March 8th, 2007

Mr. Lakshman One
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

Re: ENSC 440/305 Design Specification for Theater Ushering System

Dear Mr. One,

Please find the attached document, *Design Specification for Theater Ushering System*, outlining the design approach for our prototype device. Our project is to build a simple ushering system for movie entertainment industry use. The system is aimed to provide convenience and other fun features for moviegoers.

The purpose of this design specification is to layout the detailed design and testing for our system at each stage of development. The document lists the design approach of the prototype system, which is scheduled for demonstration during April 2007.

U-Nexus Inc consists of four senior-level undergraduate students in the electronics option. Team members consist of Danny Chan, Gordon Lee, Bo Wang and Eric Wang. Please do not hesitate to contact us if you have any concerns, we can be reached at ensc-unexus@sfu.ca.

Best Regards,

Eric Wang

Eric Wang
CEO
U-Nexus Inc

Enclosure: *Design Specifications for Theater Ushering System*

ENSC 305/440 Design Specification:

Theater Ushering System

Project Team:	Danny Chan Gordon Lee Bo Wang Eric Wang
Contact Person:	Eric Wang ensc-unexus@sfu.ca
Submitted to:	Mr. Lakshman One – ENSC 440 Mr. Steve Whitmore – ENSC 305
Issued date:	March 8th, 2007
Revision:	1.0

Executive Summary

U-Nexus Inc is currently in the process of developing a system capable of changing the movie going experience forever. The system is called *Theater Ushering System* and is targeted specifically for the movie entertainment industry. As its name suggests, it acts as a seating usher by providing moviegoers with location on seating priori to their entrance of a movie screen. The system is also capable of other features such as advertisement display or movie trailers. With additional add-ons, interactive features such as opinion polling and trivia games can be incorporated as well. The goal of the system is to enhance and promote movie experience to a new level.

The system is split into three phases, where we will present a working prototype in first phase, engineering prototype in the second phase and manufacturing model in the last phase. Working prototype will be used to demonstrate usability with limited features and will be crude in terms of aesthetic. Engineering prototype will have all features completed and the system running on embedded system. The manufacturing model will then have all components from engineering prototype, except all components will be safely enclosed with wiring path properly shown. Phase 1 will be targeted for completion for April 2007, and will have the following features

1. System will be able to monitor status of seats and display them via display panel
2. System will communicate seating information wirelessly using low power design
3. System display will be able to run advertisements, trailers and seating information simultaneously

The two subsequent phases are scheduled depending on feature implementation progress. The remainder of this document is dedicated to the detailed discussion of functional specifications for each design phase.

Table of Contents

List of Figures

List of Tables

List of Flowcharts

1. Introduction

The Theater Ushering System is a device specifically designed for movie theater industry, used to provide convenience and interaction to moviegoers. One of the major tasks of the systems is to provide seating information on display panel prior to entrance of a movie screen. Moviegoers can gather location of vacant seats before entrance, thus save the daunting tasks of seat finding. A second major task is for the system display panel to have additional features such as movie trailers and advertisements capabilities. The system uses wireless communication protocols for transmitting seating information to the display panel, thus minimizing the changes introduced to the infrastructure.

The project will be divided up into stages, where a working prototype will be completed during mid April 2007. Engineering prototype with embedded system is targeted for August 2007 and the manufacturing model is targeted for February of 2008.

1.1. Scope

The scope of this document will be descriptions on the design approaches to the Theater Ushering System. Note that the design methods discussed are for the working prototype scheduled to complete during April 2007. Further models such as engineering and production models are expected to retain majority of the design of the prototype; however, we cannot guarantee the completeness of this documentation. Due to possible changes, we will frequently revise this document to reflect the changes accordingly.

1.2. Acronyms

FAQ	-	Frequently Asked Questions
TUS	-	Theater Ushering System
SSM	-	Seating Status Monitor
SST	-	Seating Status Transmitter
RF	-	Radio Frequency
MCU	-	Microcontroller unit
DU	-	Display Unit
AU	-	Acquisition Unit
MPEG	-	Motion Picture Expert Group
QT	-	QuickTime
WMV	-	Window Media Video
RM	-	Real Media
JPEG	-	Joint Photographic Expert Group
GIF	-	Graphic Interchange Format
PNG	-	Portable Network Graphics
IR	-	Infrared

Txt	-	Text File
VGA	-	Video Graphics Array
DVI	-	Digital Visual Interface
S-Video	-	Separated Video
MAC	-	Medium Access Control
PHY	-	Physical Layer
LR-WPAN	-	Low-Rate Wireless Personal Area Networks
LED	-	Light Emitting Diode
RAM	-	Random Access Memory

1.3. Intended Audience

This document is intended for design engineers, product managers, marketing specialists and lawyers.

- Design engineers can use this document as a guideline during development.
- Product managers can use this document as an overview of the project to better estimate development time and milestones.
- Marketing specialists can use this document to present to potential customers and develop promotional materials.
- Lawyers can use this document as reference in case if there is a dispute presents.

2. System Overview

2.1. Simplified System Overview

In this section, we discuss the designs to key components of the *Theater Ushering System*. Figure 2.1 shows the major components in a system block diagram. The Seating Status Transmitter (SST) is responsible for data acquisition used to gather the current status of seats around the theater. After acquisition, information is then transmitted to the Seating Status Monitor (SSM). Upon receiving the seating information, SSM will perform the necessary processing and present the seating information to the audience via a large screen display. So the general flow of the system is data acquisition, data transmission, data receiving, data processing and lastly, data output [1].

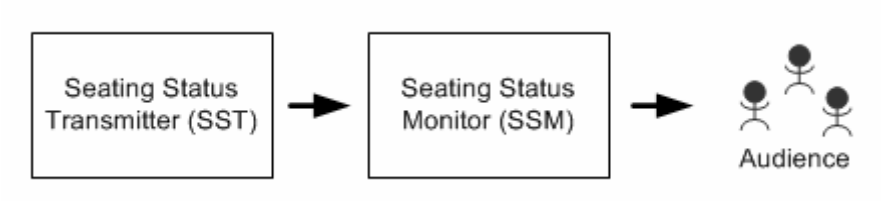


Figure 2-1: Simplified System Block Diagram

2.2. Detailed System Overview

Figure 2.2 shows the detailed system block diagram of the Theater Ushering System. Seating Status Transmitter (SST) is further divide into two subsystems, known as Acquisition Unit, and Beacon. In short, Acquisition Unit (AU) utilizes the proximity sensor to monitor the seat occupancy information, and Beacon is the wireless transceiver device for data delivering.

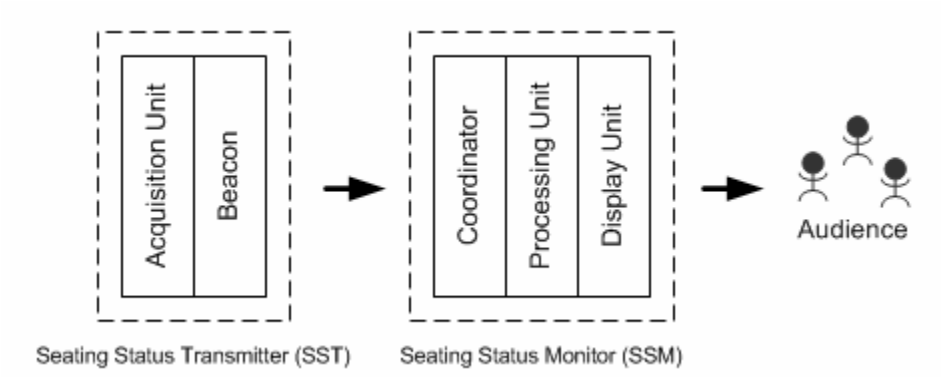


Figure 2-2: Detailed System Block Diagram

Similarly, Seating Status Monitor (SSM) is further divide into Coordinator, Processing Unit and Display Unit. In short, Coordinator is the wireless transceiver device for communicating with

Beacon. Processing Unit processes the seat's occupancy information and advertisements for display unit. Display Unit presents such information to the audiences.

In summary, SST obtains the seat's occupancy information through Acquisition Unit and transfers such information via Beacon to the SSM's wireless transceiver. After SSM received such information via the Coordinator, it then processes the seat's occupancy information and the preloaded advertisements for displaying to the audiences.

In the reset of this section, we give more in depth descriptions and summarize the duties for each individual component in the Theater Ushering System.

2.2.1. Acquisition Unit

Acquisition unit interfaces with about 8-16 proximity sensors to obtain the relevant seat occupancy information. This unit shall be capable of processing the proximity sensor information and transfer such information to the SST's wireless transceiver also know as Beacon.

2.2.2. Beacon

Beacon is a wireless transceiver which compliant with the IEEE 802.15.4 telecommunication protocol [2]. Its duty is to process the seat occupancy information obtained from the Acquisition Unit, and packaging such information so that it is compliant with the IEEE 802.15.4 protocol for delivering to the SSM's wireless transceiver or namely, the Coordinator.

2.2.3. Coordinator

Coordinator is also a wireless transceiver which compliant with the IEEE 802.15.4 telecommunication protocol [2]. Its duties are to initiate the handshaking procedure with all the Beacon stations, to schedule the wireless communication slot, and to maintain the wireless traffic control in a TUS. Simultaneously, Coordinator also delivers the seat occupancy information obtained from the Beacon to the Processing Unit through the RS232 communication protocol.

2.2.4. Processing Unit

In the working prototype stage, Processing Unit is a personal laptop computer. Its duties include communicating with the Coordinator to obtain the seat's occupancy information, processing the seat's information for display unit, scheduling the advertisements, and providing input for setup/maintenance on the TUS.

2.2.5. Display Unit

Lastly, the Display Unit obtains the seat information or advertisements from the Processing Unit through VGA, DVI, S-Video, or Component Video for displaying to the audience.

Due to time limitation, our working prototype uses the laptop computer as both the processing unit and the display unit. However, one can always build an embedded system to replace the computer. This is scheduled in the engineering prototype.

2.3. Overall System Overview

Previously in the Figure 2.1.1, we presented the simplified TUS system which only capable of handling 8-16 seats due to the fact that an Acquisition Unit only interface with 8-16 proximity sensors. However in real world scenario, typical movie theatres have more than 150 seats. In addition, the number seats per screen are subject to the design of the individual infrastructure. To overcome these difficulties, we will expand our simplified design mode and make use of the wireless advantage. As the result, the overall TUS design model has many sets of SSTs communicating wirelessly with single SSM at scheduled time interval as shown in the Figure 2.3. The single SSM will process the information, and then display seating information to the audiences at the entrance. This design solution overcomes the difficulty with number of seats required by different size of movie threat and as a bonus it allows ease of deployment.

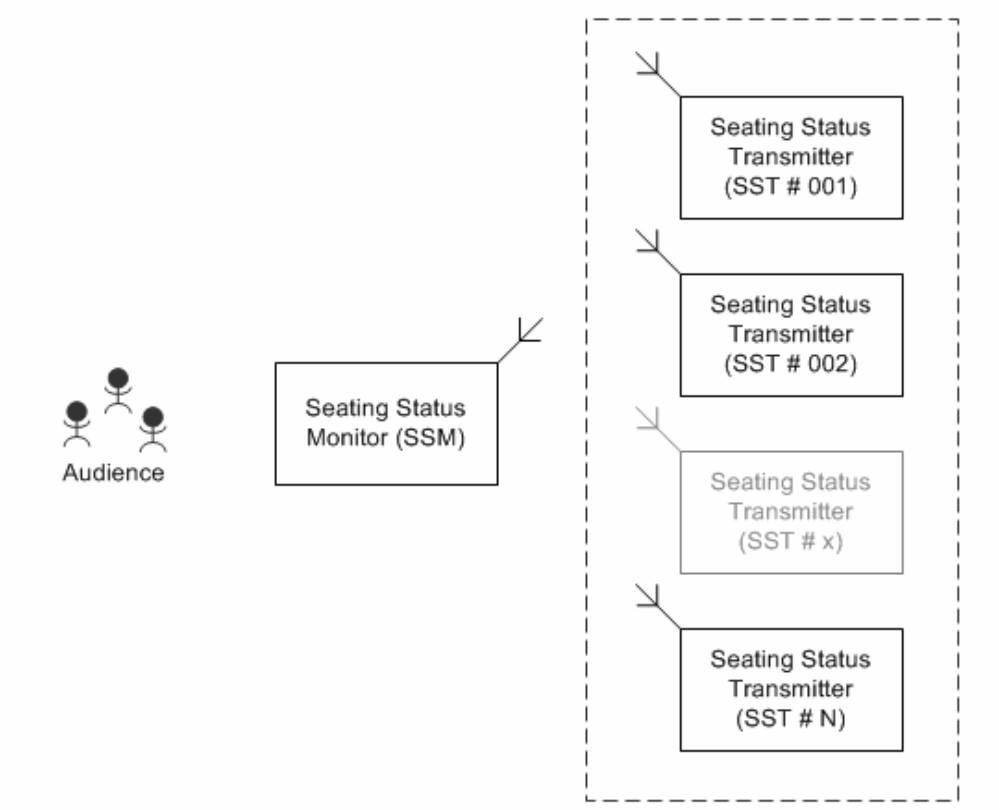


Figure 2-3: TUS Design Model

In our working prototype type demo, we are targeting to have a single SSM wireless communicating with 2 to 3 SST managing about 16-24 seat occupancy status.

3. System Hardware

In this section, we describe the fundamental hardware required for developing the Theater Ushering System (TUS). We present the hardware by each component of the system as discussed in the Figure 2.2.1.

3.1. Acquisition Unit

The acquisition unit (AU) is used to collect seating data and transmit it to Beacon. AU is connected in series and implemented on each chair. A Beacon is then connected to several AU, so that seating information can be transmitted.

3.1.1. D Flip Flop: LS74N and LS74C

LS74N and LS74C are two the D Flip-Flop which stores the seats occupancy information. D Flip Flop will be used along with the NAND gate to form the parallel in serial out register. Please refer to Section 4 for detail design.

3.1.2. NAND Gate: SN74LS00

SN74LS00 is a Quad 2 input NAND gate circuit. We will use it along with D Flip-flop in the previous section to build our parallel in serial out register. Please refer to Section 4 for detail design.

3.1.3. IR Proximity Sensor

Infrared proximity sensor is used to determine whether the seat is occupied or not. Our design is targeted to use the Sharp GP2Y0D340K compact distance measuring sensor, which has a detecting distance of 10 to 60cm.

3.2. Coordinator / Beacon Design

Coordinator, and Beacon described in Figure 2-2 are identical hardware units. The difference are in the way how they response or the software that we write. Thus by loading different firmware, we can modify their behaviors differently.

Figure 3-1 shows an overview of CC2420DB. It also includes the description with various communications. Please refer to the reference document [4] for more detail information. We will briefly go through the major part of the device.

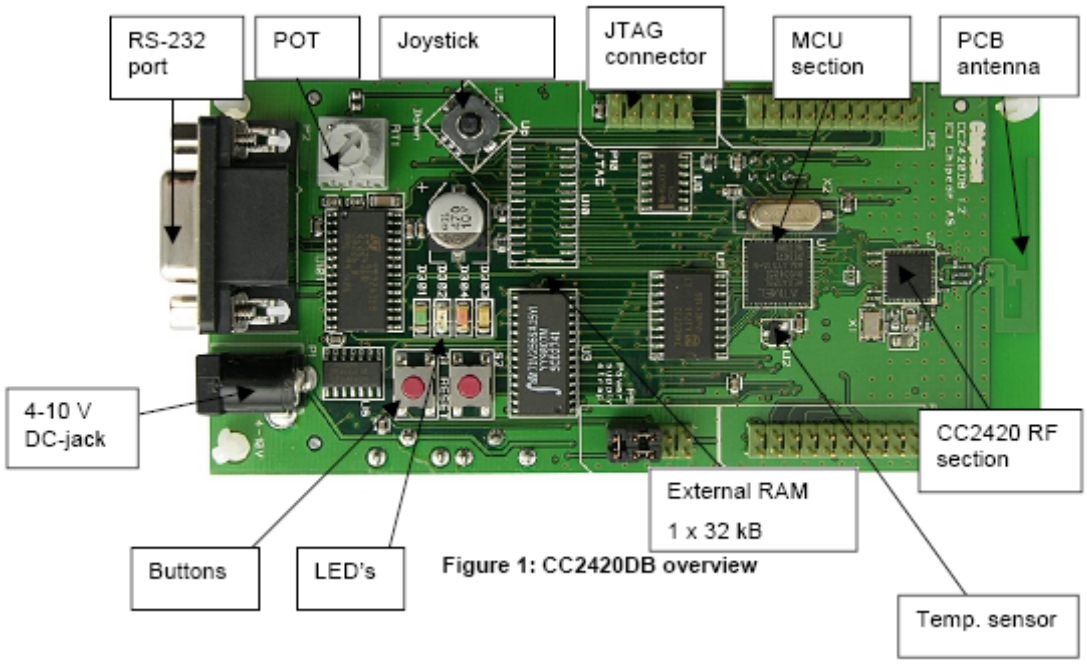


Figure 1: CC2420DB overview

Figure 3-1: CC2420DB Board Layout

3.2.1. AVR Atmega 128L

This microprocessor is the brain of the wireless communication in our TUS. It communicates with the Chipcon CC2420 RF chip, manage the wireless networking, and run application to transfer data. In additional, it has 128KB of in system flash program memory, and 4KB of SRAM data memory, and 4KB of non-volatile EEPROM data memory. The controller is connected to the CC2420 via its build-in SPI interface as well as some general I/O pins for interrupts.

3.2.2. IEEE 802.15.4 RF Chip

Chipcon CC2420 RF is an IEEE 802.15.4 compliance RF chip. It's responsible for all the RF transmission and reception. In additional, it has the CRC hardware error checking algorithm. For more information please visit www.chipcon.com [4].

3.2.3. Power Supply

The power supply contains two voltage regulators: a 3.3 V regulator is used for the microcontroller and the I/O pins of the CC2420. The CC2420 also have its own build in voltage regulator which generate a 1.8V to supply the CC2420 core. Two power sources are available, a 2.5mm DC jack type connector, and a 9V battery.

3.2.4. RS-232 Interface

A serial port is included on the CC2420DB. We will use this port to transmit and receive data from AU or PU when necessary.

3.2.5. LED and Button

Four LEDs and two buttons are included on the CC2420DB. We will use the status of the LED to indicate and help us developing the code. Buttons are used for resetting the device and programming the onboard flash memory.

3.3. Processing Unit

Processing unit is essential and is used to decode the received seating information from SSTs. Our initial target design for processing unit was to use an X86 embedded system. The advantage of using embedded system is that we can customize the specifications to our need and to reduce unit costs. Another prime advantage is for the aesthetic of the unit, which can be much more appealing than a PC computer. However, due to time constraints, processing unit for our prototype stage will simply be a traditional PC computer. The processing unit must be capable of running the designed software interface.

An ideal processing unit must meet the following minimum requirement

- Pentium 3 processor or greater
- 512 MB of RAM
- 40 GB of hard disk space
- RS232 communication interface
- Video output such as VGA or DVI

The specification given is simply a typical home user computer. By having such processing power, we can improve our display to three dimensional if necessary.

3.4. Display Unit

The engineering prototype of the display unit should be a large 17 inch flat panel capable of displaying resolution up to 1280 x 1024 pixels at 16 bit color. The engineering prototype display unit will be connected to the embedded process unit and both will be mounted of the wall with proper support.

For the working prototype design, the display unit will be the 14 inch flat display screen of the laptop computer. As previously mentioned, the laptop will also be used as the processing unit. The 14 inch display screen is already attached and integrated onto the laptop; therefore, the laptop itself can be represented as a fully functional working display unit connected with the

embedded processing unit. In addition, the laptop can be connected to projector to project the display content, hence emulating a very large display unit

4. SST Design

In this section, we describe the design specification for our SST unit. We will focus on designs regarding data acquisition from seats and data interface with Beacons. In data acquisition, we discuss design of combinatorial circuit, which allows us to reduce the number of wires, avoid irregular voltage problems and still maintain compact design. In Beacon interfacing, we describe our circuit layout and software approach used to communicate reliably.

The SST unit works as a functional block of the whole system. The SST unit would stay in standby mode during idle state, and will only provide seating information when control unit is requesting for data.

4.1. Overall Design

Our overall system is designed so that each row has a Beacon, which communicates with the Coordinator wirelessly to update the seating information upon request. Each Beacon will have various numbers of data acquisition units. The number of seats connected to one Beacon can vary and can be adjusted by the operator according to theater arrangement. The overall system diagram for the AU is shown as following.

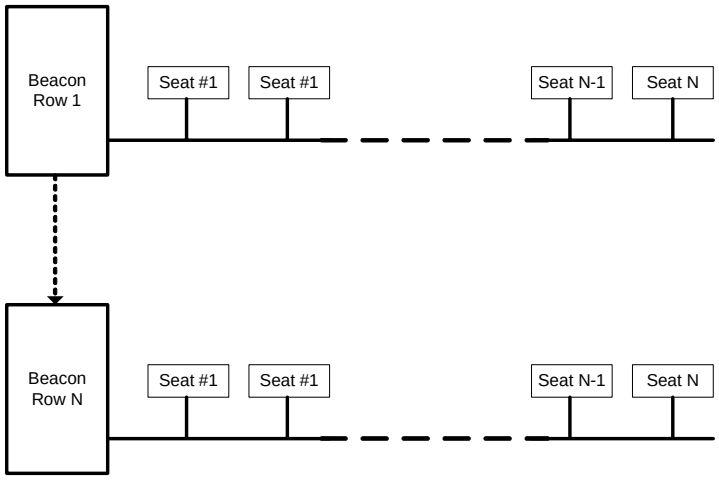


Figure 4-1: Overall System for AU

One of the concern was that large amount of wiring will be needed, thus to reduce wiring, we will use a serial connection to connect all the data acquisition units. By using such serial approach, we can reduce the total number of wires of our implementation down to only four wires. Specifically, they are load, shift, clear, and data wires. We will provide more detail in the following sections. Also, to reduce parts usage, we will take advantage of the microcontroller on CC2420 development board, and we will make use of the pre-connected RS232 setup to take advantage of the easy connection.

4.2. Data Acquisition Unit Design

We decided to use parallel-in/serial-out shift-register for the serial communication. Besides reducing wiring complexity, this combinational logic design gives us another advantage of low power consumption. In this section, we will describe our combinational logic circuit design of the data acquisition unit. An example will be used to explain our design approach.

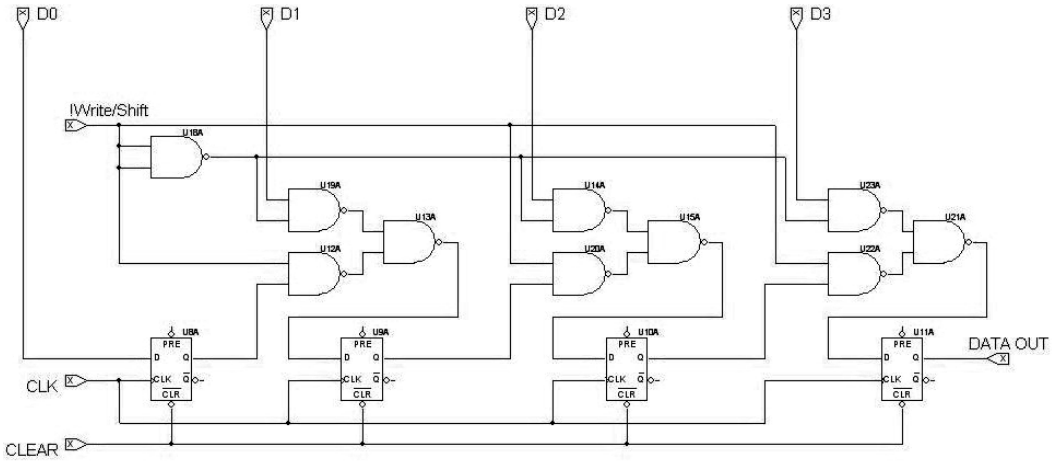


Figure 4-2: Data Acquisition Unit Logic Design

This circuit is divided into four sections where each section is responsible of reading a separated seat status. From Figure 4-3, only the first section is different from other sections, meaning all following sections share the same circuit design. Such repeated circuit use gives us an easier implementation for our final product. Each board will only have two chips so that power usage can be minimized.

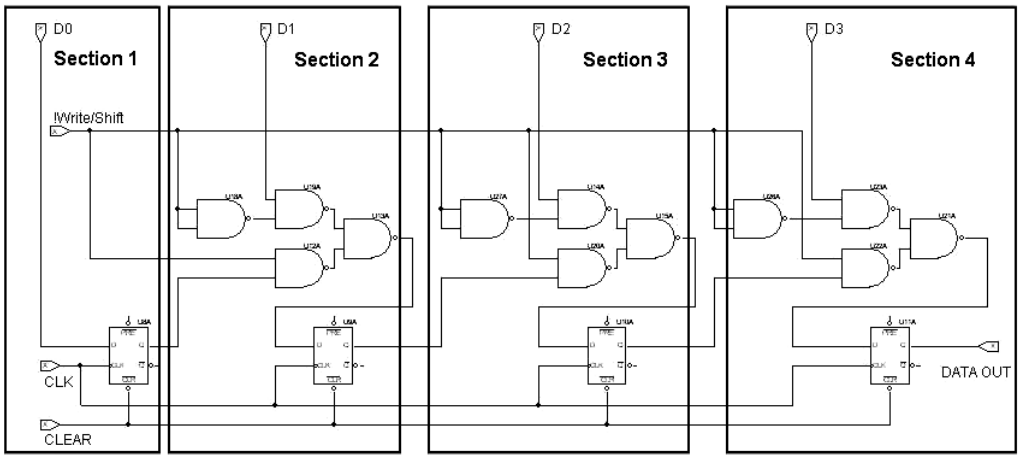


Figure 4-3: Data Acquisition Unit Logic Design Breakdown

One future improvement for our design is to connect two D flip-flops in series to avoid meta-stability problem. We currently do not have this design implemented because our operating speed is relatively slow, hence we do not expect to see such phenomenon to occur. However, to make sure that meta-stability does not happen, we will perform extensive testing. If the problem does arise, we will then introduce the two additional D flip-flops as needed.

4.3. Interfacing with Beacon

In this section, we focus on the communication protocol between AU and the Beacon. Since we will use parallel-to-serial shift register in our design, we will need CLK and !WRITE/SHIFT signals to drive the combinational logic circuit. As mentioned previously in Section 4.2, the data acquisition units share the same microcontroller with the Beacon and they will be connected using DB9 connector.

On the development board, the ATMEGA128 microcontroller uses CMOS standard on I/O ports and there is MAX3243 transceiver that converts the CMOS standard into RS232 standard in order to be able to communicate with PC. Therefore, the signal at DB9 connector is at a level of +/- 12V. Since our combinational logic circuit is operated at TTL level, we need to add a transceiver to change the signal type

The overall system diagram for interfacing with Beacon is shown in Figure 4-4.

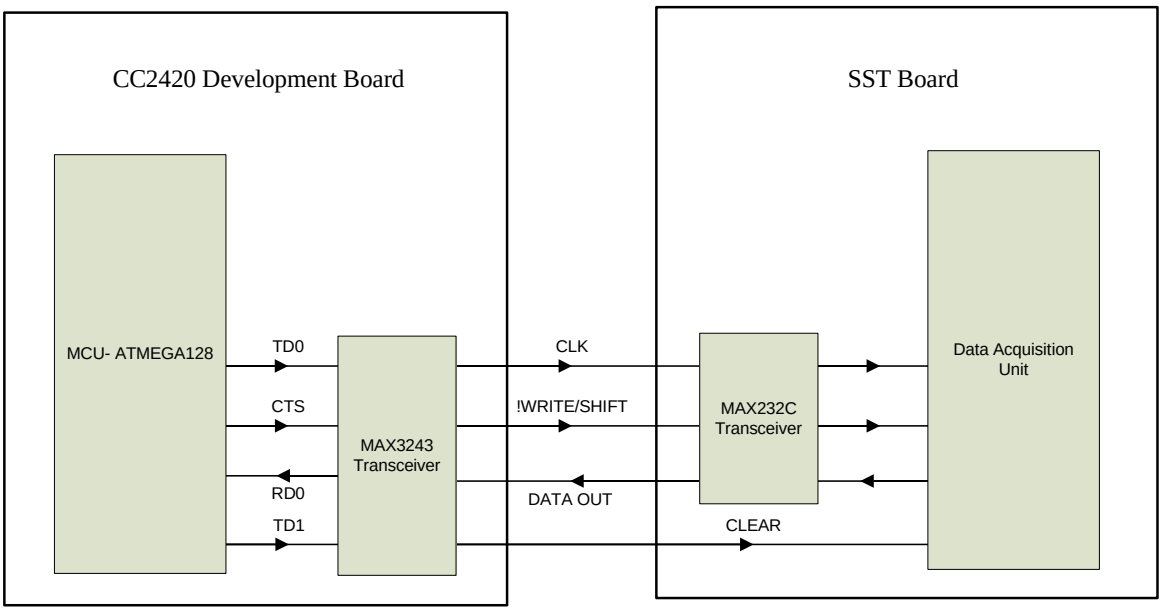


Figure 4-4: System Overview for Interfacing with Beacon

Additionally, we will not use RS232 standard serial communication, but instead, uses another simpler form of serial communication method. We will disable the RS232 communication on the microcontroller during operation, and use those pins as our control and data signal line. As

shown in Figure 4-4, we adapted *TD0* pin as *CLK*, *CTS* pin as *!WRITE/SHIFT*, *RD0* pin as *DATA_OUT* and *TD1* pin as *CLEAR*.

4.4. AU Schematic Circuitry Design

In this section, we describe the schematic circuit of the data acquisition unit. There are three types of circuit boards in our design.

- *TYPE I:* First board that connects to the beacon.
- *TYPE II:* Middle boards that are connected in series are identical.
- *TYPE III:* Last board that connects to beacon with data output line.

4.4.1. Type I Board

The Type I board is connected to beacon. A total of three chips will be placed on the board: ***7805*** voltage regulator, ***MAX232*** transceiver and ***SN74LS74*** Dual D flip-flop. Input port consists of six pins with one unconnected: *Power*, *GND*, *CLK*, *!WRITE/SHIFT* and *DATA_OUT*. Output port also consists of six pins: *Power*, *GND*, *CLK*, *!WRITE/SHIFT*, *DATA_CARRY* and *DATA_OUT*. The first type is shown in Figure 4-5 and all the ports are circled in red.

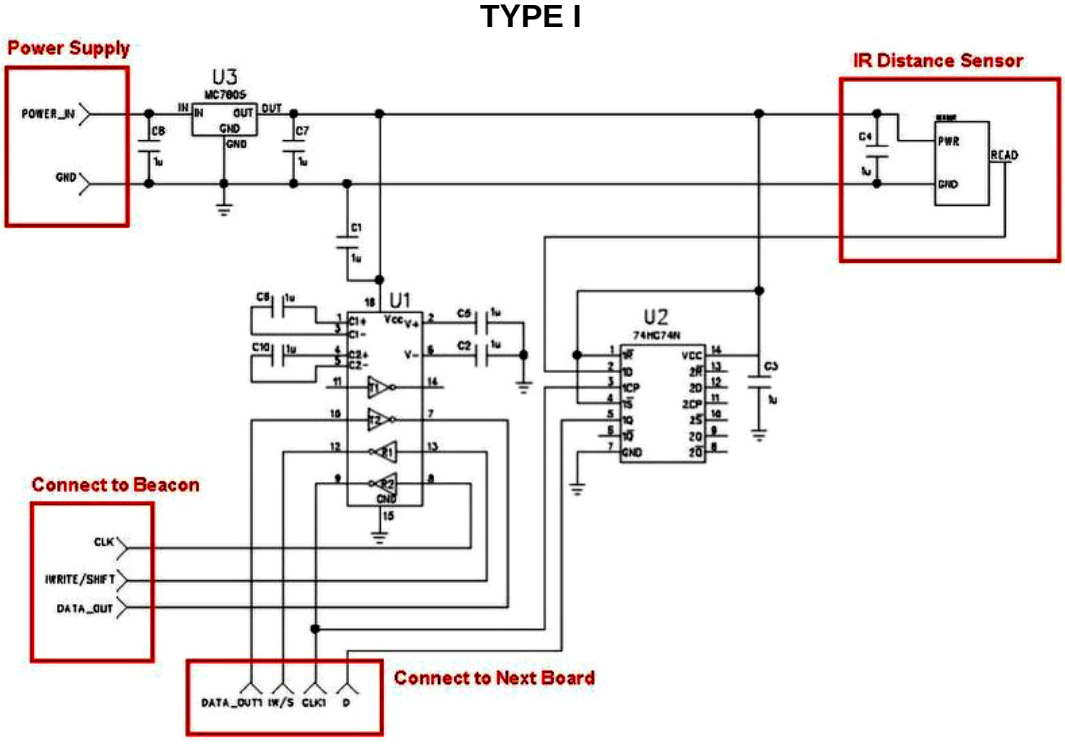


Figure 4-5: Data Acquisition Schematics Circuitry – First board

4.4.2. Type II Board

The Type II board is connected after the first board. A total of three chips are on the board: ***7805*** voltage regulator, ***SN74LS00*** Quad 2-input NAND gate and ***SN74LS74*** Dual D flip-flop. Input port consists of six pins: *Power*, *GND*, *CLK*, *!WRITE/SHIFT*, *DATA_CARRY* and *DATA_OUT*. Output port also consists of six pins: *Power*, *GND*, *CLK*, *!WRITE/SHIFT*, *DATA_CARRY* and *DATA_OUT*. The schematic of second type is shown in Figure 4-6 with ports circled in red.

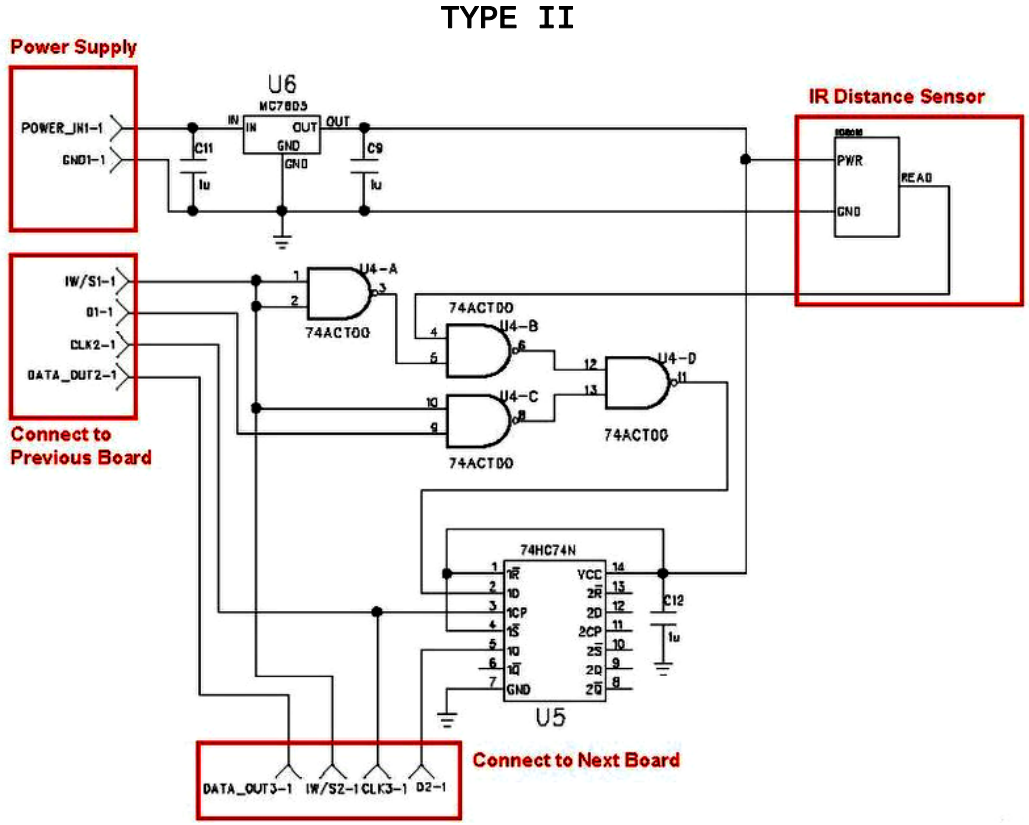


Figure 4-6: Data Acquisition Schematics Circuitry – Middle Boards

4.4.3. Type III Board

The Type III board is the last board connected in series. A total of three chips are on the board: ***7805*** voltage regulator, ***SN74LS00*** Quad 2-input NAND gate and ***SN74LS74*** Dual D flip-flop. Input port consists of six pins: *Power*, *GND*, *CLK*, *!WRITE/SHIFT*, *DATA_CARRY* and *DATA_OUT*. There is no output port for this last board. The third type is shown in Figure 4-7 with ports circled in red.

TYPE III

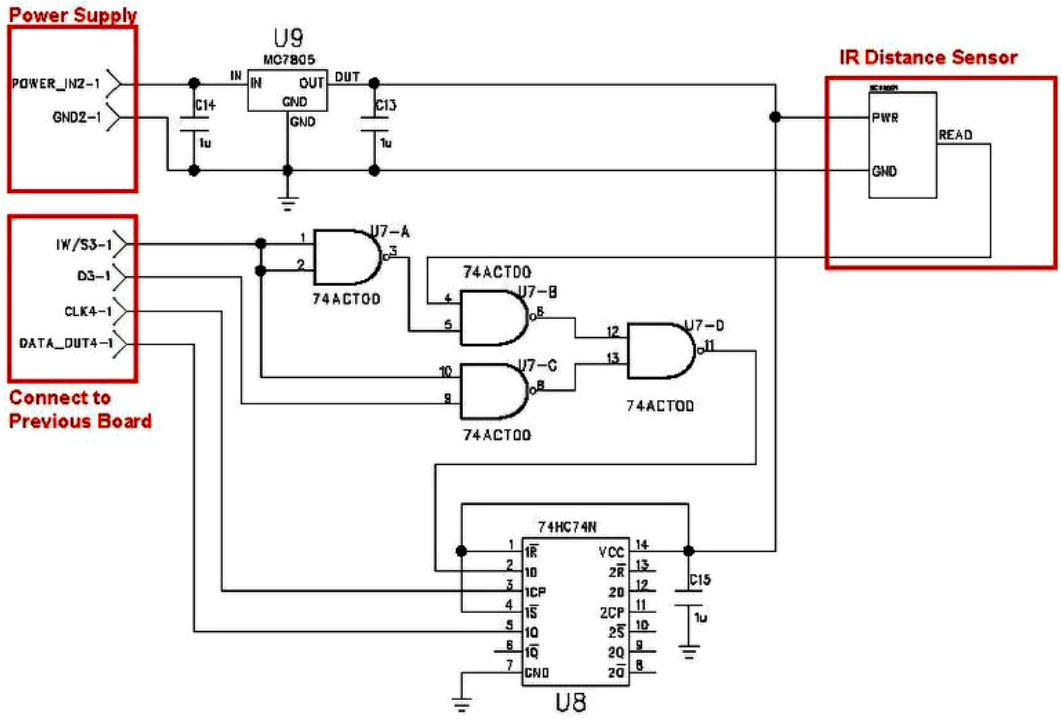
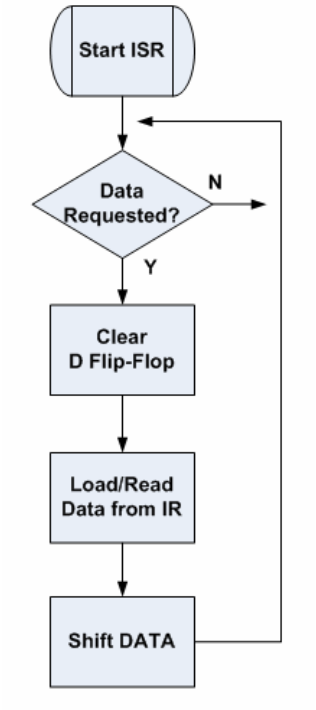


Figure 4-7: Data Acquisition Schematics Circuitry – Last board

4.5. Operating Algorithm

4.5.1. Hardware Algorithm

Flowchart 4-1 describes the algorithm for the Acquisition Unit. When Beacon first starts, it will wait for a triggering signal before any action. If Processing Unit is querying for seat occupancy, the AU will first clear out all the content in the D Flip-Flop. Then, it will read the status from the IR proximity sensor, and store it. Lastly, depending on how many seats are connected to a Beacon, the data will be shift back through the data line by triggering the CLK port in the D Flip-Flop. In summary, when the Processing Unit is querying for seat occupancy information, the AU will clear data in D Flip-Flop, read/store the current status from IR, and transmit the individual seat occupancy back to Beacon.



Flowchart 4-1: Flow Chart of Data Acquisition Unit

4.5.2. Software Algorithm

Since the serial ports of ATMEGA128 are located at port E and D, we need to configure the direction and data registers of those ports. Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both VCC and Ground. This means that in external circuit, we do not need pull-up resisters. Figure 4-8 and 4-9 shows the data direction and data register of port D. Figure 4-10 and 4-11 shows the same register for port E.

Port D Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 4-8: Port D Direction Register

Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 4-9: Port D Data Register

Port E Data Direction Register – DDRE

Bit	7	6	5	4	3	2	1	0	
	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	DDRE
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 4-10: Port E Direction Register

Port E Data Direction Register – DDRE

Bit	7	6	5	4	3	2	1	0	
	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	DDRE
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 4-11: Port E Data Register

Some of the mappings will be essential:

- RD0 is mapped to PE0; set PE0 as INPUT by `DDRE = DDRE || 0x01`
- TD0 is mapped to PE1; set PE1 as OUTPUT by `DDRE = DDRE && 0xFE`
- CTS is mapped to PD7; set PD7 as OUTPUT by `DDRD = DDRD && 0xBF`
- TD1 is mapped to PD3; set PD3 as OUTPUT by `DDRD = DDRD && 0xFB`

4.6. Firmware Class/Object Design

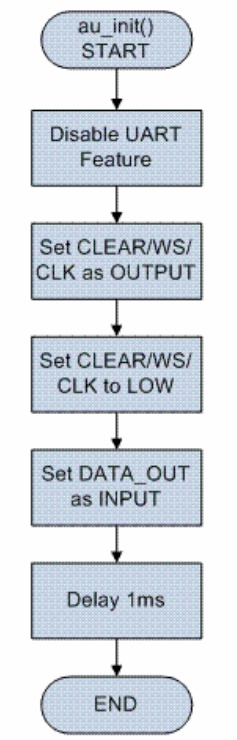
In this section, we describe the functions that will be programmed for the microcontroller.

```
Structure {
    Char NS [32];          // number of seats per SST
    Char SOS [64];         // seat occupancy status, assume a SST have maximum of 16 seats
    void au_init();        // initializing the customized serial port
    void au_clear();       // used to clear the AU unit
    void au_read();        // used to read the AU unit
    void au_tx();          // shift the data out to beacon
}
```

Here is the brief explain of the each function:

- **au_init():**

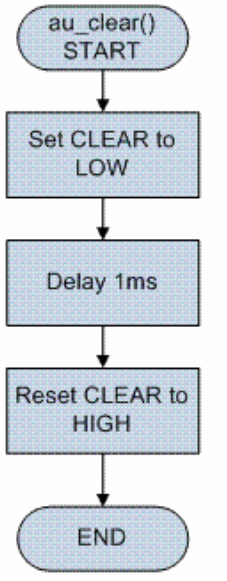
  The function initializes the customized serial port on the microcontroller. The flow chart is shown in Flowchart 4-2.



Flowchart 4-2: Flow Chart of Data au_init() Function

- **au_clear():**

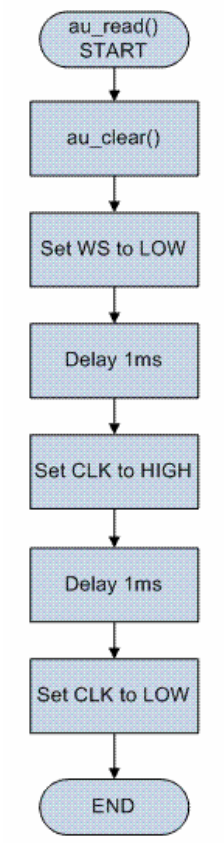
  This function commands microcontroller to generate the CLEAR signal to AU circuit; hence it will clear all the information stored in the D Flip-Flop. The flow chart is shown in Flowchart 4-3.



Flowchart 4-3: Flow Chart of Data au_clear() Function

- **au_read():**

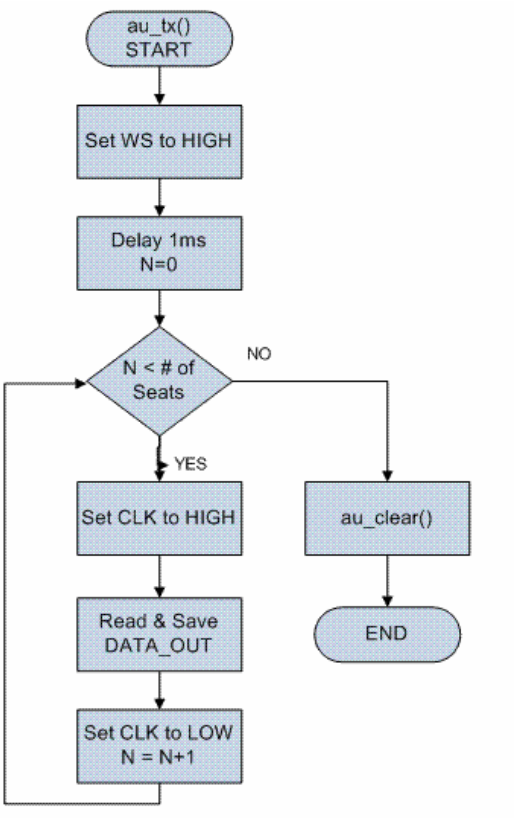
  This function commands microcontroller to generate the WRITE signal to read the input, which is read from the current IR proximity sensor. Data read will be stored in the associated D Flip-Flop. The flow chart is shown in Flowchart 4-4.



Flowchart 4-4: Flow Chart of Data au_read() Function

- **au_tx():**

  This function commands microcontroller to generate CLOCK signal to shift the data stored in the D Flip-Flop back to the Beacon. The process is again shown in Flowchart 4-5.



Flowchart 4-5: Flow Chart of Data au_tx() Function

4.7. Hardware Test Plan

In this section, we describe the hardware test plan. For our project, several circuits have to be soldered by hand. Not only we need to verify our design is correct, but also we need to check the soldering and connection.

4.7.1. Component / Component Connection Verification

At this stage, the testing engineer should make sure that all the circuit component such as NAND gates, the D Flip-Flops, and the IR proximity sensors are working properly by testing their behaviour. After assemble the circuitry, the test engineer should verify that all the nodes are connected properly.

4.7.2. Interfacing with Beacon Unit Testing

After connecting with the AU unit, the testing engineer should perform the following:

1. Check connection, especially diode and all polarized capacitor connection.
2. Set CLEAR to high and test the output of each D flip-flop.

3. Set CLEAR to LOW and set WRITE to LOW; and test whether output is equal to input.
4. Use function generator to generate CLOCK for AU unit, and use the oscilloscope to see whether output match the input.
5. Combine AU with Beacon and test it from Step 1 to 3 again.

4.8. Software Test Plan

In this section, we describe the software test plan. The software test plan is based on testing the program we put on the microcontroller. The program is supposed to generate several signals and take the serial data as input.

4.8.1. Function Testing

At this stage test engineer should test and verify that all the individual functions described in Section 4.6 are working as desired. Test engineer should write testing code to test the behaviour to these functions. Theses tasks include the following:

1. Run sample software and to make sure all pieces are functioning..
2. Run au_clear() and test output pin.
3. Run au_write() and test output pin.
4. Run au_tx() and test weather a correct CLOCK signal is generated.
5. Combine AU with Beacon and test it from Step 1 to 3 again.

4.8.2. High Level Testing

After performing the individual functions testing as describe in the previous section. Test engineer should test the algorithm for the AU as describe in Section 4.5.1. Test engineer must verify the result obtain from the AU and make sure it is correct.

5. RF Wireless Networking Design

Our TUS runs on the IEEE 802.15.4 telecommunication protocol. Due to timing constrain, we will use Chipcon CC2420 RF development kit to handle wireless communications. This development kit includes all the hardware components required to comply with IEEE802.15.4 protocol. Hence, our focuses are in the software design instead of building the entire hardware from scratch.

In this section, we describe the RF communication in our TUS. We will briefly go over the basic knowledge of IEEE 802.15.4 protocol in terms of frame format. And we will focus on the design of our wireless communication packet design, flowchart/algorithm, and the implementation. Lastly, we will discuss the hardware/software testing plans. We should mention again that Coordinators and Beacons are identical hardware, both using CC2420 development Board (CC2420DB). The differences are in how they respond to the firmware implementation.

Lastly, we assume the reader has some basic knowledge of the IEEE 802.15.4 protocol and its terminologies. For further information, please see [3] in the referenced document.

5.1. IEEE 802.15.4 Protocol General

In this section, we define the entire frame format for our TUS. For the handshaking procedure, we will use the MAC command frame format type. For the sensor information we will use the Data frame format type. The purposes of using different frame format type is that later on in development, we can add additional features such as RF power on/off in the MAC frame type without alternating the Data frame type.

5.1.1. IEEE 802.15.4 Background

IEEE 802.15.4 is a telecommunication protocol specified for low rate wireless private area network [3]. In our TUS, this protocol is used between a Coordinator and multiple Beacons. Data that are to be transmitted through this protocol must be arranged such that it is compliant to the IEEE 802.15.4 protocol. In the next section, we simply go through the basic IEEE 802.15.4 frame format and we will use it as our guideline to create our own proprietary command. Please refer to the reference document [2] for more detailed information.

5.1.2. General Frame Format Definition

Table 5-1 shows the general MAC layer frame formats used in the IEEE 802.15.4 protocol. We will use this general MAC frame format as our basic guideline to define our frame definitions such that it will achieve all the requirements defined in the Functional Specification Document [2]. All the MAC frame formats consists of the following three basic components.

- A MHR, MAC Header, which comprises frame control, sequence number, and destination/source address information.

- A MAC payload, which contains the variable length of data information.
- A MFR, MAC Footer, which contains FCS.

Octets: 2	1	(see 7.2.2.4.1)	1	variable	2
Frame control	Sequence number	Addressing fields	Command frame identifier	Command payload	FCS
MHR			MAC payload		MFR

Table 5-1: General Frame Format

Table 5-2 shows format for frame control field (FCF). It is 16 bits in length and contains information regarding how to decode the general frame format. For example, refer to Table 5-1 the 0/2/8 bits on top of the destination/source address field describe the addressing mode that should be used, and this is achieved by the 10-11 bit and 14-15 bit in the FCF.

Bits: 0–2	3	4	5	6	7–9	10–11	12–13	14–15
Frame type	Security enabled	Frame pending	Ack. request	Intra-PAN	Reserved	Dest. addressing mode	Reserved	Source addressing mode

Table 5-2: Format for Frame Control

In IEEE 802.15.4 protocol, there are four defined frame type and they are Beacon frame type, Data frame type, Acknowledgment frame type, and MAC Command frame type. In general, our TUS used the MAC Command frame type to perform the network formation, and uses the Data frame type to transmit/receive our sensor information, and embedded Acknowledge inside our packet to confirm the transmission. The general frame format for MAC Command and Data are show in Table 5-3 and Table 5-4 respectively.

Octets: 2	1	(see 7.2.2.4.1)	1	variable	2
Frame control	Sequence number	Addressing fields	Command frame identifier	Command payload	FCS
MHR			MAC payload		MFR

Table 5-3: MAC Command Frame Format

Octets: 2	1	(see 7.2.2.2.1)	variable	2
Frame control	Sequence number	Addressing fields	Data payload	FCS
MHR			MAC payload	MFR

Table 5-4: Data Frame Format

5.2. Networking Frame Packet Design

In this section, we define the entire frame format for our TUS. These frames set the foundation of our wireless communication. For the inter-device low level communication, we will use the MAC command frame format type. For the sensor information communication, we will use the Data frame format type as defined in the protocol. As mentioned previously, the purposes of using different frame format type is that later on, we can add feature such as RF power on/off in the MAC frame type without alternating the Data frame type. In addition, it offers much more convenience in packet debugging when using the packet sniffer.

5.2.1. Coordinator Announcement Packet (Coordinator)

Coordinator Announcement Packets will be sent regularly to check if there is any newly setup SST station waiting to join the network. Table 5-5 shows the fields and the design values to be used in the Coordinator Announcement Packet. We will go through individual field and briefly describe its purpose.

Byte(s)	**1**	**2**	**1**	**2**	**2**	**2**	**1**	**1**	**1**	**2**
Field Name	FL	FCF	SN	DPAN	DA	SA	CFID	WFR	RNR	FCS
Value	0x0E	0x8843	var	0xFFFF	0xFFFF	IEEE Add	0x47	20	20	var
	MHR						**MAC Payload**			**MFR**

Table 5-5: Coordinator Announcement Packet

- **FL (Frame Length):**

  FL contains a value of 0x0E to indicate that this packet has 14 bytes in length. Please note that the length of the frame doesn't include the FL itself.

- **FCF (Frame Control Field):**

  FCF contains a value of 0x8843 which is definition in IEEE std. 802.15.4-2003. The higher byte 0x88 shows that we are using the short address mode (16bits) for both source and destination address. Hex value 0x4 shows that the destination and source PAN ID are the same with security, acknowledgement and frame pending disable. Lastly, the hex value 0x3 indicates the frame type for this command is a MAC command.

- **S# (Sequence Number):**

  Sequence number field is an 8 bit number field which specifies a unique sequence identifier for the frame.

- **DPAN (Destination Private Area Network):**

  DPAN describes the destination PAN that the packet is sending to. This field is set to 0xFFFF which indicates that this packet is a broadcasting packet, and it is sending to all devices that are within the same PAN.

- **DA (Destination Address):**

  DA contains value of 0xFFFF which indicates that the recipients are the devices in the same PAN.

- **SA (Source Address):**

  SA contains Coordinator's 16 bit or short IEEEE address.

- **CFID (Command Frame Identifier):**

  The CFID field identifies the MAC command being used in the IEEE 802.15.4 Protocol. It contains a value of 0x47 which is reserved in IEEE std. 802.15.4-2003.

- **WFR (Wait For Response):**

  This one byte field defines how much time is reserved for the SST to respond on this packet. After the packet is received, Beacon has to send out its Beacon Response Packet within four times of WFR in milliseconds. Every time slot is 4 ms and the Beacon has to pick one randomly base on the next random number range field. This filed is set to 20 which enables 20 SST to connect; hence we will allocate first 100ms for network formation.

- **RNR (Random Number Range):**

  This one byte field defines random number modular base that device should use to determine response time slot defined above. For now, we will set to 20, but in the future, one can used this field to limit number of response send from Beacon.

- **FCS (Frame Check Sequence):**

  FCS checks the validity of a transmitted frame. The receiver can used this information to find if the frame is corrupted by the medium.

A WFR and RNR field does not belong to the IEEE 802.15.4 Protocol. It is one of our own design parameter to handle the formation of our network. The other fields are required in the IEEE 802.15.4 communication.

5.2.2. Beacon Response Packet (Beacon)

When SST receives Coordinator Announcement Packet, and it has not join a network yet, SST shall send this packet as an acknowledgement to indicate a new SST is joining the network. SST keeps responding every time when it receives a Coordinator Announcement Packet until it

receives the Response Received Packet dedicated to it, which indicates that the SST has been registered with the Coordinator. Table 5-6 shows the fields and the design values to be used in the Beacon Response Packet. Most of the fields are identical to the Coordinator Announcement Packet; however, we will still go over individual field for completeness. The main difference is that this is not broadcasting, and the recipient is the Coordinator as indicated by the field Destination Address (DA).

Byte(s)	1	2	1	2	2	2	1	2
Field Name	FL	FCF	SN	DPAN	DA	SA	CFID	FCS
Value	0x0C	0x8843	var	0x2420	IEEE Add	IEEE Add	0x49	var
	MHR						MAC Payload	MFR

Table 5-6: Beacon Response Packet

- **FL (Frame Length):**

  FL contains a value of 0x0C to indicate that this packet has 12 bytes in length.

- **FCF (Frame Control Field):**

  FCF contains a value of 0x8843. Hex value of 0x88 means we are using short address mode or 16 bits address mode for both destination and source address. Hex value of 0x4 means this packet is sent to the same PAN which has value of 0x2420. Lastly the Hex value of 0x3 means that we are using the MAC Command type frame format as describe in the Table 5-4.

- **S# (Sequence Number):**

  Sequence number field is an 8 bit number field which specifies a unique sequence identifier for the frame.

- **DPAN (Destination Private Area Network):**

  DPAN describes the destination PAN that the packet is sending too, in this case we will use value of 0x2420.

- **DA (Destination Address):**

  DA contains Coordinator's 16 bit or short IEEEE address.

- **SA (Source Address):**

  SA contains Beacon's 16 bit or short IEEEE address.

- **CFID (Command Frame Identifier):**

  The CFID identifies the MAC command being used in the IEEE 802.15.4 Protocol. It contains a value of 0x49 which is reserved in IEEE std. 802.15.4-2003.

- **FCS (Frame Check Sequence):**

  FCS checks the validity of a transmitted frame. The receiver can used this information to find if the frame is corrupted by the medium.

5.2.3. Response Received Packet (Coordinator)

When Coordinator receives the Beacon Response Packet, it sends the Response Received Packet to the sender as an acknowledgement so that the sender ceases responding on Coordinator Announcement Packet. Table 5-7 shows the fields and the design values to be used in the Response Received Packet. The fields are identical to the Beacon Response Packet. The main difference is that Coordinator sends this packet.

Byte(s)	1	2	1	2	2	2	1	2
Field Name	FL	FCF	SN	DPAN	DA	SA	CFID	FCS
Value	0x0C	0x8843	var	0x2420	IEEE Add	IEEE Add	0x4a	var
	MHR						**Mac Payload**	**MFR**

Table 5-7: Response Packet

- **FL (Frame Length):**

  FL contains a value of 0x0C to indicate that this packet has 12 bytes in length.

- **FCF (Frame Control Field):**

  FCF contains a value of 0x8841. Hex value of 0x88 means we are using short address mode or 16 bits address mode for both destination and source address. Hex value of 0x4 means this packet is sent to the same PAN which has value of 0x2420. Lastly the Hex value of 0x1 means that we are using the Data frame type format as describe in the Table 5-4.

- **S# (Sequence Number):**

  Sequence number field is an eight bit number field which specifies a unique sequence identifier for the frame.

- **DPAN (Destination Private Area Network):**

  DPAN describes the destination PAN that the packet is sending too, in this case we will use value of 0x2420.

- **DA (Destination Address):**

  DA contains Beacon's 16 bit or short IEEEE address.

- **SA (Source Address):**

  SA contains Coordinator's 16 bit or short IEEEE address.

- **CFID (Command Frame Identifier):**

  The command frame identifier field identifies the MAC command being used in the IEEE 802.15.4 Protocol. It contains a value of 0x4a which is reserved in IEEE std. 802.15.4-2003.

- **FCS (Frame Check Sequence):**

  FCS checks the validity of a transmitted frame. The receiver can used this information to find if the frame is corrupted by the medium.

5.2.4. Query Packet (Coordinator)

The Query Packet starts the querying process for obtaining the seats occupancy information from all the SSTs, and it also schedules the order for the SST to reply. Table 5-8 shows the fields and the design values to be used in the Query Packet. The SD field in this packet describes the duration for a Beacon to reply, and the SSnDA field defines the order of Beacon to communicate.

Byte(s)	**1**	**2**	**1**	**2**	**2**	**2**	**1**	**1**	**variable**	**2**
Field Name	FL	FCF	SN	DPAN	DA	SA	CFID	SD	SSnDA	FCS
Value	var	0x8843	variable	0x2420	IEEE Add	IEEE Add	0x40	var	IEEE Adds	var
	MFR						**MAC Payload**			**MFR**

Table 5-8: Query Packet

- **FL (frame length) field:**

  FL contains a value of the variable length of packet size, and maximum value of the FL is 127. The MSB is not used. This field depends on the number of SSnDA (Sending Slot n Device Address) append in the MAC payload section. Each SSnDA used two bytes; therefore, maximum of the 57 address can be appended.

- **FCF (frame control field) field:**

  FCF contains a value of 0x8843. Hex value of 0x88 means we are using short address mode or 16 bits address mode for both destination and source address. Hex value of 0x4 means this packet is send to the same PAN which has value of 0x2420. Lastly the Hex value of 0x1 means that we are using the Data frame type format as describe in the Table 5-4.

- **S# (sequence number):**

  Sequence number field is an 8 bit number field which specifies a unique sequence identifier for the frame.

- **DPAN (Destination Private Area Network):**

  DPAN describes the destination PAN that the packet is sending too, in this case we will use value of 0x2420.

- **DA (Destination Address):**

  DA contains Beacon's 16 bit or short IEEEE address.

- **SA (source address):**

  SA contains sender's 64-bit or short IEEE address.

- **CFID (Command Frame Identifier):**

  The command frame identifier field identifies the MAC command being used in the IEEE 802.15.4 Protocol. It contains a value of 0x40 which is reserved in IEEE std. 802.15.4-2003.

- **SD (Sending Duration):**

  This field is a byte, and describe the duration of the timing slot. The duration can vary depend on the number of data to be transferred. Since our application sends less than 10 bytes of data per period, the Sending Duration can set to be approximately 20ms per sender

- **SSnDA (Sending Slot n Device Address):**

  This field specifies the device that can use the time slice (SSP which is defined in the next field) after the Query Packet is received. When this field contains 0xFFFF, it implies no one can use this time slice.

- **FCS (Frame Check Sequence):**

  FCS checks the validity of a transmitted frame. The receiver can used this information to find if the frame is corrupted by the medium.

5.2.5. Status Packet (Beacon)

Status Packet is sent by the Beacon to reply the Query Packet, and it replies at his designated time slot as defined in the Query Packet. Table 5-9 shows the fields and the design values to be used in the Status Packet.

Byte(s)	1	2	1	2	2	1	1	2	2
Field Name	FL	FCF	SN	DPAN	DA	SA	NS	SS	FCS
Value	0x0E	0x8841	variable	0x2420	IEEE Address	IEEE Address	var	var	var
	MHR						Mac Payload		MFR

Table 5-9: Status Packet

- **FL (frame length) field:**

  FL contains a value of the length of this packet.

- **FCF (frame control field) field:**

  FCF contains a value of 0x8841. Hex value of 0x88 means we are using short address mode or 16 bits address mode for both destination and source address. Hex value of 0x4 means this packet is send to the same PAN which has value of 0x2420. Lastly the Hex value of 0x1 means that we are using the Data frame type format as describe in the Table 5-4.

- **S# (sequence number):**

  Sequence number field is an 8 bit number field which specifies a unique sequence identifier for the frame.

- **DPAN (Destination Private Area Network):**

  DPAN describes the destination PAN that the packet is sending to, and in this case we will use value of 0x2420.

- **DA (destination address):**

  DA contains Coordinator's 16-bit or short IEEE address.

- **SA (source address):**

  SA contains Beacon's 16-bit or short IEEE address.

- **NS (Number of Seats):**

  This field is used to determine how many number of seats that are connected to one SST station. The length of the field is one byte, which allows 255 seats at maximum.

- **SS (Seat Status):**

  The length of this field can be varied from 0 up to 97 Byte, and is based on the NS field above.

- **FCS (Frame Check Sequence):**

  FCS checks the validity of a transmitted frame. The receiver can used this information to find if the frame is corrupted by the medium.

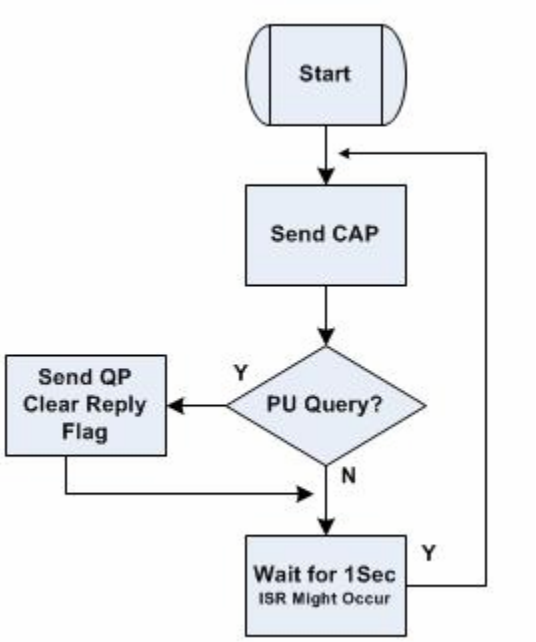
5.3. Networking Flowchart / Algorithm Design

In this section, we will discuss the algorithm or the flowchart for wireless communication in our TUS.

5.3.1. Coordinator Flowchart

Flowchart 5-1 describes the main routine for Coordinator. After starting the hardware, Coordinator will start the network formation by sending Coordinator Announcement Packet (CAP) every second. This enables the orphan SSTs to join the network. When an orphan SST receives this broadcasting packet, it will respond to Coordinator and start the network formation.

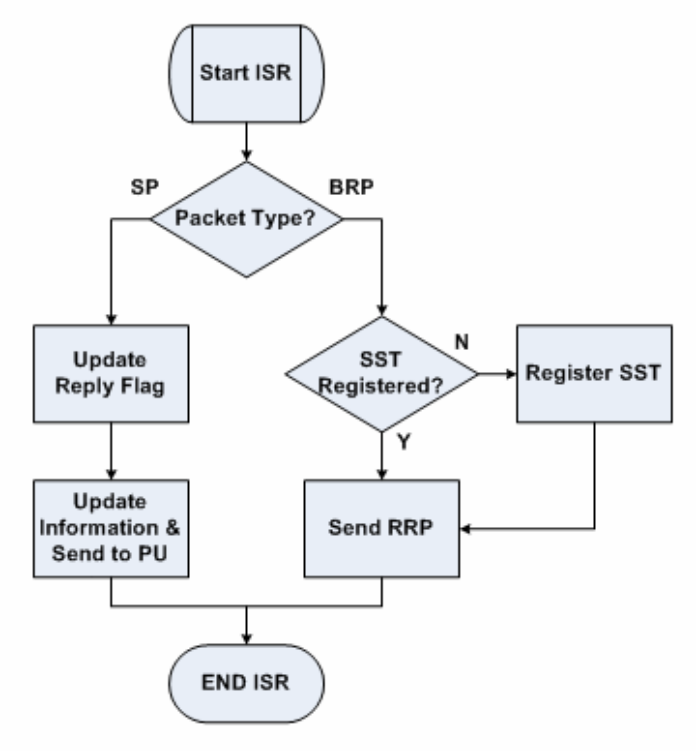
After sending the Coordinator Announcement Packet, the Coordinator should check to see if the Processing Unit (PU) would like to query the current seating occupancy information on the registered SST. If a query has been requested, the Coordinator shall send out the Query Packet (QP) which indicate the query command, and it will also include the schedule information for SSTs to reply. Lastly, the Coordinator will idle for one second before repeating this procedure once again.



Flowchart 5-1: Coordinator Main Routine Flow Chart

During the one second phase, the Coordinator will wait for an interrupts to occur to indicate it had received either a Beacon Response Packet (BRP), or Status Packet (SP) from the SST. The ISR is show on Flowchart 5-2. When the Coordinator performs the ISR, first it checks the type of the packet it received. If the received packet is a Beacon Response Packet (BRP), it will then check its SST list. If the SST has been register it simply returns the Response Received Packet (RRP) to the sender. If the SST has not been register in the SST list, it will register the sender's address and reply the RRP to tell the sender that it has been register.

Other the hand, if the receive packet is a Status Packet (SP), which indicate the particular SST station seating occupancy information. It will then transmit such information to the Processing Unit via the RS232 protocol, and the PU will update the screen for displaying.

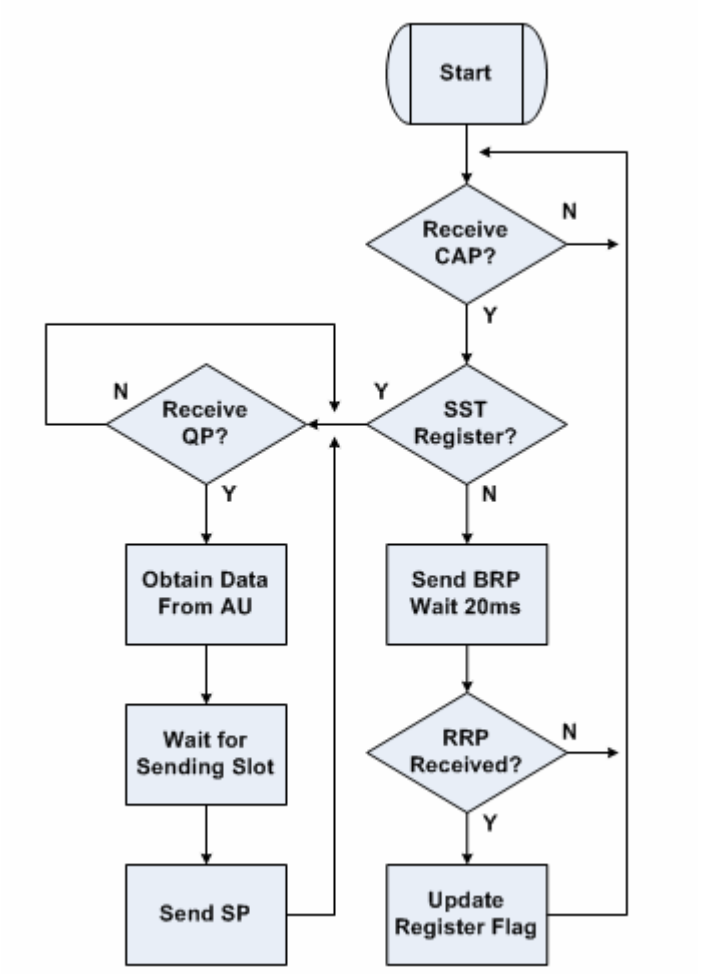


Flowchart 5-2: Coordinator ISR Flow Chart

5.3.2. Beacon Flowchart

Flowchart 5-2 shows the flowchart for Beacon. After starting the hardware, Beacon will wait on the Coordinator Announcement Packet (CAP) to register its address with Coordinator. If Beacon receives the CAP, it will respond with the Beacon Response Packet to indicate that this particular SST would like to join a network. Beacon then waits for about 100ms to see if Coordinator replies the Response Received Packet (RRP). If Coordinator replies with RRP, that means the Coordinator have registered the Beacon's address and will set the register flag to true. After setting the flag, this indicates that this SST station will be capable to start transmitting the seat occupancy information. However, if Beacon does not receive the RRP from Coordinator, it will return to the beginning of its procedure and wait for CAP, and try to join the network again.

When a SST is registered, it then waits for the Query Packet (QP) from the Coordinator which indicates that Processing Unit is querying for the seat occupancy information. If this packet is received, Beacon will obtain the seats occupancy information from the Acquisition Unit, and wait for its sending time slot for delivering. The Beacon then enters an infinite loop which it waits on Query Packet, obtain seats occupancy information from Acquisition Unit and send it at the scheduled time slot.



Flowchart 5-3: Beacon Flow Chart

5.3.3. Network Timing

In the wireless networking environment, it is important to know that the packet may collide with another packet, which results in data delivery failure. With our proprietary simple private area network scheduling implementation, we believe that packet collision will less likely to occur.

Figure 5-1 shows the sequence of events in wireless medium. Firstly, Coordinator will have approximately 50ms duration to send the Coordinator Announcement Packet (CAP). Second, TUS allocates about 150ms for orphan SSTs to join the network. Third, Coordinator will send query Packet (QP) if is required. Lastly, if Coordinator is querying the seat occupancy information, the SST shall respond only at its designated time slot. When Coordinator receives the seat occupancy status, it will notify the PU. After updating all the seat information, the system will repeat those actions again.

By allocating time slot for individual SSTs and Coordinator, we can avoid packet collision which improves the reliability of information transfer. Please note that the entire looping period and the transmission duration are subjected to change according to network size.

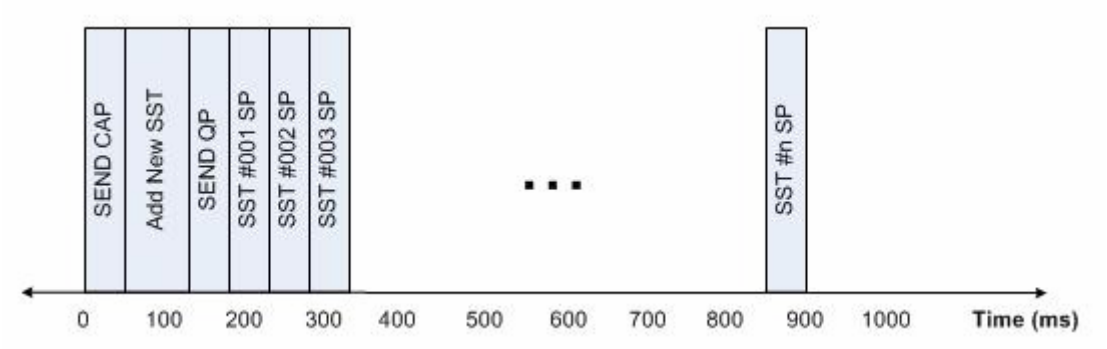


Figure 5-1: Network Timing

5.4. Firmware Design

In this section, we listed the higher level function calls for the Coordinator, and the Beacon. In addition, we designed a class for Coordinator to maintain the SST List.

5.4.1. Basic Functions for Coordinator

These are the basic functions for Coordinator in general.

- **Send_CAP();**

  This function shall enable the program to send a Send Coordinator Announcement Packet. The software engineer shall make sure all the packet is design according to the Section 5.2.1.

- **Send_RRP();**

  This function shall enable the program to send a Response Received Packet. The software engineer shall make sure all the packet is design according to the Section 5.2.3.

- **Send_QP();**

  This function shall enable the program to send a Query Packet. The software engineer shall make sure all the packet is design according to the Section 5.2.4.

5.4.2. Basic Functions for Beacon

There are the basic functions for Beacon in general.

- **Send_BRP();**

  This function shall enable the program to send a Beacon Response Packet. The software engineer shall make sure all the packet is design according to the Section 5.2.2.

- **Send_SP();**

  This function shall enable the program to send a Status Packet. The software engineer shall make sure all the packet is design according to the Section 5.2.5.

- **GET_SOS();**

  This function will get the current seat occupancy information for that SST station. The detailed algorithm is described in the SST section.

5.4.3. Coordinator SST List Object / Class Design

The follow are the members and the prototypes need for the SST List class. We will briefly describe the functionality of individual members and the function prototype.

```
Struct SSTList
{
    Char SSTAddress [32];  // SST address
    Char NS [32];          // number of seats per SST
    Char SOS [64];         // seat occupancy status, assume a SST have maximum of 16 seats
    Bool Replied [32];     // flag to determine if SST have replied
}
```

- **Add(char SSTAddress[2]);**

  The function adds a new SST station to the SST list.

- **Delete(char SSTAddress[2]);**

  The function deletes a SST station from the SST list.

- **Update(char SSTAddress[2], char SOS);**

  The function updates the seats occupancy information for a particular SST station in the SST List.

- **Find(char SSTAddress[2]);**

  The function tries to find if the given SST address have or haven't registered in the SST List. The return of this function call will either true or false.

- **GetSchedule();**

  The function generates a list of SST Addresses which shown the order of SST reply. This function is used by the Send_QP() function describe in the Section 5.4.1.

- **ClearReply();**

  The function sets the entire Replied variable in the SST List to false.

- **SetReply(char SSTAddress[2]);**

  The function sets given SST Address's Replied variable in the SST List to true.

5.5. Hardware Test Plan

In this section, we include three sections for our hardware testing plan.

5.5.1. Component Connection Verification

The functionality of the component level verification is almost always guaranteed to work since is a ready to go development board. We assume that the hardware component part shall function as expected. However, we will use oscilloscope, and digital multi-meter to debug if problems arise.

5.5.2. DC Power Verification

At this stage, the testing engineer should perform the following:

1. Apply correct DC voltage by transformer or the 9V battery to power up the board
2. Verify the correct DC voltage at all the power nodes.
3. Verify that the functionality of the board.

5.5.3. CC2400EB and CC2420EM for ZigBee Packet Sniffing

At this stage, we will use the CC2400EB and CC2420EM to sniff the network traffic on the IEEE 802.15.4 protocol. It can also be used as the IEEE 802.15.4 compliant packed parser as well. Figure 5-2 shows this device. For detail operational instructions please refer to the CC2420 DK User Manual [4].

Figure 5-2: CC2400EB and CC2420EM

5.6. Software Test Plan

In this section, we briefly go over the software testing plan for our TUS.

5.6.1. Chipcon Packet Sniffer Software

For the wireless communication testing, we will use the Chipcon Packet Sniffer Software for IEEE 802.15.4 to verify the packet sent by Coordinator and the Beacon. In addition, we will use the sniffer to analyze the network scheduling. Figure 5-3 shows the software layout of this program.

Moreover, we will use the Chipcon Packet Sniffer Software to verify our network scheduling as describe in the Section 5.3.3.

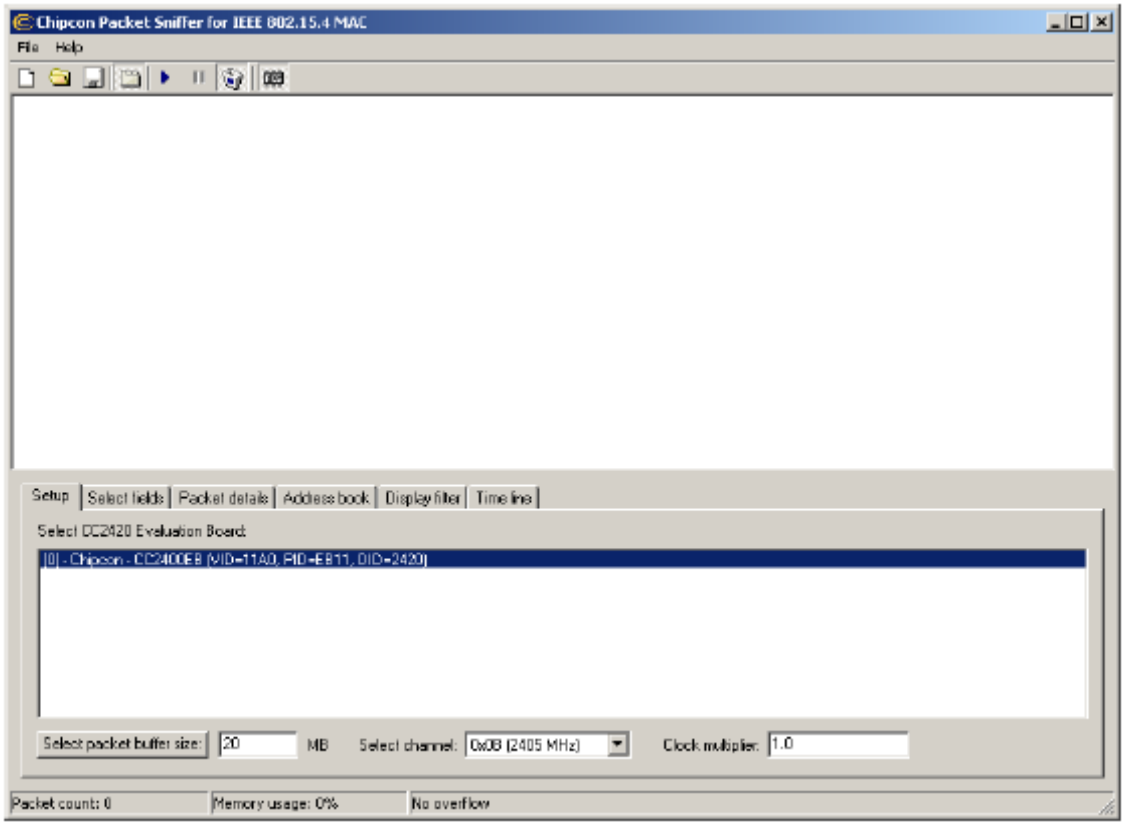


Figure 5-3: Chipcon Packet Sniffer Software

5.6.2. Networking Formation Testing

We will write test programs to make sure that the formation part of our RF communication is indeed operate as we desired. We will load the connected SST devices information via the

Windows HyperTerminal Program, which is a software enable communication in the RS232 protocol, for visual displaying and confirmation. In additional, we will use the onboard four led to debug, since we do not have a LCD to print the texts. All our development code will be load via the WinAVR Programmer using the RS232 protocol onto the onboard flash memory.

5.6.3. SST List Testing

We shall verify the design in the Section 5.4.3 (SST List Class) function by function by writing simply testing routines to retrieve/verify the information in the SST List, and we will show this information via the HyperTerminal or the onboard LEDs.

6. SSM Design

In this section, we describe the fundamental design approaches for the SSM. The first major component is the software interface which the moviegoer will see. This is important as we need to successfully communicate useful information to the moviegoers without causing ambiguity. The second major component is the need to successfully update seating information reliably. The software for SSM must be able to communicate via serial port to acquire the seating information transmitted by SST.

6.1. Display Interface/ Layout Design

The display unit will obtain seating information from the Processing Unit and display the information which indicates the seat's occupancy on the screen. This information will be updated with a 2 seconds period. In addition, the display units will also display advertisements both in still image or movie formats and movie show times along with seating information. All of the above processes will be done by the display program which is loaded in the Processing Unit and the program output will be on the display unit. Due to time constraint, the working prototype will be a typical 14 inch laptop as the display unit along with the laptop itself as the processing unit. Further enhancements can be made by separating the unit into larger display along with smaller embedded system.

6.1.1. Display Program Design

The display program will be a Windows interface program, which operates under the Windows operating system environment. The program will be written using Microsoft Visual Studio 2005 C# developing environment. We will use C# to develop the display program because we can take full advantage of C language along with other features such as garbage collection for convenience. Also, extensive resources such as RS232 interfacing in C# are readily available, thus can shorten our development time slightly.

6.1.2. Display Program Interface Design

The display program layout consists of the items shown in Figure 6-1. We will briefly go through individual section.

1. **The Main Display:** Displays the seating occupancy using the layout of the theater
2. **The Trailer/Advertisement Display:** Displaying movie trailers or various advertisements in various movie formats
3. **Movie Show Times:** Displaying the movie show times for current theater screen
4. **Static Advertisement Display Area 1:** Displays pictorial advertisements. The pictorial advertisements will change periodically.

5. **Static Advertisement Display Area 2:** Displays pictorial advertisements. The pictorial advertisements will change periodically.

In the following subsections, these items will be discussed in greater detail.

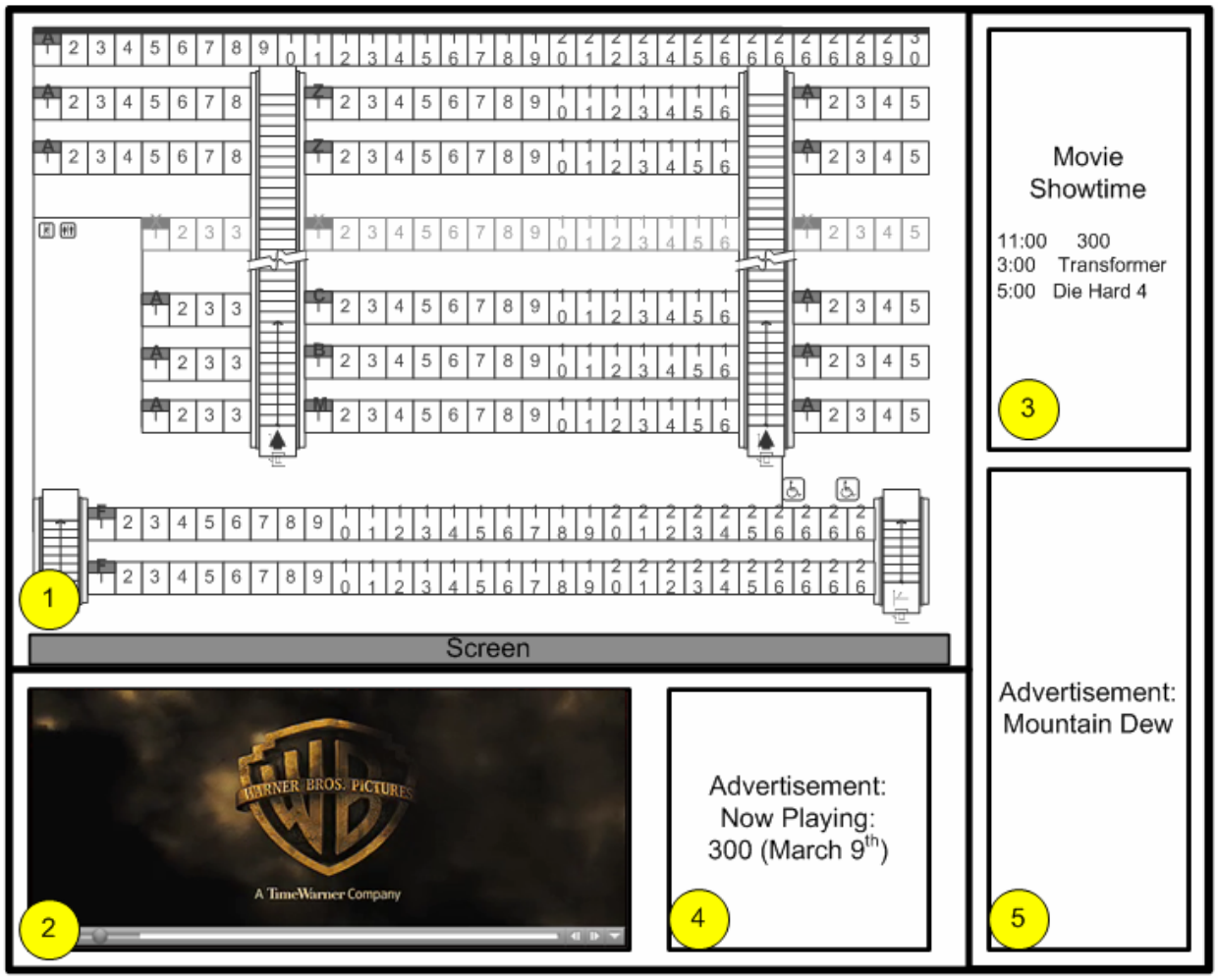


Figure 6-1: Overall Display Layout

6.1.3. The Main Display

The Main Display panel displays the seating occupancy on a layout of the theater. If the seat is occupied, the seat position on the layout will be marked in red. If the seat is not occupied, then it will be marked as green. There will be landmarks shown on the layout of the theater to help moviegoers navigate. These landmarks are:

- Movie Screen at the bottom of the layout
- Entrance/Exits locations at the left and right of the theater
- Stairs between seats

On the layout, those landmarks will be easy to be identified because they will be drawn in detail instead of simple blocks with text names on them. The typical main display screen should look like that presented in Figure 6-2.

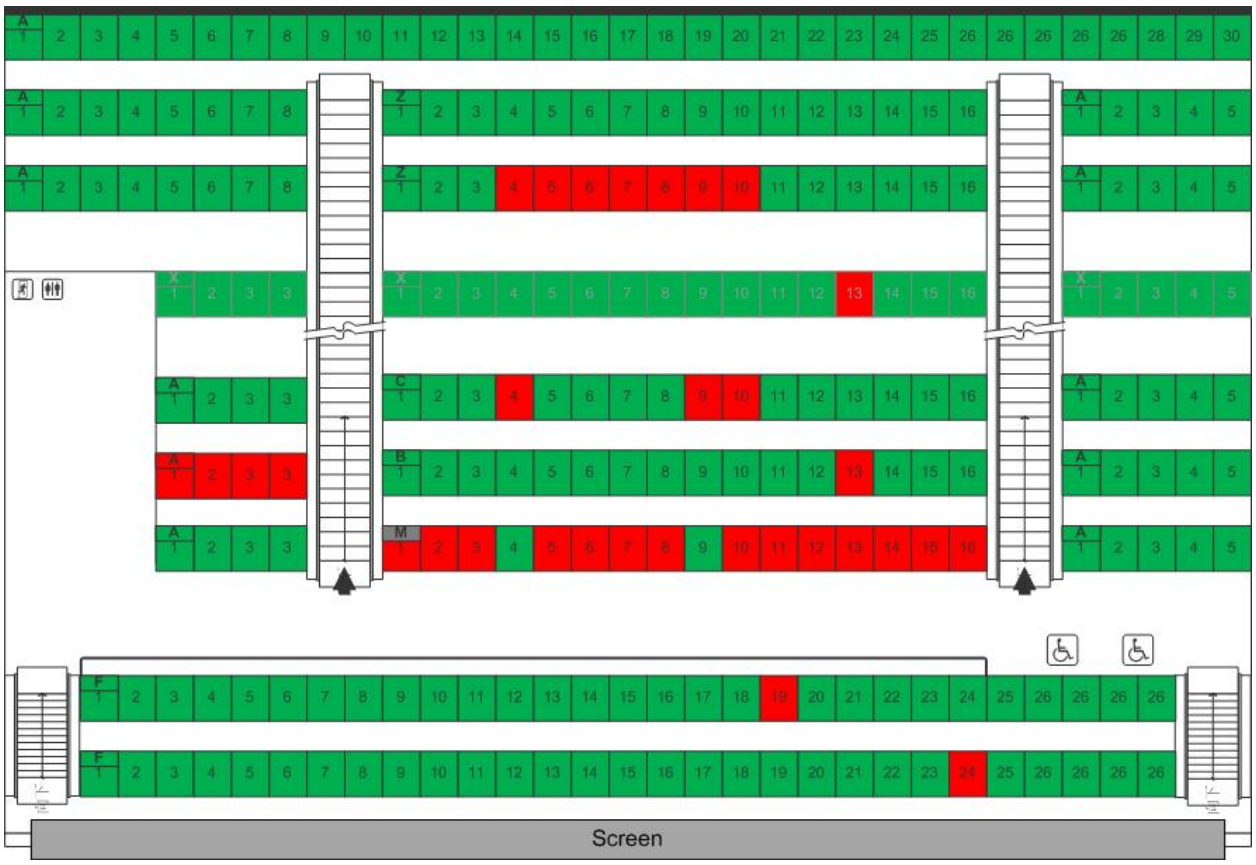


Figure 6-2: Main Display Theatre Layout with Empty and Occupied Seats

Each row will be governed by a Beacon, where it is responsible for collecting data from all 16 seats. Since each Beacon will have its distinct address, we then need to map these address to the corresponding row of our interface to make sure that proper rows are updated.

Before transmission, the data needs to be packaged to a uniform standard. The format we will use consists of the assigned beacon address in the beginning, followed by the seat occupancy data. Seat occupancy data are then presented continuously, seat by seat, from the predefined location. The typical data segment in the packet should look like that in Table 6-1.

BN	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
10	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0

Table 6-1: Typical Data in the Packet, BN represents the Beacon Number

From the above example Table 6-1, the data from Beacon No.10 is indicating the seat No.2, 6, and 8 in the corresponding row are occupied. This information will be relayed to the main display screen. In which the main display screen will react correspondingly to this data and change the seating status indicated on the screen.

The seating status display will be updated every 10 seconds by refreshing the main display screen to check the seating status displayed.

6.1.4. The Trailer and Advertisement Display

The Trailer and Advertisement display will support broadcasting movies in various formats. Therefore, a primitive media player will be integrated into the display program. This media player will have the basic functions:

- Open Files
- Play
- Pause
- Stop

This media player plays the movie trailer or advertisement clips loaded in a designated folder, and will have a play list of movie trailer or advertisement clips. The play list will loop until the Stop command is issued.

6.1.5. The Movie Show Time Display

The movie show time display will read text file inputs from a designated file. Those text file inputs contain the movie show time of the day which can be easily changed by changing the content of the text files. There will also be a clock which indicates the current time. The movie display time may be removed once that particular movie started. This is done by comparing the time read from the text file to the actual time from the clock.

6.1.6. Static Advertisement Display

The static advertisement display will read image format files from a designated folder. The static advertisements will change to another after 60 seconds. In order to do so, an internal timer will be implemented on the static advertisement display section which resets itself after 60 seconds. Upon the reset, another image file from the designated folder will be loaded onto the display section. This is similar to a play list for media files. This process will keep looping until a stop command is issued.

6.2. PU/Coordinator Serial Communication Design

RS-232 protocol for serial binary interconnection communication will be used to acquire data from Coordinator. In the TUS, RS-232 is utilized between the AU to Beacon and Coordinator to

PU. Through out this document we will us our proprietary data frame format defined in the next section to establish the communication between devices.

In this section, we will design a set of general frame format definition for device that will use the RS232 serial protocol.

6.2.1. General Frame Format Definition

In our serial communication, frames consist of Command Type, Frame Length, and Payload as illustrated in Table 6-2.

Byte(s)	**2**	**1**	**variable**	**1**
Field Name	SFD	Command	SOI	End
	Header		Payload	Footer

Table 6-2: General Frame Format for Serial Communication

- Header, which describe the properties of the frame.
- Payload, which contains variables length of information.
- Footer, which describe the end of a frame

6.2.2. Coordinator/PU Communication Frame Definition

For the communication frame between Coordinator and PU, it is defined in the same format as the general communication frame where there are Header, Payload, and Footer presents. All of these have their own frame length corresponding to data each frame carries. For the Coordinator/PU communication frame, the sub frames inside the general frame definition are further specified as shown in Table 6-3.

Byte(s)	**2**	**1**	**2**	**1**	**variable**	**1**
Field Name	SFD	Command	Beacon ID	NS	SOI	End
	Header		Payload			Footer

Table 6-3: Detailed Frame Format for Serial Communication

The header frame (3 bytes) consisted of 2 sub frames:

- **SFD (Start Frame Delimiter):**

  The segment marks the end of preamble and beginning of the frame. The specific SFD will be defined for PU to identify for signal validation purpose.

- **Command:**

  This segment consists of MAC commands, such as destination and source MAC address. The specific Command will be defined for PU to identify for signal validation purpose.

The Payload frame (4~7 bytes) consisted of 3 sub frames:

- **The Beacon ID:**

  This segment is used to mark the identification code for the specific beacon. The beacon identification codes are pre-defined specifically for each beacon so they can be matched from a pre-defined list. In addition, since the beacon is stationary in each row, the software can easily acknowledge the beacon location by the beacon ID.

- **Number of Seats:**

  This segment contains the number of seat occupancy data the beacon transmits

- **SOI (Seat Occupancy Information):**

  This segment is the Seat Occupancy Data in binary format. The MSB of the bytes represents the first seat in the row and the LSB of the bytes represents the last seat in the row. “0” means the seat is not occupied and “1” means otherwise. For Example, if SOI has a value of 0xFF, this means that all the seats in that particular SST station are all occupied.

The Footer (1 byte) frame is the indication of end of the data frame.

6.2.3. Coordinator/PU Communication Synchronization

In order for PU to communicate with the Coordinator, the interrupt signal will be queried and received by the Coordinator to confirm there is an incoming data frame signal. If there are no interrupt signal presents, the PU will re-query the Coordinator for interrupt signals. If there is no signal present within 2 seconds, the PU will keep querying the Coordinator until there is an interrupt. Once the interrupt is present, the data frame signal will be sent from Coordinator to PU.

The data frame will be temporarily stored in a buffer array. The size of the buffer array will match the maximum frame length of the data frame. In this case, the maximum size will be 10 bytes of memory. Because RS232 is a serial port, which means the incoming data will be serialized, hence one letter of data can be stored at one time. Therefore, once the incoming data started to pour in, the first incoming byte will be stored into the first buffer array cell. The second byte of incoming data will be stored into the second buffer array cell and so on. This is demonstrated in Figure 6-3.

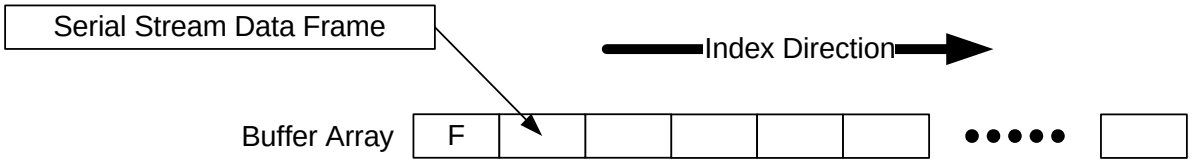


Figure 6-3: Data Storage in Buffer Array when Buffer Array Empty

Once the buffer array is full, the array index will be reset so that the incoming data will overwrite the first buffer array cell as showing in the Figure 6-4.

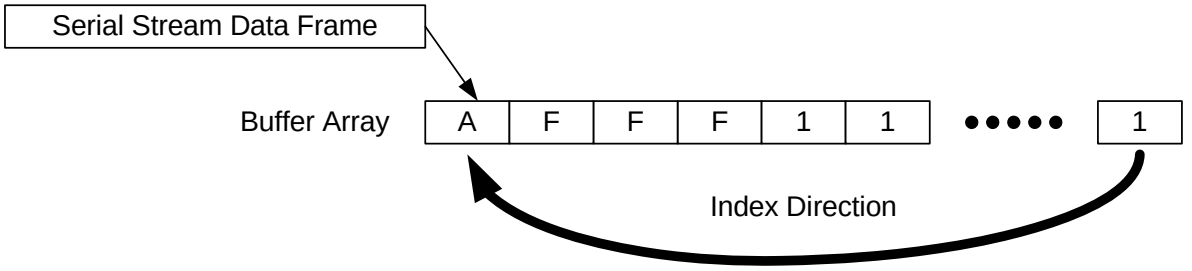


Figure 6-4: Data Storage in Buffer Array when Buffer Array Filled

There will be another temporary array which is used for storing the valid data for PU to process. The PU will test the validity of the data by comparing the information in the header sections (SFD and Command) against the pre-defined parameters to acknowledge that the signal contains proper seating information. If the parameters in both SFD and Command match the pre-defined parameters, then the data is marked as valid for processing. Figure 6-5 shows such process.

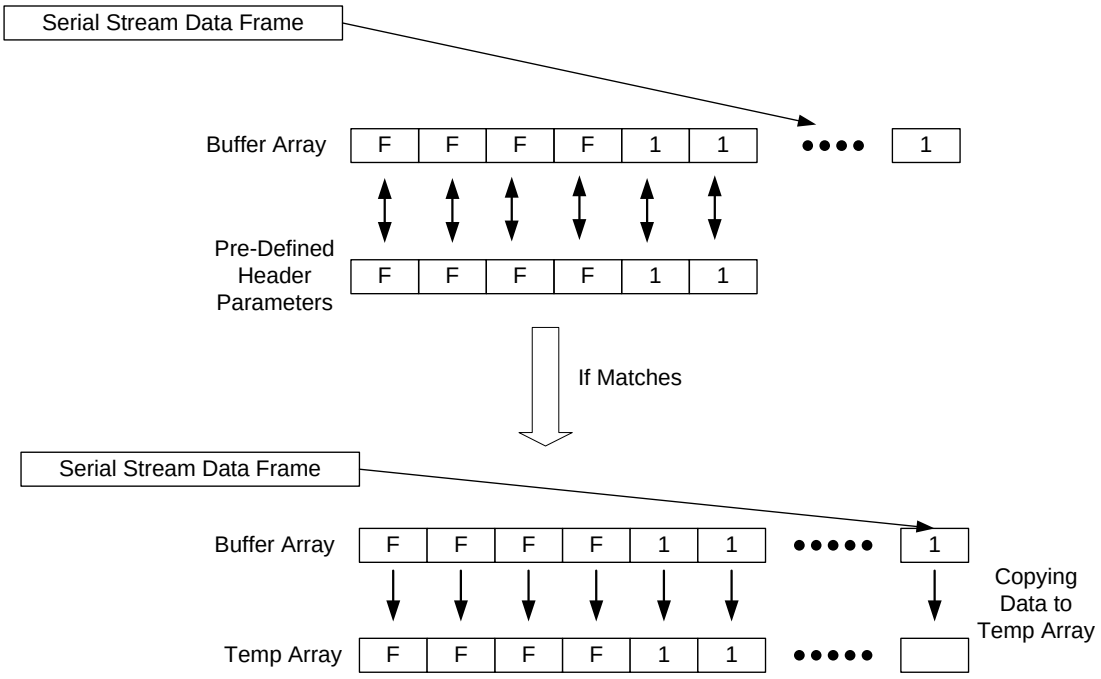


Figure 6-5: Matching the Data in Buffer Array to Predetermined Sequence

Once the temporary array is filled up, the PU will start to decode process to interpret the seating information in the data frame.

6.3. Application Software Design

In this section we discuss the interface software in a background perspective as the moviegoers will not explicitly see this part of software in work. Specifically, this part of software focuses on how we construct the interface.

6.3.1. User Interface Object/Class Design

Majority of the interface design will be done using existing GUI types in the integrated development environment. We shall omit their implementation here since it is fairly standard practice.

Our primary concern is the updating of seating information, which will require new classes and objects defined to meet our need. A Beacon class will first be defined to hold attributes such as beacon ID, number of seats on this beacon and the seating status of this beacon.

```
class Beacon
{
    char  beaconed[4];
    short int numOfSeats;
    int beaconStatus[16];
}
```

During handshaking at startup, SSM will also need to manage a list of SST to keep track of the seats for update. To do so, we will require a list and need the following functions.

- **add (beacon);**

  The function adds beacon to the current list and maps to the appropriate predefined display location

- **find (beacon);**

  The function tries to find the beacon in the managed list, if available then returns 0 else returns 1

- **edit (beacon, numOfSeats, seatStatus);**

  The edit function allows program to update information with the associated beacon

Additional functions will be required to decode and update the display screen.

- **decodeSeatingStaus();**
  The decoding function will be used to decode information from serial port
  The detailed flow chart of decoding is discussed in Section 6.5
- **updateScreen();**
  The update function is required to update the display when processing and decoding is complete

6.3.2. USART Interface Object/Class Design

Due to popularity of RS232 devices, many libraries of various programming language has been developed and available freely. We will be using such free library in order to shorten development time.

6.3.3. Media File Interface Object/Class Design

The media player used for movie trailer will not be designed by us. Instead we will use the Windows Media Player SDK freely available to developer. The advantage of using embedded Windows Media Player is to reduce our workload and reduce complexity to the program. We can also take advantage of the multi-format capabilities of Windows Media Player to eliminate the need of using specific video format.

6.3.4. Movie Schedule Object/Class Design

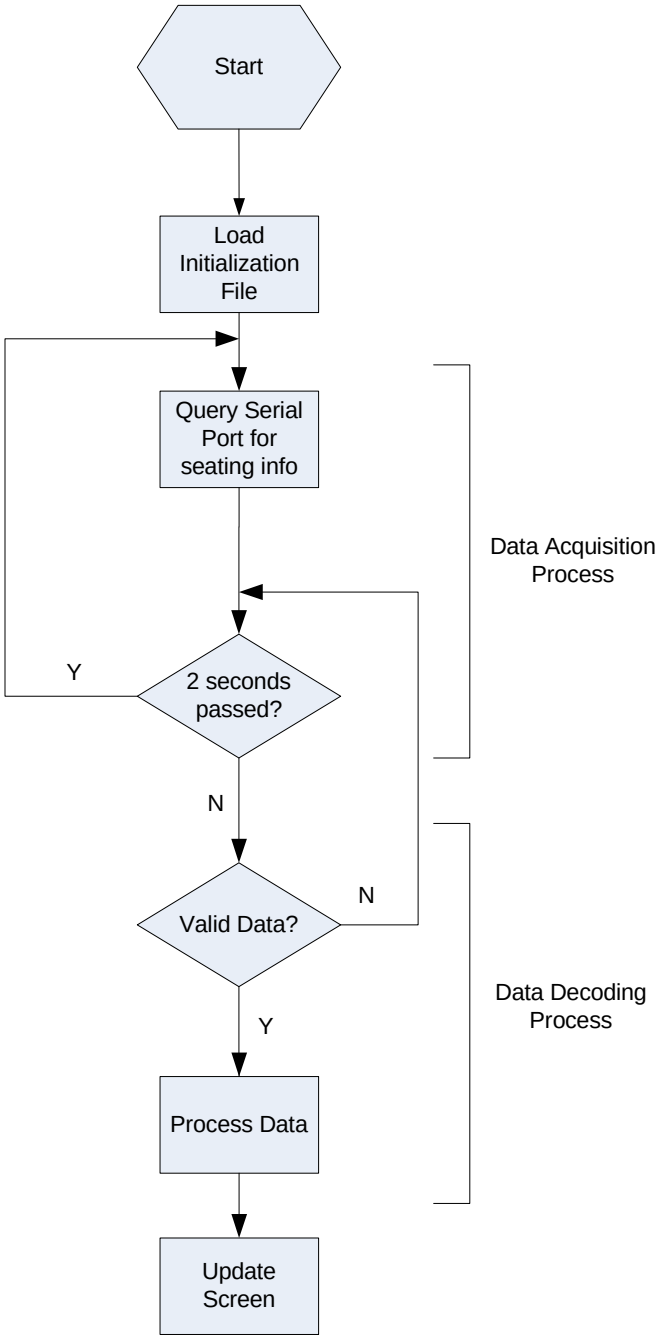
Movie schedules will be placed in a text file and loaded when requested. The easiest approach is to use the approach similar to initialization files, where settings are read during update.

6.4. Flow Chart / Algorithm

In this section, the flow charts and algorithms of the application software will be described in detail. Starting from overall process, then the details between each process will be examined in greater details.

6.4.1. Overall Process

First, when the software is starting, it will load the initialization files to initialize the peripheral functions of the program. These peripheral functions are for advertisement display purpose. These advertisements will be displayed in cycling loop. The cycling times and conditions are set in our preferences file. Therefore, loading those preferences file will ensure the correct display and looping process of advertisements function correctly. After finishing the initialization of peripheral functions, the program will query the RS232 serial ports for seating information received by the Coordinator connected in RS232 serial port. The software will wait for 2 seconds for interrupt signal. If there is no interrupt signal, then it will continue to wait every 2 seconds until an interrupt signal is received. Once an interrupt is received within the 2 seconds period, the data from RS232 serial port will be stored in a buffer. Then the software will read the data from the buffer and begin to decode and process the data. Once the data is decoded and processed, the display screen of the software will update according to the data received. The flowchart below shown in Flowchart 6-1 shows the process.

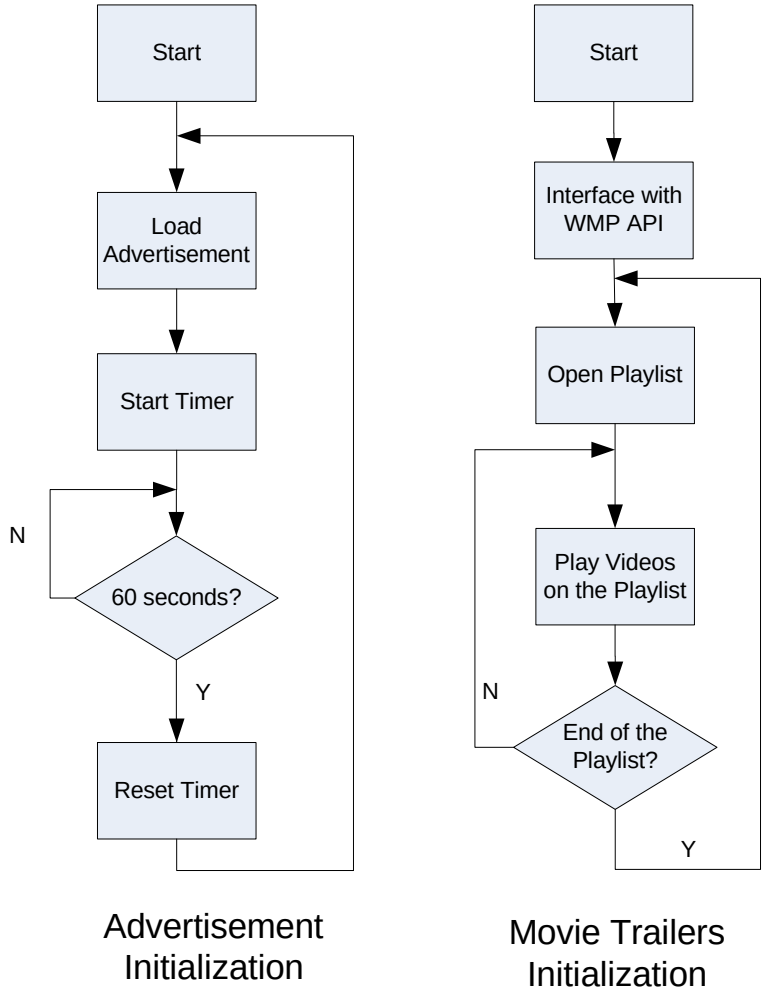


Flowchart 6-1: The Overall Process of the Display Program

The following sections will discuss the initialization process, data confirmation-storage process, and data decoding process in greater details

6.4.2. Peripheral Function Initialization Process

The peripheral functions of this software are the following: Displaying static advertisement, displaying movie trailers/video advertisement, and displaying movie time. First, for the displaying static advertisement function, the software will load the preferences file which includes the locations of static advertisement images, and then a timer will start. Once the timer reaches 60 seconds, it will reset and another static advertisement image file will be loaded. Secondly, as for displaying video files, the locations of those video files and the video play list will also be included in the preferences file. Once the locations of those video files are acknowledged, the software will interface with windows media player API to play the play list in a continuous loop. Third, for the move time display, after loading the initialization file, the software will know where to look up the pre-defined movie times in a text file. The software will read off this text file and display its contents. The process flow for displaying static advertisement and video files are shown in Flowchart 6-2.



Flowchart 6-2: Initialization Procedures for Advertisement and Movie Trailers

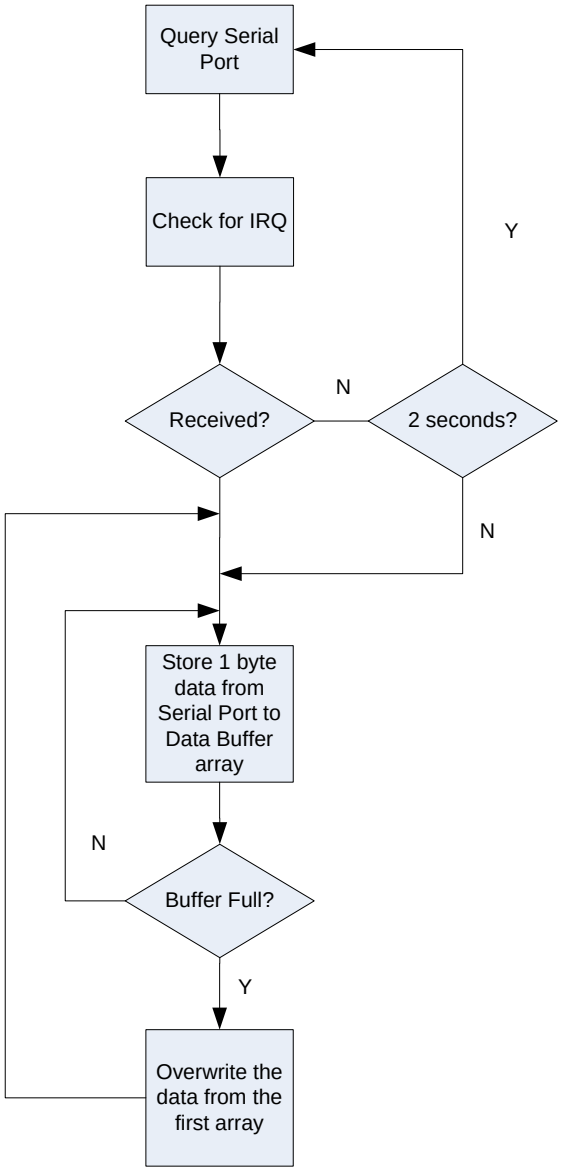
6.4.3. Data Acquisition Process

For the data acquisition process, when the software queries the RS232 serial ports for data, it will wait for an interrupt signal to know if the data is ready to be received or not. If there is no interrupt signal present within 2 seconds, the software will query the serial port again. If there are interrupt signals present, the serial data from the serial port will be stored into a buffer array 1 byte at the time. The process will repeat itself until there are no interrupt signals present. The buffer array is shown in Table 6-4.

2 Byte	1 byte	2 byte	1 byte	1~3 Byte	1 Byte
SFD	Command	Beacon ID	No. of Seats	Data	End
Header		Payload		Footer	

Table 6-4: Buffer Array Format

Since we are dealing with serial data, and if the buffer array is full, then the buffer will begin to overwrite the data stored in front of the buffer array. The data decode process function reading from the buffer array will be fast enough to read off the data due to the small data size. The process flow is shown in Flowchart 6-3.



Flowchart 6-3: Data Acquisition Process

6.4.4. Data Decode Process

After the buffer array started to be fill with data, the data decode function will read the header section of the buffer array to determine whether or not it is the seating information data. If the first 2 bytes SFD is “FFFF” and the following 1 bye is “11”, then the data is marked as valid seating information. If not, then the program will wait idle until footer is met, indicating the end of this particular data stream. After the footer, the program will repeat the validation process for the data stream to determine the validity of the next data segment. Once the signal is validated, the payload section will be decoded into three separate subsections: Beacon ID (2 byte), Number

of seats (1 byte), and seating occupancy data (1~3 bytes). An example seating occupancy data is shown in Table 6-5.

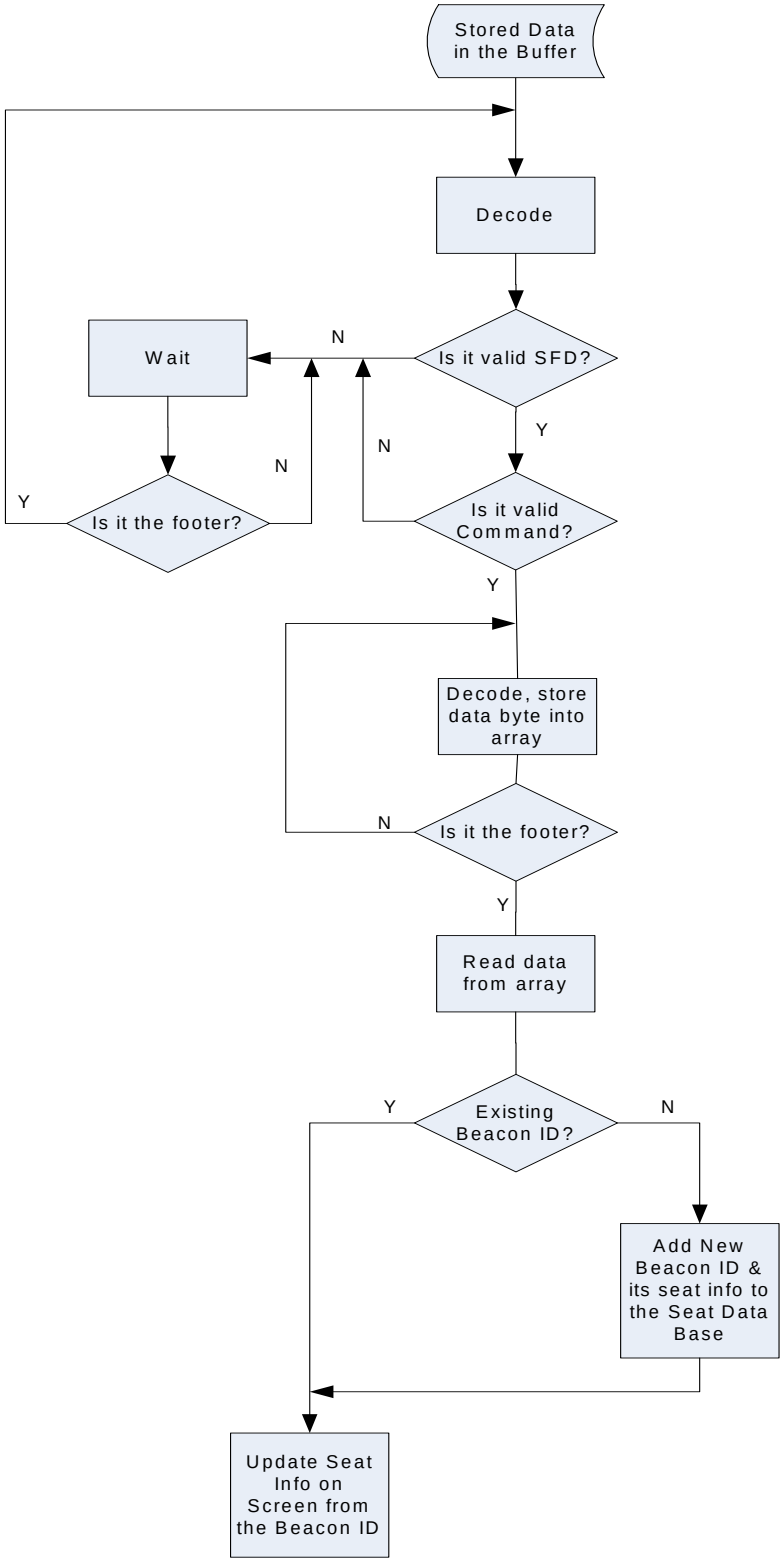
Seat No.	1	2	3	4	5	6	7	8	9	10
Occupied?	0	0	1	0	1	1	0	0	0	0

Table 6-5: Example of Seat Occupancy Data Format

The colored part is the actual data stream in the data subsection of payload section. The “0” indicates the seat is unoccupied and “1”indicates the seat is occupied. The 3 subsection of the definite data will be stored into another array for functions to access. Once the decode process finished, the software will check whether or not the data stream ends by checking the footer.

After the decode process ended, the software will match the beacon ID from the data stream to the pre-defined list of beacon ID’s. The pre-defined list of beacon ID’s are stored in a text file which the software will read from it. If they match, then the display screen of the software will be updated by matching the seat occupancy data and beacon ID to the location representations on the display screen. If not, then the new beacon ID will be added to the pre-defined list of beacon ID, and then the new beacon ID, Numbers of seat, and seat occupancy data will also be used to update the data on the display screen.

The data decode flow process are shown in Flowchart 6-4.



Flowchart 6-4: Data Decoding Procedure

6.5. Hardware Test Plan

Communication between Coordinator to processing unit is focused in this section.

6.5.1. Inter-Hardware Communication Testing

To test the communication between Coordinator and SST, we will require another operational transmitting SST. Specifically, we will transmit several variations of predetermined packets, and upon receiving at the Coordinator, we will immediately grab the data to see if we receive what we are expecting. Testing at this stage will capture bugs that occur between SST and SSM transmission. Extensive testing will allow us to make sure our design is reliable and interferences do not lead to incorrect information.

6.6. Hardware Test Plan

Tests will be divided into two sections, where we first test for the GUI display and then test for the functionality.

6.6.1. GUI Testing

Low level GUI testing will be conducted to test if any media player and advertisement files are displayed and updated at correct interval. General display will be tested and it is expected that any bugs related to the GUI will occur at this stage.

6.6.2. High Level Functional Testing

Functional testing will be conducted to test software behaviors. We will focus on area such as movie playing in media player, and if the play lists are followed. Advertisement cycling will also be tested, to make sure all advertisements are cycled in correct time interval. Last but not least, seating information update will be tested. Specifically, we plan to write test scripts which send predetermined seating information to the serial port, and when received by SSM, the interface should display seating information for comparison. Such test can eliminate bugs related to the software interface and can be tested independent to any SST.

7. Conclusion

Design specifications for the *Theater Ushering System* has been outlined and discussed. These specifications have been carefully planned to ensure that our system will meet the performance previously defined. This document will be regularly updated to ensure accurate information. Working prototype is targeted for demonstration in April 2007, and upon completion, we will evaluate if additional features needs to be implemented.

8. Reference

Referenced documents can be found in project file of U-Nexus Website.

[1] Chan, D. Lee, G. Wang, B. Wang, E. *Proposal for Theater Ushering System*. Project File. Simon Fraser University. 2007

[2] Chan, D. Lee, G. Wang, B. Wang, E. *Functional Specification for Theater Ushering System.* Project File. Simon Fraser University. 2007

[3] IEEE Computer Society, *IEEE 802.11.4-2003 Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)* The Institute of Electrical and Electronics Engineer, Inc. 2003

[4] Chipcon RF Staff. *User Manual SmartRF ® CC2420 ZigBee DK Development Kit. Rev. 1.0.* Chipcon AS. 2004

