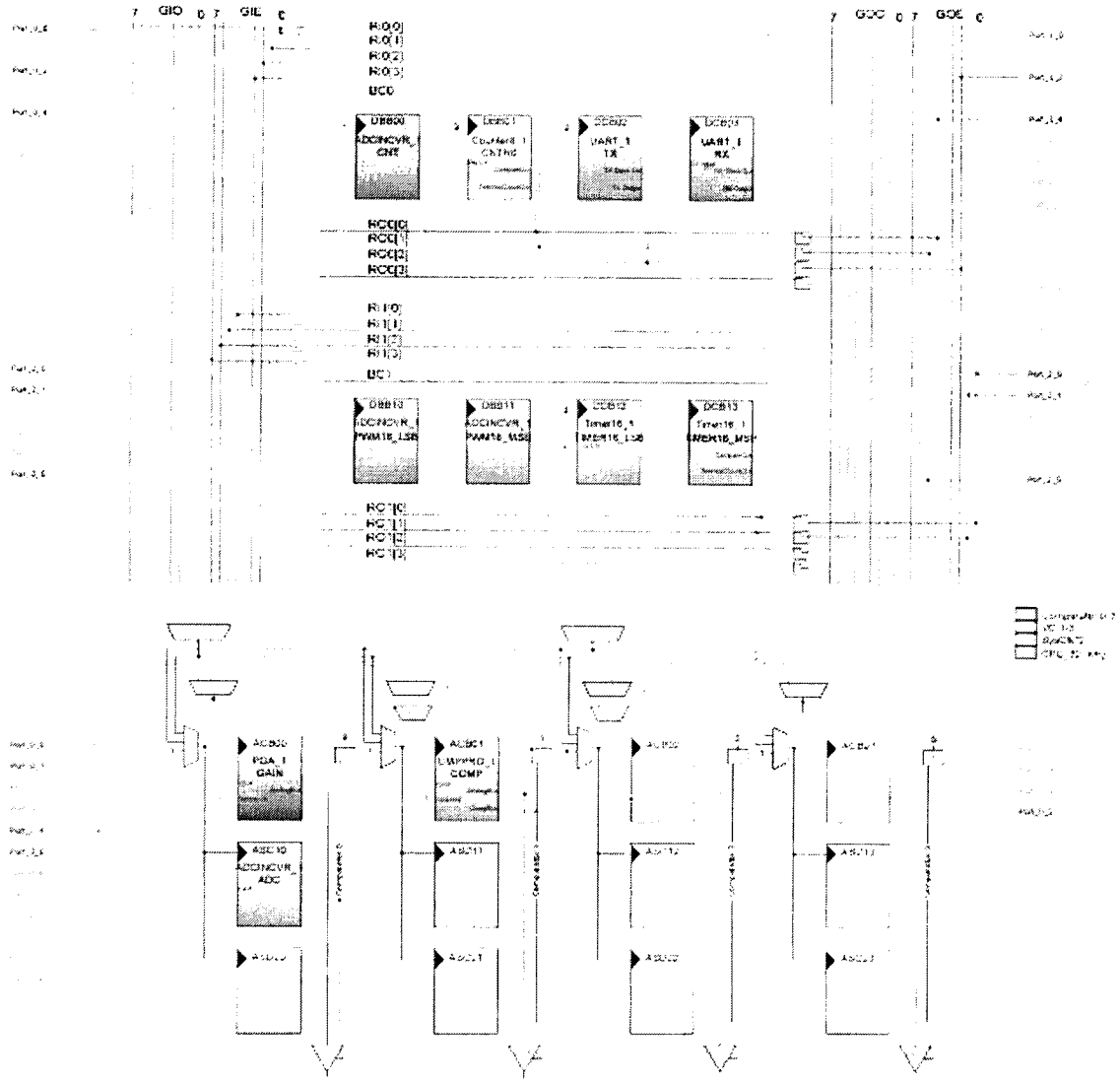


Figure 6.4: Placing of peripheral blocks and routing of signals in Device Editor for a cardiac monitoring system.

Both, the timer and the counter are clocked by a 32.967 KHz clock, which is derived from the system clock using in-built clock dividers. The period of each clock is approximately 30usec. Therefore to set the counter to interrupt every 3.88msec, a count of 127 (decimal) is loaded in the count register and terminal count interrupt is enabled. The count register is decremented at every positive edge of the clock and the

microcontroller is interrupted to jump to the Counter ISR when it reaches its terminal count of 0.

The timer is initialized with a maximum count of 0xFFFF (65535, decimal) in the count register. It is decremented with each clock. Since the timer interrupt is set to a capture interrupt, care should be taken that the capture input is asserted once every 65535 clocks (i.e. within 1.96 seconds); otherwise there will be an overflow when subtracting with the previously captured value of the timer-count-register. This problem is overcome using the counter, which gives indications incase an R-wave does not occur for 1.92 seconds.

6.2.5 Application Editor:

In the AE, each of the peripheral components must to be initialized and the ISR for the counter, timer and GPIO interrupts need to be defined. For the calculation of standard deviation, the measured values of RR-Interval are stored in an array within the timer ISR. After 10 values of RR-Interval are saved in the array, their standard deviation is calculated using equation 6.2. In our design, since the cardiac signal is being generated by a function generator it has a constant RR-Interval instead of having periodic variations. Therefore an event detection was made possible, to check for a reduced SD, which might not have been possible if an ECK sensor was interfaced, because it would have required monitoring of the cardiac signal of a person with a cardiac problem. However, the downside is that we could not test the system for natural periodic variations of RR-Interval.

6.2.6 For Measuring Standard Deviation over a period of 5 Minutes:

In order to analyze the HRV, a short term interval of 5 minutes or long interval of 24 hours is recommended for the calculation of SD of RR-Intervals. In paper [39] different time intervals for measuring HRV has been analyzed. For measuring the SD over an interval of 5 minutes, it would require enough memory to store at least 300 values (60 beats per minutes is the heart rate of a trained athlete) of RR-Interval so as to subtract individual RR-Intervals from the calculated average. The PSoC family used in our design has only 256 bytes of RAM therefore it would require a different family of PSoC which has more RAM space provided. Otherwise a different formula of standard deviation can also be used, which allows us to compute the SD on the fly. The short-cut formula of SD (equation 6.3), is derived below, from the SD formula in equation 6.2.

$$SD = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i-\bar{x}\right)^2} \qquad \textit{; From Equation 6.2}$$

$$\Rightarrow \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i{}^2+\bar{x}^2-2x_i\bar{x}\right)} \qquad \text{; Expanding the square term in Equation 6.2}$$

$$\Rightarrow \sqrt{\frac{1}{n-1}\left[\sum_{i=1}^{n}x_i{}^2+\sum_{i=1}^{n}\bar{x}^2-2\,\bar{x}\sum_{i=1}^{n}x_i\right]} \qquad \text{; Taking summation of individual terms}$$

$$\Rightarrow \sqrt{\frac{1}{n-1}\left[\sum_{i=1}^{n}x_i{}^2+n\bar{x}^2-2\,\bar{x}\sum_{i=1}^{n}x_i\right]} \qquad ;\because \sum_{i=1}^{n}\bar{x}^2 = n\bar{x}^2$$

$$\Rightarrow \sqrt{\frac{1}{n-1}\left[\sum_{i=1}^{n}x_i{}^2+n\bar{x}^2-2\,n\,\bar{x}^2\right]} \qquad ;\because \sum_{i=1}^{n}x_i = n\bar{x}$$

$$\Rightarrow \sqrt{\frac{1}{n-1}\left[\sum_{i=1}^{n}x_i{}^2-n\,\bar{x}^2\right]} \qquad \text{; Combining the last 2 terms in the bracket}$$

$$\Rightarrow \sqrt{\frac{1}{n-1}\left[\sum_{i=1}^{n} x_i^{\,2} - n\left[\frac{\sum_{i=1}^{n} x_i}{n}\right]^2\right]} \qquad ; \because \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\Rightarrow \sqrt{\frac{1}{n(n-1)}\left[n\sum_{i=1}^{n} x_i^{\,2} - \sum_{i=1}^{n} x_i^{\,2}\right]} \quad ; \text{Making } n \text{ in denominator common}$$

SD $$\Rightarrow \sqrt{\frac{n\sum_{i=1}^{n}(x_i^{\,2}) - (\sum_{i=1}^{n} x_i)^2}{n(n-1)}} \qquad ; \textit{Equation 6.3}$$

where, n : Number of samples

x_i : Value of each sample (or RR-Interval)

From equation 6.3, we can calculate the standard deviation on the fly for an interval of 5 minutes or more because it only requires the current value of the RR-Interval and does not require us to subtract it from individual RR-Intervals after computing the average value of all RR-Intervals.

7. Results and Performance Analysis

Cypress MicroSystems has made a Power calculator available to the PSoC users, which helps developers to estimate the current and power consumption in the chip based on the project. It is dependant on the number of peripheral blocks used, frequency of clock used for each of them, the number of pins and rows (on which the signals are routed) to be driven etc. Below is the estimated current and power consumption for the thresholding and the cardiac monitoring system.

7.1 Thresholding Application:

Estimated PSoC Power Calculations for CY8C27xxx Family Device

Version 1.0

project name	Thresholding
part number	CY8C27443
pin count	28

Cypress MicroSystems, Inc.

Enter parameters in yellow
See estimated result in green

CPU Clock	3.000	MHz
Vdd	3.30	V
CPU Current	2.049	mA

Digital Block	Block Clk (MHz)			GPIO Count	GPIO Clk (MHz)		
DBB00	2.400	0.029	mA	0	0.000	0.000	mA
DBB01	0.000	0.000	mA	0	0.000	0.000	mA
DCB02	0.461	0.009	mA	1	0.461	0.021	mA
DCB03	0.461	0.009	mA	0	0.000	0.000	mA
DBB10	2.400	0.029	mA	0	0.000	0.000	mA
DBB11	2.400	0.029	mA	0	0.000	0.000	mA
DCB12	0.000	0.000	mA	0	0.000	0.000	mA
DCB13	0.000	0.000	mA	0	0.000	0.000	mA
Digital Block Current		0.107	mA	GPIO Current		0.021	mA

Analog Block	Power Level (0-3)		
ACB00	2	0.600	mA
ACB01	2	0.600	mA
ACB02	0	0.000	mA
ACB03	0	0.000	mA
ASC10	0	0.000	mA
ASD11	0	0.000	mA
ASC12	0	0.000	mA
ASD13	0	0.000	mA
ASD20	0	0.000	mA
ASC21	0	0.000	mA
ASD22	0	0.000	mA
ASC23	0	0.000	mA
Analog Block Current		1.200	mA

OpAmp Bias	0	
A_Buffer Power	0	

Row	Row Clk (MHz)			Analog Power			
Row_0_Output_0	0.000	0.000	mA	SC Off Ref Low	0	0.000	mA
Row_0_Output_1	0.000	0.000	mA	SC Off Ref Med	0	0.000	mA
Row_0_Output_2	0.461	0.014	mA	SC Off Ref High	0	0.000	mA
Row_0_Output_3	0.000	0.000	mA	SC On Ref Low	1	0.385	mA
Row_1_Output_0	0.000	0.000	mA	SC On Ref Med	0	0.000	mA
Row_1_Output_1	0.000	0.000	mA	SC On Ref High	0	0.000	mA
Row_1_Output_2	0.000	0.000	mA	Reference Circuit Current		0.385	mA
Row_1_Output_3	0.000	0.000	mA				
Row Current		0.014	mA				

Analog Output Buffer			
AnalogOutBuf_0	0	0.000	mA
AnalogOutBuf_1	0	0.000	mA
AnalogOutBuf_2	0	0.000	mA
AnalogOutBuf_3	0	0.000	mA
Analog Output Buffer Current		0.000	mA

Estimated Total PSoC Current	3.774	mA
Estimated Total PSoC Power	12.456	mW

Table 7.1: Power consumption in PSoC for implementation of the thresholding algorithm

In the implementation of the thresholding principle; with

- a sampling rate of 289 Hz, a resolution of 11 bits
- a CPU clock of 3 MHz,
- power supply of 3.3V,
- a low reference voltage on the SC block (ADC),
- and one row being driven for serial transmission at 0.461 MHz (8 times the Baud rate);

the estimated current consumption is about 3.774mA (from table 7.1).

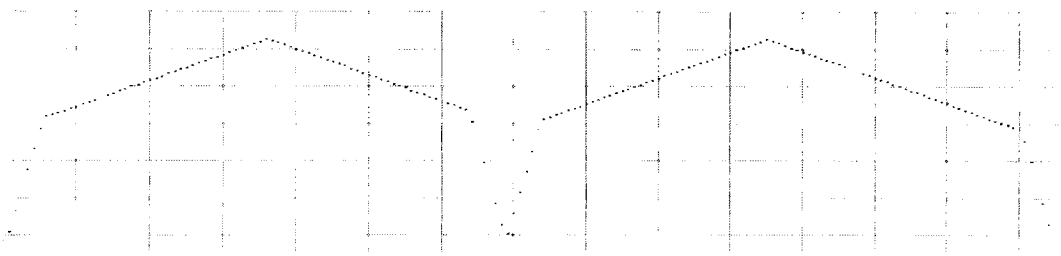


Figure 7.1: Displayed waveform at the base station after the implementation of thresholding. Note that the transmission rate is reduced when signal is below the threshold and increased when signal is above the threshold.

Table 7.2 (page 65) draws a comparison between the implemented system and the original system. For a sample rate of about 289 Hz, in the original system 28 packets are transmitted in one second. It assumes that the Radio transmits the packets for about 80% of the time within an hour and is idle for 20% the rest of the time. Model 1 is implementation without the PSoC and Model 2 is the implementation with PSoC.

In Model 2 (with the PSoC) when the signal is below the threshold, an average of 10 samples is sent. Therefore there is a reduction of 10% of packets while the signal is below the threshold. This can be observed in figure 7.1.

These models do not take into consideration the computational load on MICA2. The computational load on MICA2 would be more in Model 1 than Model 2, which would further demonstrate that the PSoC interfaced design saves power. Table 7.2 gives an overview of how PSoC serves as a power-saver by reducing the transmission rate, in turn reducing the transmission power.

In table 7.2, CASE1 represents a situation where the signal is always below the threshold, it has been assumed that the radio transmits packets for 80% of the time within an hour and is idle for 20% of the time in the original system (Model 1). Because the signal is always below the threshold, the packets would be reduced by 10% in a PSoC interfaced MICA2 (Model 2). Therefore the radio transmits only for 8% of the time and is idle for 92% of the time. The consumption reduces by approximately 50%.

In CASE2, the signal is alternating and goes below the threshold for half the time. In the original system (Model 1), the signal would be transmitted at all times, whereas in the Model 2, the packets are transmitted for only 44% of the time. This results in approximately 6% of reduced consumption.

In CASE3, where the signal is always above the threshold, the PSoC interfaced MICA2 would have to take a hit because of the current consumption of the PSoC. However, the threshold should be decided such that the signal is not always above the threshold.

Such a system could be used in auto-controlled indoor environment.

CASE I 100% of the time below the threshold

		Model 1		Model 2	
		% time within an hour	mA-hr used within an hour	% time within an hour	mA-hr used within an hour
Radio					
Current xmit	12 mA	80%	9.6	8%	0.96
Current Sleep	2uA	20%	0.0004	92%	0.0018
PSoC					
Active current	3.774mA	0%	0	100%	3.774
Computed mA-hr used each hour			9.6004		4.7358
Consumption reduced by			50.67%		

CASE 2: 50% of the time below the threshold

		Model 1		Model 2	
Radio					
Current xmit	12 mA	80%	9.6	44%	5.28
Current Sleep	2uA	20%	0.0004	56%	0.00112
PSoC					
Active current	3.774mA	0%		100%	3.774
Computed mA-hr used each hour			9.6004		9.05512
Consumption reduced by			5.68%		

CASE 3 0% of the time below the threshold

		Model 1		Model 2	
Radio					
Current xmit	12 mA	80%	9.6	80%	9.6
Current Sleep	2uA	20%	0.0004	20%	0.0004
PSoC					
Active current	3.774mA	0%	0	100%	3.774
Computed mA-hr used each hour			9.6004		13.3744
Consumption increased by			39.31%		

Table 7.2: Evaluation of power consumption with and without PSoC interface. Model 1: without PSoC and Model2: with PSoC.

7.2 Cardiac Monitoring System:

In order to test the SD calculated for the RR-interval of 10 samples, the debugger is used along with the fast running emulator. The results are checked in the watch windows shown in figure 7.2.

Watch/Global Name	
iRR_Interval10	0x8180, 0x8180, 0x8181, 0x8181, 0x8180, 0x8182, 0x8181, 0x8183, 0x8181, 0x8182
iSample	0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000
k	0x0a
mean	33153.097656
std_dev	0.943401
total	0x00050f0b
variance	0.890005
variance_tot	8.900055
wait	0x2000

Figure 7.2: Watch variables window in the debugger running the cardiac monitoring system.

Note that in figure 7.2, the 10 samples of RR-Interval are all around 0x8181 (33153, decimal. Since the clock to the timer has a period of 30usec, the measured RR-Interval is about 33153*30usec = 0.9945 seconds, which is very close to the applied 1Hz cardiac signal from the function generator. This proves that the RR-Intervals were being measured correctly. The system was also checked for a varying RR-Interval. However, the variation in the frequency was done manually and therefore was not the same as a natural variation in RR-Interval. The system was checked for a varying frequency and it worked as per the requirements.

To test the system for a skipped beat, the frequency of the signal was reduced below 0.5Hz and the base station was observed, it was noticed that the remote mote immediately sent out the sampled cardiac signal to the base station, because it had not detected an R-

wave for an interval of about 2 seconds (or an overflow of the timers terminal count interrupt.). Therefore the system worked correctly for skipped beat detection.

Initially the timer interrupt did not read the RR-Intervals correctly. This is because of bouncing at the output of the comparator (at the capture input of the timer). The bouncing of the signal was causing the capture interrupt to occur at every positive edge and record incorrect values of RR-interval. Hence for de-bouncing, a wait variable (in figure 7.2) is used for applying a short delay after the first positive edge at the output of the comparator and the impending interrupts are cleared.

The power consumption of the PSoC in a cardiac monitoring system is calculated using the power calculator (table 7.3). For a system with:

- a sampling rate of approximately 74 Hz, with a resolution of 13bits
- a CPU clock of 12 MHz,
- power supply of 3.3V,
- a high reference voltage for the analog blocks
- and one row being driven for serial transmission at 0.461 MHz (8 times the Baud rate);

the estimated current consumption is about 10.939mA.

Estimated PSoC Power Calculations for CY8C27xxx Family Device

Version 1.0

Enter parameters in yellow
See estimated result in green

project name	Cardiac_Monitoring_Sys	Cypress MicroSystems, Inc.
part number	CY8C27443	
pin count	28	

CPU Clock	12.000	MHz
Vdd	3.30	V
CPU Current	2.764	mA

Digital Block	Block Clk (MHz)		
DBB00	2.400	0.029	mA
DBB01	0.032	0.005	mA
DCB02	0.461	0.009	mA
DCB03	0.461	0.009	mA
DBB10	2.400	0.029	mA
DBB11	2.400	0.029	mA
DCB12	0.032	0.005	mA
DCB13	0.032	0.005	mA
Digital Block Current		0.122	mA

	GPIO Count	GPIO Clk (MHz)	
	0	0.000	0.000 mA
	0	0.000	0.000 mA
	1	0.461	0.021 mA
	0	0.000	0.000 mA
	0	0.000	0.000 mA
	0	0.000	0.000 mA
	0	0.000	0.000 mA
	0	0.000	0.000 mA
GPIO Current			0.021 mA

Row	Row Clk (MHz)		
Row_0_Output_0	0.000	0.000	mA
Row_0_Output_1	0.000	0.000	mA
Row_0_Output_2	0.461	0.014	mA
Row_0_Output_3	0.000	0.000	mA
Row_1_Output_0	0.000	0.000	mA
Row_1_Output_1	0.000	0.000	mA
Row_1_Output_2	0.000	0.000	mA
Row_1_Output_3	0.000	0.000	mA
Row Current		0.014	mA

Analog Power		
SC Off/Ref Low	0	0.000 mA
SC Off/Ref Med	0	0.000 mA
SC Off/Ref High	0	0.000 mA
SC On/Ref Low	0	0.000 mA
SC On/Ref Med	0	0.000 mA
SC On/Ref High	1	5.419 mA
Reference Circuit Current		5.419 mA

OpAmp Bias	0	

Analog Block	Power Level (0-3)		
ACB00	2	0.600	mA
ACB01	2	0.600	mA
ACB02	0	0.000	mA
ACB03	2	0.600	mA
ASC10	0	0.000	mA
ASD11	0	0.000	mA
ASC12	0	0.000	mA
ASD13	0	0.000	mA
ASD20	0	0.000	mA
ASC21	0	0.000	mA
ASD22	0	0.000	mA
ASC23	0	0.000	mA
Analog Block Current		1.800	mA

A_Buffer Power	0	

Analog Output Buffer			
AnalogOutBuf_0	0	0.000	mA
AnalogOutBuf_1	1	0.800	mA
AnalogOutBuf_2	0	0.000	mA
AnalogOutBuf_3	0	0.000	mA
Analog Output Buffer Current		0.800	mA

Estimated Total PSoC Current	10.939	mA
Estimated Total PSoC Power	36.097	mW

Table 7.3: Power consumption in PSoC of the cardiac monitoring system.

If the sampling frequency is increased beyond 74 Hz, which is about 7 to 8 packets in 1 second, the packets start dropping due to which the display or recording of cardiac signal at the base station gets affected. This is shown in figure 7.3

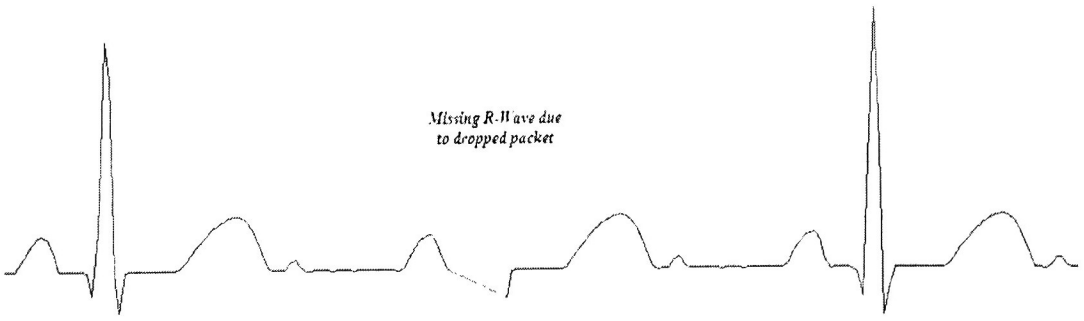


Figure 7.3: Distorted waveform received at the base station due to dropped packet.

Therefore the sampling frequency must be kept within 100Hz to keep the signal from being distorted at the base station.

Since the signal is transmitted only when there is a reduced HRV or a skipped heart beat, there are fewer patients transmitting the signal and therefore the network traffic is reduced drastically.

8. Energy Model for Single-Hop, Single-Sender Network.

This chapter models the energy consumption in the implemented single-hop, single-sender sensor network. Model 1 shows the energy consumption without the interfacing of PSoC at the remote node, whereas model 2 shows the energy consumption with the interface of PSoC at the remote node. It demonstrates how interfacing of PSoC can help reduce transmit power at the remote node and receive power at the base node.

8.1 Energy Model of Individual Components:

8.1.1 Sensor Energy Consumption; E(S)

As discussed in [45], we assume that the energy consumed by the sensor per bit is a constant, $E_{sensor/bit}$. Therefore the total energy consumed by the sensor in sensing r bits would be equal to E(S) [46]:

$$E(S) = E_{sensor/bit} * r \qquad \textit{-Equation 8.1}$$

8.1.2 Microcontroller Energy Consumption; E(MCU)

Because the network is assumed to be a single-hope, single-sender network, only the transfer of data from the remote node to the base node is considered. In the remote node, the central microcontroller would essentially be used to transfer readings from the sensor to the transceiver as shown in figure 8.2. Let N_{op} be the number of operations performed by the microcontroller to complete the transfer of one bit from the sensor to the transceiver. Assuming that $E_{mcu/op}$ is the mean energy consumed within the MCU to perform a single operation, the energy consumption in the MCU would be given as:

$$E(MCU) = E_{mcu/op} * N_{op} * r \qquad \text{-Equation 8.2}$$

$E_{mcu/op}$ will be a function of applied voltage, frequency of operation and internal capacitance of the processor's logic. r indicates the number of bits sensed by the sensing units that are to be transmitted to the base station.

8.1.3 Transceiver Energy Consumption:

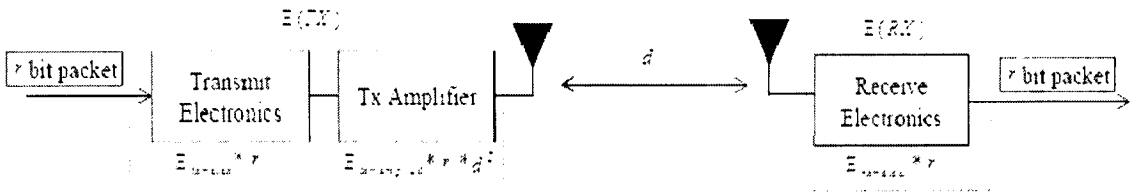


Figure 8.1: First Order Radio Model. Source [47]

Figure 8.1 shows the transceiver block diagram at both the remote node, which is in the transmit mode as well as the base-station node, which is in the receive mode. The Radio Model discussed in [45] and [47] is used as the transmitter and receiver's energy model.

Transmit Mode Energy Consumption; $E(TX)$ **:** It is assumed that $E_{tx-elex/bit}$ is the energy consumed per bit by the transmitter electronics and $E_{tx-amp/bit}$ is the energy consumed per bit by the transmit op-amp. We consider a first order system as the one shown in figure 8.1. It has only one transmitter and one receiver, separated by a distance d. The path loss in such a system would be d^2. Therefore the energy consumption of the transmitter would be given as [45][47]:

$$E(TX) = (E_{tx-elex/bit} * r) + (E_{tx-amp/bit} * r * d^2) \qquad \text{- Equation 8.3}$$

Receive Mode Energy Consumption; $E(RX)$ **:** It is assumed that the transceiver electronics consumes a constant, $E_{rx-elex/bit}$ energy per bit in the receive mode. Therefore energy consumed by a transceiver in a receive mode can be given as [45][47]:

$$E(RX) = E_{rx-elex/bit} * r \qquad \textit{-Equation 8.4}$$

8.2 Model 1: Without interfacing PSoC.

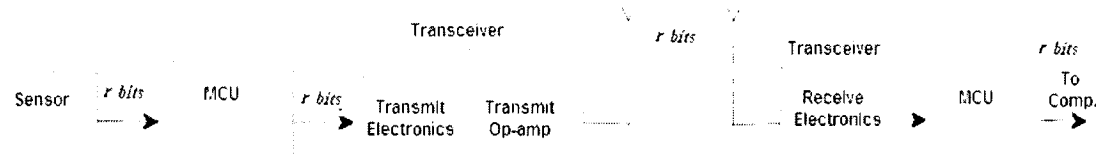


Figure 8.2: Block Diagram of Remote Node and Base Node in a single-hop, single node n/w

The total energy consumption in a remote node in a single-hop, single-node network would be the summation of the energy consumed by the MCU, Sensor and the Transceiver (in the transmit mode). It can be represented by the following energy model:

$$E(REMOTE) = E(S) + E(MCU) + E(TX) \qquad \textit{-Equation 8.5}$$

$$\Rightarrow E(REMOTE) = (E_{sensor/bit} * r) + (E_{mcu/op} * N_{op} * r) + [(E_{tx-elex/bit} * r) + (E_{tx-amp/bit} * r * d^2)] \qquad \textit{-Equation 8.6}$$

$$\Rightarrow E(REMOTE) \propto r \qquad \textit{-Equation 8.7}$$

Equation 8.6 is formed by substituting Equation 8.1, 8.2 and 8.3.

The energy consumed at the base node, would be the summation of the transceiver in the receive mode and the MCU computations, which would essentially transfer the incoming data from the transceiver to the computer at the base station.

Therefore the total energy model for the node at the base station in a single-hop, single node network can be represented as:

$$E(BASE) = E(RX) + E(MCU) \qquad \text{-Equation 8.8}$$

$$\Rightarrow E(BASE) = (E_{rx-elex/bit} * r) + (E_{mcu/op} * N_{op} * r) \qquad \text{-Equation 8.9}$$

$$\Rightarrow E(BASE) \propto r \qquad \text{-Equation 8.10}$$

Equation 8.9 is formed by substituting equation 8.2 and 8.4.

It can be deduced that the total energy consumption of the remote node as well as the base node depends on the number of bits sensed by the sensor and transmitted to the base station. By trying to limit the number of bits transmitted to the base station the total energy consumption can be reduced. Based on this premise a means for low-power event detection was found such that only important information is transmitted to the base station instead of all the sensed data. This is discussed on Model 2.

8.3 Model 2: With interfaced PSoC

An external component PSoC is required in addition to the MCU at the remote node, in order to perform complex computations. PSoC as discussed in Chapter 4 helps perform complex event detection. Figure 8.3 shows the block diagram of PSoC interfaced MICA2 remote node.

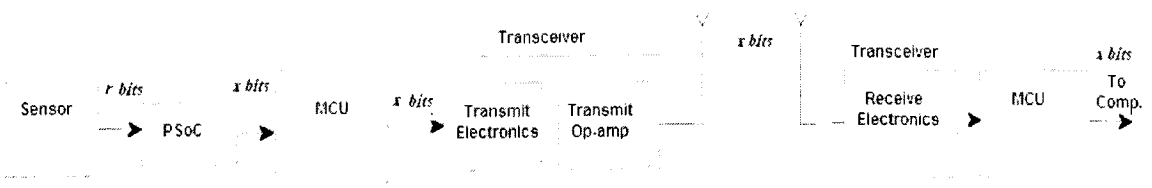


Figure 8.3: Block diagram of the Remote Node with PSoC along with the Base Node.

8.3.1 PSoC Energy Consumption; E(*PSOC*)

Let r be the bits sensed by the sensor. These bits are fed to the PSoC for performing event detection. Let x be the number of bits that are to be transmitted to the base station after being processed by the PSoC. x will be a function of the probability P(*event*), at which an 'event occurs' and/or the number of bits to be transmitted to the base station after periodic time interval t.

$$x = f(\mathrm{P}(event), N_t) \qquad \textit{-Equation 8.11}$$

where N_t is the number of bits to be transmitted to the base station after a periodic time interval t. It can be decided by the programmer and depends on the application for which the sensor network is used. Then the efficiency of the PSoC can then be computed as

$$\eta = \frac{r - x}{r} \qquad \textit{-Equation 8.12}$$

The efficiency of the PSoC indicates the energy saved by interfacing the PSoC. It is a ratio of the number of bits not transmitted (to the base station) to the total number of bit sensed by the sensor. Without the interfacing of PSoC all r bits sensed by the sensor would be transmitted to be base station. However, in model 2 only x bits will be required to be transmitted. The efficiency of the PSoC will range from zero to one. Zero being the worst case, whereby it transmits all the bits it receives (i.e. $x = r$) and one being the best case wherein it does not transmit any bits (i.e. $x = 0$) incase there is no occurrence of an event. Efficiency would typically be a fraction between one and zero.

It is assumed that $\mathrm{E}_{psoc-comp/bit}$ is the energy consumed by PSoC for performing computations on a single bit received from the sensor. It depends on the number of

analog and digital blocks configured, frequency of operation, supply voltage and capacitance. It is also assumed that a constant $E_{psoc-tx/bit}$ is the energy consumed per bit to transmit the bits to the MCU, incase of event detection and/or periodic sends. Let r be the bits sensed by the sensor and x be the number of bits transmitted by the PSoC. Then the energy consumption of PSoC can be represented as:

$$E(PSOC) = (E_{psoc-comp/bit} * r) + (E_{psoc-tx/bit} * x) \qquad \textit{-Equation 8.13}$$

The energy consumption of the MCU and the transceiver will now depend on x bits from the PSoC instead of r bits from the sensor. Therefore the total energy consumption at a remote node is as shown below:

$$E(REMOTE) = E(S) + E(PSOC) + E(MCU) + E(TX) \qquad \textit{-Equation 8.14}$$

$$\Rightarrow E(REMOTE) = (E_{sensor/bit} * r) + [(E_{psoc-comp/bit} * r) + (E_{psoc-tx/bit} * x)] + (E_{mcu/op} * N_{op} * x) + [(E_{tx-elex/bit} * x) + (E_{tx-amp/bit} * x * d^2)]$$

-Equation 8.15

$$\Rightarrow E(REMOTE) = r * [E_{sensor/bit} + E_{psoc-comp/bit}] + x * [E_{psoc-tx/bit} + (E_{mcu/op} * N_{op}) + E_{tx-elex/bit} + (E_{tx-amp/bit} * d^2)]$$

-Equation 8.16

$$\Rightarrow E(REMOTE) = r * \{ [E_{sensor/bit} + E_{psoc-comp/bit}] + (1-\eta) * [E_{psoc-tx/bit} + (E_{mcu/op} * N_{op}) + E_{tx-elex/bit} + (E_{tx-amp/bit} * d^2)] \}$$

-Equation 8.17

Equation 8.15 is formed by substituting equation 8.2 and 8.3 with x instead of r and also substituting equation 8.13 for $E(PSOC)$. In equation 8.17, x has been substituted by the efficiency of PSoC (η) from equation 8.12. The efficiency will always lie between zero and one as mentioned earlier. It can be observed that as the efficiency of the PSoC increases, the total energy consumption of the remote mote will reduce. In an ideal case, the efficiency of PSoC would be one. In such a case, the second half of equation 8.17 will be zero, resulting in a substantial amount of energy saving due to reduction in transmit power as well as MCU computation power.

However, if the efficiency of the PSoC is low, for example zero, then there will be an overhead of the PSoC energy consumption compared to the model 1. Therefore the PSoC should be programmed such that the efficiency is high. This can be achieved with better event detection algorithms so that fewer bits are to be transmitted to the base station.

At the base node, the energy model will look similar to Model 1, except that it will now be dependant on transmitted bits x instead of r (in the first model). Therefore the energy model at the base station is given as:

$$E(BASE) = E(RX) + E(MCU) \qquad \textit{-Equation 8.18}$$

$$\Rightarrow E(BASE) = (E_{rx-elex/bit} * x) + (E_{mcu/op} * N_{op} * x) \qquad \textit{-Equation 8.19}$$

$$\Rightarrow E(BASE) = (1-\eta)r * [E_{rx-elex/bit} + (E_{mcu/op} * N_{op})] \qquad \textit{-Equation 8.20}$$

Equation 8.19 has been formed by substituting equation 8.4 and 8.2 with x instead of r. Equation 8.20 is formed by substituting x with efficiency η from equation 8.12.

The energy consumption of base node like the remote node in model 2 also depends on the efficiency of the PSoC. Higher the efficiency, lower the energy consumption in the transceiver as well as the MCU at the base station.

This thesis implements the Model 2 to verify the energy savings. As seen from the Chapter 6 and 7, the interfacing of PSoC helps perform complex event detection and indeed results in saving number of transmitted bits, thereby saving energy.

9. Future Enhancement

9.1 Power Reduction within the Interface

This section describes a technique that could be explored to further reduce power consumption in the PSoC interfaced MICA2. Gated-clock is an important technique for reducing power consumption in a system-on-chip. The principle is to disable the clock to the logic circuits that are not operational within that time frame. This is possible to implement because individual circuit usage within a system varies depending on the application. Since the power consumption of CMOS circuits is directly proportional to the clock frequency (equation 9.1), this technique helps save considerable amount of power without affecting the performance of the system.

$$P \propto f_c C_L V_{dd}^2 \qquad \textit{Equation 9.1}$$

where,

P = Power consumed

f_c = Frequency of Clock

C_L = Load Capacitance

V_{dd} = Supply Voltage.

In figure 9.1(a), Cg represents the cumulative gate capacitance of the flip flop. The clock signal applied to a flip-flop, charges and discharges Cg in each cycle. This consumes power even if the Data-in and Data-out are constant or unused. In such a case, a gated clock, as seen in figure 9.1(b) can be used. The clock to the flip-flop can be shut off when

it in not in use. The capacitance of the AND-gate is lower than that of the flip-flop and hence the power is saved. [48]

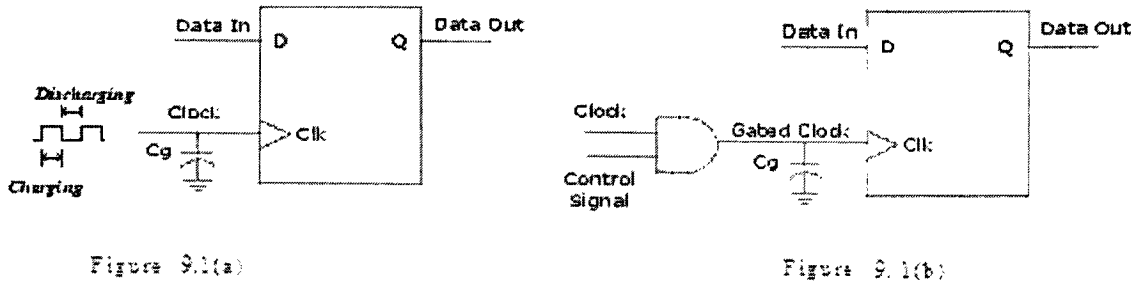


Figure 9.1(a) Figure 9.1(b)

Figure 9.1: (a) Flip-flop without gated clock; (b): Flip-flop with gated clock.

In the applications of PSoC interfaced MICA2, the PSoC performs complex processing and intelligent monitoring of the sensor signal whereas the MICA2 performs the wireless communication and implements the network protocols. The samples are transmitted from PSoC to MICA2 only during the occurrence of certain events. Therefore the UART block remains unused for most of the time. For example, in the cardiac monitoring system, the heart rate variability is sent only once in 5 minutes. Therefore, by turning off the clock to the UART block when there is no transmission from the PSoC to MICA2, would further save power.

In the interface of PSoC to MICA2, a TinyOS Message Structure (section 5.3) is used. According to the TOS Message structure, the samples are transmitted from the PSoC to the UART in structured packets. The entire packet is enclosed within a start and a stop byte, 7E. The central processor in the PSoC could use 7E as an indicator to control the clock to the UART block. Figure 9.2 shows the proposed change to the design in red.

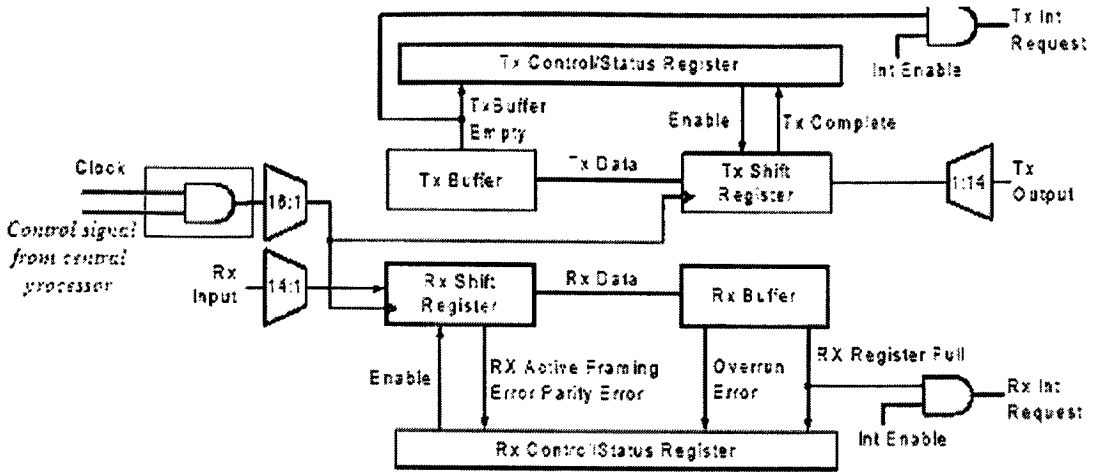


Figure 9.2: Proposed change (boxed) to UART design in PSoC.

This technique can help reduce power consumption within the PSoC serial communication block.

9.2 Wakeup/Sleep Protocol

The PSoC's active power is variable and is in mW range, whereas its sleep power is about 21.5uW. The Atmel microcontroller used in MICA2s active power is 33mW and sleep power is 75uW. The power consumption of the components depends on the mode of operation. In order to increase power saving, the MCU and the PSoC must be put to sleep when not in use. Therefore a Sleep/ Wakeup protocol could be implemented for applications with low-sampling or low-transmission rate.

9.3 Wireless Vital Signs Monitoring Device

The PSoC interfaced MICA2 design could be researched to make it a more sophisticated physiological monitoring device. It could be extended to measure body temperature, blood pressure and pulse to build a vital signs monitoring device and could be developed to be use as an alert system for health care.

9.4 Integrating into a System on Chip

It would be of great potential benefit to design a mixed signal system-on-chip, which would combine all the components required for sensing, analog as well as digital signal processing, event detection and RF communication into one single chip. It reduces the number of discrete components, the cost and most importantly it reduces wiring delays. Integrating different components into a single chip primarily reduces various line capacitances to be driven, thereby allowing us to overcome the speed limitations and at the same time reducing the overall power consumption and heat dissipation compared to discrete components. However, developing such a system-on-chip will have many design challenges: software as well as hardware and requires careful conceptualization to make the system noise tolerant, reliable and efficient. Once developed, it would incorporate all the necessary functionalities of a wireless sensor node into a single tiny integrated chip and reduce the power consumption of the sensor nodes significantly.

Research and development continues in the field of mote architecture to make smaller, more power-efficient sensor nodes. At the same time newer families of PSoC with increased computational capabilities are available. With the advances in these individual fields and with the convergence of these two technologies, it will give way to an improved computational, power-efficient, cheap and reliable monitoring system.

10. Conclusion

This thesis presents the successful implementation of an interface between MICA2 motes and a low-power Programmable System on Chip (PSoC) and demonstrates its capability of performing complex event detection. Two applications: thresholding technique and a cardiac monitoring system were developed and analyzed for their performance. It was observed that PSoC helped reduce the number of bits required to be transmitted to the base station and at the same time its current consumption was less than that of the transceiver's (CC1000) transmission current consumption in both the applications. Therefore a PSoC interfaced MICA2 would help curtail the overall power consumption in an individual node.

The reduction in transmission data in individual nodes also gives rise to fewer packets being hoped from node to node within the network path, which in turn reduces network traffic, chances of congestion and keeps the reception ratio high. This reduces the overall power consumption of the network, which in turn increases the lifetime of individual nodes and makes the network more reliable.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, *"Wireless sensor networks: a survey"*, Computer Networks 38, Elsevier, 2002. pp 1-20.

[2] Gregory J. Pottie and William J. Kaiser. *"Embedding the internet: wireless integrated network sensors"* Communications of the ACM, 43(5):51–58, May 2000.

[3] Ichiro Kawachi in collaboration with the Allostatic Load Working Group. *"Heart Rate Variability"*, John D. and Catherine T. MacArthur Research Network on Socioeconomic Status and Health; online summary available **at** www.macses.ucsf.edu/Research/Allostatic/notebook/heart.rate.html

[4] Jason Hill, *"System Architecture for Wireless Sensor Networks"*, doctoral thesis, Dept.Electrical Eng. and Computer Sciences, Univ. of California, Berkeley, 2003.

[5] EYES - Enery Efficieny Sensor Networks' website: www.eyes.eu.org

[6] Rex Min, Manish Bhardwaj, Seong-Hwan Cho, Eugene Shih, Amit Sinha, Alice Wang, Anantha Chandrakasan. *"Low-Power Wireless Sensor Networks"* vlsid, p. 205, The 14th International Conference on VLSI Design (VLSID '01), 2001

[7] Dimitrios Lymberopoulos, Andreas Savvides *"A Wireless Sensor Node Architecture for Exploring Distributed Sensor Network Applications"*, Technical paper, Yale University available at http://www.eng.yale.edu/enalab/publications/XYZ_paper.pdf

[8] TinyOS Community Forum, An open-source OS for the networked sensor regime: www.tinyos.net TINYOS

[9] Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler and Mani Srivastava, *"A Dynamic Operating System for Sensor Nodes"*, Proceedings of the 3rd international conference on Mobile systems, applications, and services, Seattle, Washington Year of Publication: 2005

[10] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. *"Contiki – a Lightweight and Flexible Operating System for Tiny Networked Sensors"*, Proceedings of the First IEEE Workshop on Embedded Networked Sensors 2004, Tampa, Florida, USA, November 2004.

[11] W. Ye, John Heidemann, Deborah Estrin , *"An Energy-Efficient MAC Protocol for Wireless Sensor Networks Wei"*, In Proceedings of the IEEE Infocom, pages 1567--1576, New York, NY, USA, June 2002. USC/Information Sciences Institute, IEEE.

[12] Wei Ye and John Heidemann, *"Medium Access Control in Wireless Sensor Networks"* University of Southern California/Information Sciences Institute TECHNICAL REPORT ISI-TR-580, OCTOBER 2003

[13] Suresh Singh and Mke Woo C. S. Mghavendra, *"Power-Aware Routing in Mobile Ad Hoc Networks"* Department of ECE Aerospace Corporation Oregon State University El SeWndo, CA 90245

[14] S. Bandyopadhyay, E. J. Coyle. "An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks". INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 3, pp 1713-1723

[15] Chalermek Intanagonwiwat, Deborah Estrin, Ramesh Govindan, and John Heidemann. "*Impact of network density on data aggregation in wireless sensor networks*".Technical Report 01-750, University of Southern California, November 2001

[16] "*2001 EDN Innovation Awards*", Reed Business Information, NY. EDN is an Electrical Design News Magazine that annually awards innovators. PSoC received an award in the 8/16 bit processor category. Available online at http://www.edn.com/info/119980.html;

[17] By Nathan John, Cypress MicroSystems, *"PSoC: Programmable System on Chip"*, EPN Online; Reed Electronics Group, available at **http://www.epn-online.com/page/11461/psoc--programmable-system-on-chip-programmable-system-on-chip.html**

[18] Victor Shnayder, Borrong, Chen, Konrad Lorincz, Thaddeus R. F. FulfordJones and Matt Welsh *"Sensor Networks for Medical Care"*, Technical Report TR-08-05, Division of Engineering and Applied Sciences, Harvard University, 2005.

[19] T. R. F. Fulford-Jones, Gu-Yeon Wei, Matt Welsh, *"A Portable, Low-Power, Wireless Two-Lead EKG System"* Proceedings of the 26th Annual International Conference of the IEEE EMBS, San Francisco, CA, USA • September 1-5, 2004,

[20] "*Heart Rate Variability ,Standards of Measurement, Physiological Interpretation, and Clinical Use*" Circulation. 1996;93:1043-1065.) © 1996 American Heart Association, Inc.

[21] "*ATmega128L Datasheet*", Atmel Corporation, revision M, updated 11/04. Available online at http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

[22] "*CC1000 Datasheet*", Chipcon Products from Texas Instruments, Edition 2.3, August 2005.

[23] "*AT45DB041 Flash Memory Datasheet*", Atmel Corporation, Updated 01/01.

[24] "*PSoC™ Designer: User Guide*", Revision 1.19 (Cypress Revision *C), PSoC Designer 4.1, Spec.# 38-12002, Last Revised: January 21, 2004, Cypress MicroSystems, Inc.

[25] "*NesC Wiring Diagram*" Wiring diagram generated by the NesC compiler. Provided by TinyOS Developers for MICA2.

[26] David Gay, Phil Levis, Rob von Behren, Matt Welsh, Eric Brewer and David Culler, "*The nesC Language: A Holistic Approach to Networked Embedded Systems*", In SIGPLAN Conference on Programming Language Design and Implementation (PLDI'03), June 2003.

[27] "*Cypress CY8C27443 Datasheet*", Cypress Semiconductors, Revision *J, May 2005

[29] http://www.xbow.com/, Crossbow Technology, Inc; San Jose, CA

[30] "*Programmable Gain Amplifier*", PSoC User Module: PGA Datasheet, made available by Cypress; PGAv3.2

31] "*7- to 13-Bit Variable Resolution Incremental ADC*", PSoC User Module: ADC Datasheet, made available by Cypress, ADCINCVR v3.1.

[32] "*UART*"; PSoC User Module: UART Datasheet, made available by Cypress; UART v5.2.

[33] "*MAX220-MAX249 Datasheet*", Multi-channel RS232 Drivers-Receivers, from Maxim Integrated Products; *19-4323; Rev 14; 8/04*

[34] Jeff Thorn, "*Deciphering TinyOS Serial Packets*", Octave Tech Brief #5-01, available at http://www.octavetech.com/pubs/TB5-01%20Deciphering%20TinyOS%20Serial%20Packets.pdf

[35] "*TinyOS and NesC tutorials*", made available by TinyOS Developers for TinyOS users, available at: http://www.tinyos.net/tinyos-1.x/doc/tutorial/

[36] R.M. Passos, Claudinor J.N. Coelho Jr, Antonio A.F. Loureiro and Raquel A. F. Mini, "Dynamic Power Management in Wireless Sensor Networks: An Application-driven approach", *Wireless On-demand Network Systems and Services (wons),* vol. 00, no. , pp. 109-118, Second 2005.

[37] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A Wireless Sensor Network for Structural Monitoring," in *Proceedings of the ACM Conferenceon Embedded Networked Sensor Systems*, Baltimore, MD, USA, November 2004.

[38] M Gaynor, S. L. Moulton, M Welsh, Ed LaCombe, A Rowan, J Wynne, "*Integrating Wireless Sensor Networks with the Grid*", Published by the IEEE Computer Society, IEEE INTERNET COMPUTING JULY • AUGUST 2004

[39] T. Thong, K. Li, J. McNames, M. Aboy, B. Goldstein, "Accuracy of ultra-short heart rate variability measures, "*Annual International Conference of the IEEE Engineering in Medicine and Biology Proceedings*, Cancun, Mexico, 17-21 September 2003, pp. 2424-2427.

[40] "*Hardware Overview*", Seminar by Crossbow Inc. at San Jose, February, 9-10 2005.

[41] AbouGhazaleh, N.; Lanigan, P.; Gobriel, S.; Mosse, D.; Melhem, R. "Dynamic rate-selection for extending the lifetime of energy-constrained networks", *Performance, Computing, and Communications, 2004 IEEE International Conference on 2004 Page(s):553 – 558*

[42] Chin Choy Chai; Tuan Haw Bok, "A power efficient algorithm for power/rate control in wireless transmission", *Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference.*

[43] Sridharan, A.; Krishnamachari, B.; "Max-min fair collision-free scheduling for wireless sensor networks", *Performance, Computing, and Communications, 2004 IEEE International Conference on 2004 Page(s):585 – 590*

[44] "*The family of motes preceeding Telos and their capabilities*", Updated Sept 2004, University of Berkeley's Wireless Embedded Systems website. Available at http://webs.cs.berkeley.edu/papers/hotchips-2004-mote-table.pdf

[45] M. Younis, M.Youssef, K. Arisha; "*Energy-aware management for cluster-based sensor networks*", Computer Networks: The International Journal of Computer and Telecommunications Networking. Volume 43 , Issue 5 (December 2003) Pages: 649 – 668. Year of Publication: 2003

[46] M. Bhardwaj et al., "*Upper bounds on the lifetime of sensor networks*", in: Proceedings of ICC 2001, June 2001.

[47] W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy Efficient Communication Protocol for Wireless Microsensor Networks",presentation at Massachusetts Institute of Technology. Available online at http://faculty.cs.tamu.edu/dzsong/teaching/fall2005/cpsc689/slides/Energy%20Efficient%20Commn%20Protocol.ppt

[48] GORDON E. MOORE, "Cramming More Components onto Integrated Circuits", *PROCEEDINGS OF THE IEEE, VOL. 86, NO. 1, JANUARY 1998*

[49] J. Hill et al., "System Architecture Directions for Networked Sensors," *Proc. 9th Int'l Conf. Architectural Support for Programming Languages and Operating Systems* (ASPLOS 2000), ACM Press, 2000, pp. 93–104.

A. Appendix

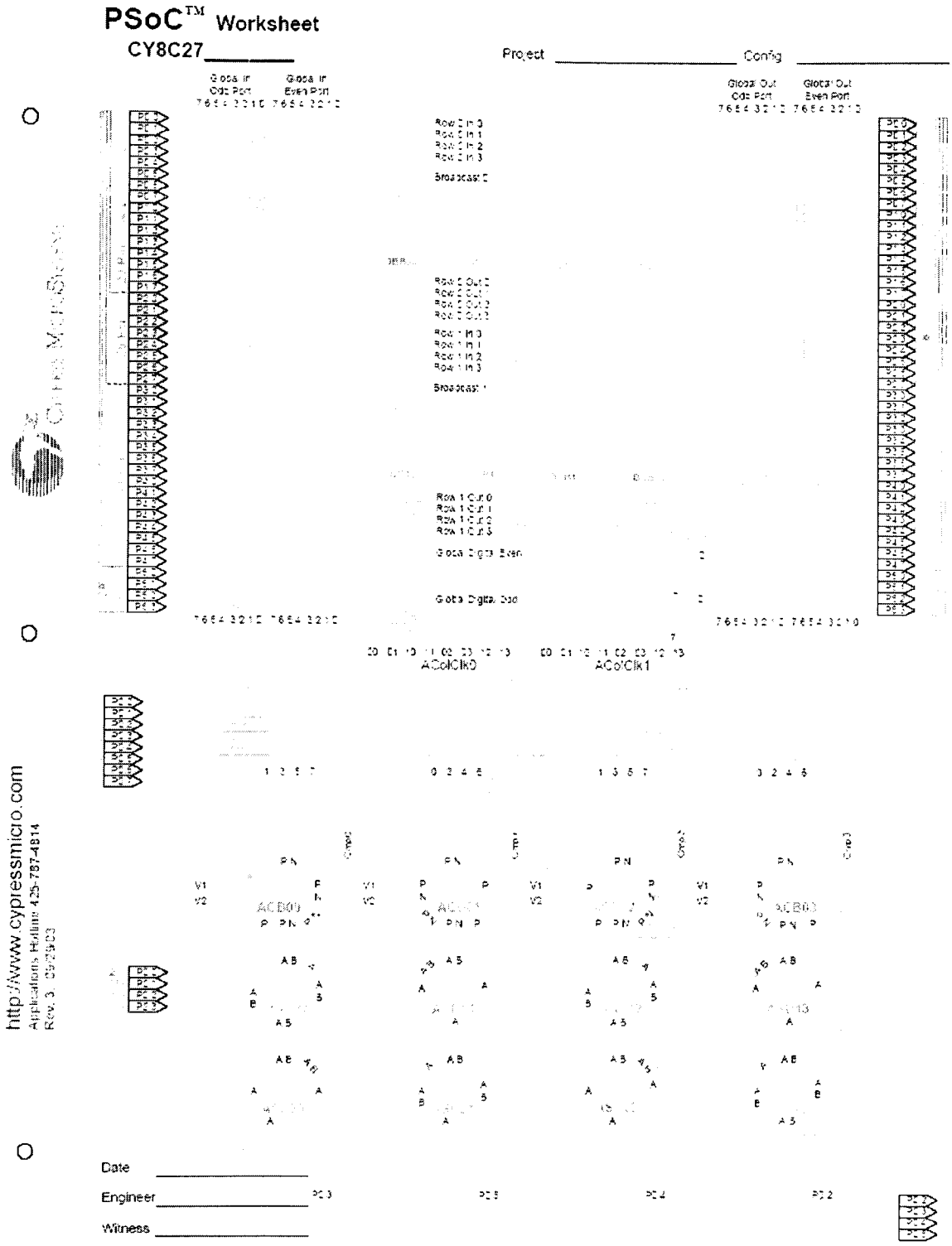


Figure A.1: Block level diagram of configurable digital and analog blocks, showing the configurable interconnects in CY8C27xxx family [28]