Implementation of the EPC Class 1 Gen 2 Protocol by Miller -2 coding for the WISP Platform

TUTON CHANDRA MALLICK
Implementation of the EPC Class 1 Gen 2 Protocol by

Miller -2 coding for the WISP Platform

MASTER OF SCIENCE THESIS

BY

TUTON CHANDRA MALLICK

Co-Supervisor : Li Xie
Supervisor : DR. Fredrik Jonsson
Examiner : Dr. Prof. Li-Rong Zheng
Wireless Identification and Sensing Platform (WISP) is a battery free and fully software defined Radio Frequency Identification (RFID) tag. WISP is an open source architecture that allows exploration to integrate with several sensors and other devices. It has a 16 bit ultra-low power microcontroller that can be used for the computation and sensing purpose. The power harvesting unit of the WISP is capable to exclusively drive the sensor that is integrated with the WISP from the RF energy. However the firmware of WISP is not fully compatible with Electronic Product Code (EPC) class 1 Generation 2 protocol.

The thesis work focus on the compatibility of the WISP according to the EPC class 1 Gen 2 protocol. The communication link is successfully implemented between the WISP and the UHF reader with Miller-2 encoding. The proper handshaking signals (Query, RN16, Req_RN, ACK and ACCESS) for the communication between the WISP and the reader is implemented. The comparative discussions about different EPC protocols are also discussed in this thesis.
ACKNOWLEDGEMENT

I would like to first thank to The Royal Institute of Technology (KTH) and my department System on Chip Design. I am really grateful to the IPACK center for giving me the opportunity and the facility to carry my thesis work. That was very nice and amazing experience to work with IPACK center.

I would like to thank my supervisor Dr. Fredrik Jonsson from the bottom of my heart for his excellence supervision to my work and for his inspiration to do my work. I would like to thank my Co-Supervisor Li Xie for her suggestion. This work would not be finished without the help of those two persons.

I would also like to thank Ramon, another master thesis student, for his cooperative mind about sharing the knowledge.

I would also like to thank to my parents, my sisters and my brother for giving me inspiration about my study. I would like to thank all of my friends whom I met in KTH.
# CONTENTS

Abstract .......................................................................................................................... 1

ACKNOWLEDGEMENT ................................................................................................. 2

Contents .......................................................................................................................... 3

LIST of FIGURES ........................................................................................................... 5

CHAPTER 1 ....................................................................................................................... 6

INTRODUCTION ............................................................................................................... 6

1.1 Background ............................................................................................................... 7

1.2 Motivation ................................................................................................................ 7

1.3 Goals ....................................................................................................................... 8

CHAPTER 2 ....................................................................................................................... 9

Wireless Identification & Sensing Platform ................................................................. 9

2.1 Introduction ............................................................................................................ 10

2.2 Properties of WISP ............................................................................................... 11

2.3 Working Principle of the WISP: ........................................................................... 11

2.3.1 Downlink .......................................................................................................... 13

2.3.2 Uplink ................................................................................................................ 13

2.4 Hardware Details of WISP: .................................................................................. 13

2.5 Operating Voltage: ................................................................................................. 14

2.6 WISP Firmware: ..................................................................................................... 14

2.6.1 Firmware status: ............................................................................................... 15

2.7 Programming the WISP: ....................................................................................... 16

2.8 Power Budget of the WISP: .................................................................................. 16

CHAPTER 3 ....................................................................................................................... 21

RFID COMMUNICATION PROTOCOL ......................................................................... 21

3.1 Introduction: .......................................................................................................... 22

3.2 History of Communication Protocol: .................................................................. 22

3.3 EPCglobal Generation 1: .................................................................................... 23

3.4 EPC global class 0: ............................................................................................. 23

3.4.1 Benefit & Drawbacks: ....................................................................................... 26
LIST OF FIGURES

FIGURE 2.1 THE PICTORIAL VIEW OF WISP ......................................................... 10
FIGURE 2.3: THE ANALOG FRONT END (AFE) SCHEMATIC OF WISP .......................... 12
FIGURE 2.2: THE BLOCK DIAGRAM OF WISP ..................................................... 12
TABLE 2.1: WISP FIRMWARE SPECIFICATIONS .................................................. 16
FIGURE 2.4: THE EXPERIMENTAL SETUP OF POWER BUDGET ............................ 17
FIGURE 2.5: THE INPUT POWER VS OUTPUT POWER CURVE .................................. 18
FIGURE 2.6: THEORETICAL POWER VS THEORETICAL DISTANCE CURVE. .................. 19
FIGURE 2.7: PRACTICAL DISTANCE VS VOLTAGE CURVE .................................... 20
FIGURE 2.8: PRACTICAL POWER VS DISTANCE ............................................. 20

Table: 3.1 SOME RFID AIR INTERFACE PROTOCOL [4] ....................................... 23
FIGURE 3.1: CLASS 0 READER SYMBOL, DEPICTED AS BASEBAND LEVEL [4] ............. 24
FIGURE 3.2: CLASS 0 TAG SYMBOL [4] .............................................................. 25
FIGURE 3.3: SIMPLIFIED CLASS 0 MAC SCHEME [4] ......................................... 25
FIGURE 3.4: SIMPLIFIED CLASS 0 STATE DIAGRAM [4] ....................................... 27
FIGURE 3.5: CLASS 1 READER SYMBOL, DEPICTED AS BASEBAND LEVEL [4] ............. 28
FIGURE 3.6: CLASS 1 READER TRANSMISSION SEQUENCE [4] ............................ 28
FIGURE 3.7: CLASS 1 TAG SYMBOL [4] .............................................................. 29
FIGURE 3.8: THE PING ID COMMAND IS FOLLOWED BY BIN MARKERS DEFINING THREE SLOTS [4] .......................................................... 30
FIGURE 3.9: SIMPLIFIED BINARY TREE SEARCH BY PingID COMMAND [4] .......... 31
FIGURE 3.11: CLASS 1 READER COMMAND STRUCTURE [4] .............................. 33
FIGURE 3.13: READER TO TAG RF ENVELOPE [6] ............................................. 35
FIGURE 3.14: INTERROGATOR POWER UP AND POWER DOWN RF ENVELOPE [6] ... 36
FIGURE 3.15: R=>T PREAMBLE AND FRAME SYNC [6] ....................................... 37
FIGURE 3.16: FM0 BASIS FUNCTION AND GENERATOR STATE DIAGRAM [6] ............. 38
FIGURE 3.17: FM0 SYMBOL AND SEQUENCE3 [6] ............................................. 39
FIGURE 3.18: TERMINATING FM0 TRANSMISSION [5] ......................................... 39
FIGURE 3.20: MILLER BASIS FUNCTION AND GENERATOR STATE DIAGRAM [6] .... 40
FIGURE 3.25: THE HANDSHAKING SIGNAL OF INVENTORY AND ACCESS COMMAND .......................................................... 45
FIGURE 4.1: THE INVENTORY ROUND PACKET FROM THE READER ........................... 48
FIGURE 4.2: TRANSMITTING SIGNAL IN MILLER-2 CODING ................................ 52
FIGURE 4.3: THE ACKNOWLEDGEMENT SIGNAL .............................................. 53
FIGURE 4.4: EXPERIMENTAL RESULT OF FIGURE 4.4 ........................................ 54
FIGURE 4.5: EXPERIMENTAL RESULT OF FIGURE 4.5 ........................................ 55
FIGURE 5.1: THE CONCEPTUAL PICTURE OF THE WISP APPLICATION .................... 58
CHAPTER 1

INTRODUCTION

BACKGROUND

MOTIVATION

GOALS
1.1 BACKGROUND:

Passive Radio Frequency Identification is nascent technology that explores the possibility for designing very small, cheap Radio Frequency Identification (RFID) tag to be remotely powered and detected by inventory device such as RFID reader. Recently the Ultra-wide band High Frequency (UHF) tag becomes more popular over the High frequency (HF) tag. Unlike HF tag, UHF tag can be identified in far field region. The earlier HF tag is designed based on inductive coupling and capable of giving a distance of a few centimeters but the UHF tag can combined with other tag capabilities [8] and capable of giving the operating distance of few meters. Even the UHF RFID tag contains information and this information can be explored by UHF reader. Several applications depending on this technology are already developed such as SL travel card in Sweden.

When two device communicating (RFID tag and the reader) with each other, the significant issues will come out with the way of communication i.e. the protocol. There are so many protocol developed for the RFID communication. This protocol allows the researcher to follow the standard for the designing of tag and reader. However different passive UHF tag was tested with several protocols, EPC Electronics Product Code (EPC) standard class 1 Generation 2 protocol is more secure and reliable comparing to the other protocol. Normally the tag can understand only one protocol from the several protocols if there is no need of interoperability or if there is no need of cost minimization.

WISP (Wireless Identification and Sensing Platform) is a first programmable platform that explores the idea of combining several sensors with a RFID tag. This platform is narrowly tested with the EPC Class 1 Gen 2 protocol and because of the storage and computational capability of the WISP it allows us to test the several possibility of different RFID communication protocol in an inexpensive way.

1.2 MOTIVATION

WISP is an open source architecture that allows the RFID designer to build RFID software and hardware application in a very convenient way. But the WISP is not fully tested with different standard and it is only tested with Impinj US reader but it is not tested with the EU reader. The flexibility of communication of WISP with different RFID reader is still under question because of the communication failure of WISP with the other US reader based on EPC Class 1 Gen 2 protocol except Impinj reader. WISP firmware is based on Miller-4 encoding (one coding style based on EPC Class 1 Gen2) for transmitting the data. Then the issue comes in the front that is it really possible to connect the wisp with an EU reader which following different Encoding style such as Miller-2. The important of Miller-2 coding is for faster reading speed than Miller-4 encoding.
1.3 GOALS

The basic goal of the project is to develop the platform for the WISP to communicate with an EU standard reader which is based on Miller -2 encoding style for receiving the data from the tag. The partial goal is also to adapt the WISP for better compatibility with the EPC Class 1 Gen 2 commands according to the standard.
CHAPTER 2

WIRELESS IDENTIFICATION & SENSING PLATFORM

INTRODUCTION

PROPERTIES OF WISP

WORKING PRINCIPLE OF THE WISP

HARDWARE DETAILS OF WISP

OPERATING VOLTAGE

WISP Firmware

PROGRAMMING THE WISP

POWER BUDGET OF THE WISP
2.1 INTRODUCTION:

WISP (Wireless Identification and Sensing Platform) is a project of Intel Research Seattle with significant input from students and faculty of the University of Washington. The lead investigator of the WISP project is Joshua R. Smith, Principal Engineer at Intel Research Seattle. The key WISP design and development personnel are Alanson Sample (Intel Research Seattle & UW EE Grad student), Dan Yeager (UW EE Grad student), and Polly Powledge (Intel Research Seattle engineer) [1].

WISP stands for Wireless Identification and Sensing Platform. WISP is a hardware device which comes from Intel Corporation Ltd. It has the all capabilities of RFID tag and more than that. Here the term identification comes from the “radio frequency identification (RFID)”’. And the word sensing resembles the capability of wisp to sense the physical quantity like temperature, acceleration, light, liquid level, strain etc.

The WISP works as a passive RFID tag and it is powered by radio signal emitted from the RFID reader and this radio signal is used to harvest power for the WISP. The research related to WISP is still limited in single wisp. But in future the work on WISP will involve the interaction between multiple wisps and explore the new era of battery free wireless sensing network. The pictorial view of wisp is given in fig 2.1.

![FIGURE 2.1 THE PICTORIAL VIEW OF WISP](image-url)
2.2 PROPERTIES OF WISP

Wisp has the following features [1]

i. Up to 10ft range with harvested RF power
ii. Ultra-low power MSP430 microcontroller
iii. 32K of program space, 8K of storage
iv. Light, temperature and 3D-accelerometers
v. Backscatter communication to reader
vi. Reader to WISP communication (ASK)
vii. Real-time clock
viii. Storage capacitor (to sense without reader)
ix. Voltage sensor (measures stored charge)
x. Extensible hardware (to add new sensors)
xii. HW UART & GPIO for external connections
xiii. Works with select EPC Class 1 Gen 2 readers
xiv. WISP software to sense and upload data
xv. Reader application to drive WISP
xvi. Industry standard development tools
xvii. Access to hardware design and source code

2.3 WORKING PRINCIPLE OF THE WISP:

The block diagram of WISP is given in figure 2.2. The MCU (Microcontroller unit) in the block diagram is used for computation and programming the WISP to work on different protocol and because of this microcontroller wisp is mentioned as programmable passive RFID tag. As a passive tag it has a power harvester to generate the proper power for all the module of the WISP from the remote power source. The MCU is powered by the power harvester. There is some inbuilt sensor such as accelerometer with the microcontroller. A simple dipole antenna does the job of air interface with analog part of the circuit.

As mentioned earlier, Wisp is passive RFID tag. So it has a power harvester. One part of the power harvester is an RF rectifier. It rectifies the radio signal which generates from the RFID reader and use this rectified dc voltage to power up the whole system of WISP. The power harvester also has a multistage voltage multiplier and according to the figure 2.3 the rectifier works as an half wave rectifier i.e. the current will only pass to the circuit for the positive phase of the RF signal from reader.
There is a transistor between the antenna ports in figure 2.3. This transistor works as a modulator. Figure 2.3 shows the schematic of WISP analog front end (AFE). The WISP AFE has connection for the two branches of dipole antenna, an output for the dc power, an output for reader to wisp communication (downlink) and an input for the wisp to reader communication (uplink) [2].
2.3.1 DOWNLINK

The reader amplitude modulates the RF carrier of 915MHz what it emits. The amplitude of the carrier wave drops by 10% and usually the form of carrier wave remain same. The duration of the low indicates one or zero but the short break means ‘zero’ and long break means ‘one’. The decoder produces a signal that matches the envelope of the carrier which is emitted by the reader and then the microcontroller decodes the duration of the lows to detect reader to tag bits (1 or 0) by software written as firmware inside of microcontroller.

2.3.2 UPLINK

Backscatter radiation, the way that is used by RFID tag is also used by WISP. It means change in the amount of energy reflected back to the reader by modulating the impedance between antennas. As mentioned earlier the transistor between the two branches of dipole antenna is used as a modulator. When the transistor is on i.e. the two branches of the antenna is short circuited and when the transistor is off, it has no effect on transistor. The downlink and power harvesting occurs when the transistor is in off state. A bipolar transistor is used in WISP 4.1 version.

2.4 HARDWARE DETAILS OF WISP:

LED: The led is controlled by pin 1.6 (P1.6) of the microcontroller.

INTERNAL TEMPERATURE SENSOR: This is built from TI (Texas Instrument).

EXTERNAL TEMPERATURE SENSOR: This sensor enables or disabled by pin 1.0 (P1.0) or TEMP_POWER port 1.

ACCELEROMETER: The accelerometer sensor power is enabled / disabled by pin 1.5 (P1.59, or ACCEL_POWER on Port 1. The accelerometer is waited to reach x% of final value. Sample INCH_ACCEL_X, INCH_ACCEL_Y, INCH_ACCEL_Z pin 2.0, 2.1, 2.2 (P2.0, 2.1, 2.2).

RECTIFIED VOLTAGE SENSOR: This one is controlled by pin 3.3 (P3.3).

CAPACITANCE: For driving the cap the CAP_SENSE should be high in port 1 (P1.7) and the output logic will be 1 and the input pin must be changed to CAP_SENSE. The length of time required for CAP_SENSE to become in input 0 logic level is measure with DCO in the MHz range.

EPROM: 1.8V non-volatile external EPROM is included in WISP.

COMMUNICATION IN/OUT: To transmit and receive the WISP need to set the TX_PIN on port 1 and sample RX_PIN on port 1. But this is normally unnecessary because the optimized assembly code do the both function
CONNECTING EXTERNAL SENSOR: P2.3 is ready to receive the analog output of external sensor and pin 1.4 (P1.4) is used to supply the power.

VOLTAGE SUPERVISOR: this is connected to P2.4 or VOLTAGE_SV_PIN on port 2 and this one is used to make wisp to wait in LM4 i.e. low power mode 4 to get enough power to do some task.

P1.0, P1.1, P1.2, P1.3, P1.6, P2.0, P2.1 and P2.2 are all multiplexed with general purpose digital I/O pin and so that it is needed to configure the pin to proper port before using.

The schematic and the PCB layout is shown in Appendix III & Appendix II respectively.

2.5 OPERATING VOLTAGE:

The microcontroller inside the WISP draws 1mA current at full speed (200uA per MHz) which limits the wireless range when it runs continuously. To overcome the limitation, the MCU is in sleep mode when it is not necessary and in the meantime a capacitor is used to store sufficient energy for MCU and when the capacitor get proper energy, the MCU wakes up to do some computation or sensing depending on the specific application.

The important issue is how long the capacitor will take to discharge the stored voltage. Because the instantaneous consumption of power is higher than the average power that is available from RF. So the needed time for the capacitor discharge is the time limitation for the MCU to do its computation.

Mathematically the energy stored in the capacitor,

\[ E = \frac{1}{2} CV^2 \]  \hspace{1cm} (2.1)

Where, \( C = 10\mu F \), \( V \) = volts on the capacitor.

The rate of capacitor discharge or current consumption,

\[ I = C \frac{dV}{dT} \]  \hspace{1cm} (2.2)

Where, \( C = 10\mu F \) and \( dV = \) change in voltage & \( dT = \) time duration.

As per design the MCU needs 1.8V if the charge of the capacitor is 2V and MCU draws 1mA and we will get 2ms for MCU operation. The MCU consumes constant power according to per instruction regardless of clock cycle. So it will be a nice trick to reduce clock cycle to reduce power consumption. The exceptional case in this matter that when MCU is consuming clock cycle without doing computation. Suck kind of operation happens when MCU involve to create some loop for delay.

2.6 WISP FIRMWARE:
The firmware of wisp is still under development. It has some major feature like read, query etc. but still it has some major drawback such as WRITE, KILL, BLOCKWRITE etc. functions are not developed in the firmware. Also the communication code for the firmware is developed for specific reader. The firmware code is written in C and assembly language and the microcontroller MSP430F2013 is used here. The supported four applications by the firmware are given below:

1. SIMPLE_QUERY_ACK which returns a hardcode value.
2. SENSOR_DATA_IN_ID returns a sensor data as tag value.
3. SIMPLE_READ_COMMAND give access of READ command and returns a word of counter data
4. SENSOR_DATA_IN_READ_COMMAND response to a READ command by returning sensor data.

Now the WISP supports only 3D accelerometer and temperature sensor.

There are three other feature options in the WISP firmware: ENABLE_SLOTS, ENABLE_SESSIONS, and ENABLE_HANDLE_CHECKING. Support for these options is both optional and highly experimental. Using these options has the following drawbacks:

1. It's brand new code, and not well tested;
2. It adds a lot of code, which means you'll likely hit the free IAR Kickstart edition compiler's 4K limit;
3. The WISP works well without these features;
4. Enabling these features means more code execution, which translates directly into greater power consumption and diminished range and performance.

2.6.1 FIRMWARE STATUS:

The following table gives the specification for the release firmware.

<table>
<thead>
<tr>
<th>Firmware source code</th>
<th>WISP hardware</th>
<th>RFID Reader</th>
<th>Functionality</th>
<th>RFID protocol</th>
<th>Modulation</th>
<th>Range (cold start)</th>
<th>Comments</th>
<th>Known Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>hw41_D4 1.c - r65</td>
<td>4.1DL Impinj3.0.2</td>
<td>query-ack/query-ack-sensor</td>
<td>EPC Class 1 Gen 2</td>
<td>Miller-4</td>
<td>10'3&quot; range with CMOS modulator</td>
<td>Implementation notes is given</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TABLE 2.1: WISP FIRMWARE SPECIFICATIONS

<table>
<thead>
<tr>
<th>re</th>
<th>read/</th>
<th>read-</th>
<th>sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>read/</td>
<td>read-</td>
<td>sensor</td>
<td></td>
</tr>
<tr>
<td>hw41_D53.c - r14</td>
<td>4.1DL Impinj 3.0.2</td>
<td>query-ack/query-ack-sensor/read-sensor</td>
<td>EPC Class 1 Gen 2</td>
</tr>
</tbody>
</table>

2.7 PROGRAMMING THE WISP:

The software is used for programming the WISP is well known IAE Embedded Workbench which is very popular to program MSP microcontroller. There are two types of debugger is provided connect the WISP with workstation. One is FET debugger and another one is USB debugger.

For our experiment we used USB debugger which is more convenient to use. The driver for the USB debugger is also provided with the device. The IAR workbench support many types of microcontroller. The model for the WISP 4.1 is MSP430F2132. The model must be selected in IAR workbench otherwise it will create software problem while downloading the code.

Sometimes if the workstation is connected to the internet, it will look for the debugger firmware update. But this update damages the debugger forever. It is a bug of the debugger. So it is better not to update the debugger firmware.

2.8 POWER BUDGET OF THE WISP:

To measure the power requirement of the WISP we used a Signal Generator at the dipole antenna port of the WISP to give the RF power. We set the Signal Generator to give the frequency of 915MHz i.e. the cut-off frequency of the WISP. By this experiment we want to calculate the how much power the WISP needs to turn on the micro controller and how much power it can harvest.
After this we replace the signal generator with a RF reader and measure the harvesting power according to the distance. Then we also calculate the theoretical receiving power of WISP and the transmitting power of WISP at different distance.

**Figure 2.4**: The Experimental Setup of Power Budget.

Figure 2.4 depicts the setup for measuring the harvesting power at the input point of Voltage regulator. In this setup the MCU is operating in low power mode 4 i.e. all the operating device inside the microcontroller is turned off so that the leakage current is zero in MCU. To measure the output power we need voltage and current. We can get the voltage from voltmeter at the point of harvester but getting current is challenging job because the wire carrying the current is the PCB track. So we put a dc power supply of turn on voltage of 1.8V at the point of Vout in WISP to measure the output current. The output power drops to zero when the harvester reaches the voltage of 1.8V. The fig 2.5 describes the relation between output power and output voltage with input power. The input power is calculated from the signal generator. From the curve we can find that the output power becomes zero at the output voltage level of 1.8V. So 1.8 V is the turn on voltage [3].
Then we measure the receiving power of the dipole antenna of WISP comparing with distance theoretically considering the reader which we used to power up the WISP. The reader has a circular antenna of 6dBi and the frequency of the reader is 867MHz. The standard polarization loss of circular antenna is $LP=3\,\text{dBi}$. The gain of the standard WISP dipole antenna is $GR = 2\,\text{dBi}$. Fig-2.6 depicts the receiving power of the dipole antenna comparing to the distance according to the following theoretical equation (2.3).

$$P_R = P_T - 20\log\left(\frac{4\pi d}{\lambda}\right) + G_T + G_R - L_P \quad \cdots (2.3)$$
Now we measure the Vout against the distance in practically. We took four data in four place and plot for four Vout in every 0.5m distance. Figure 2.7 depicts the Voltage output according to the distance in meters and in this curve we took only one data among the four which best fits with the trend of curve. By mapping between the curve 2.5 and 2.7 we got the figure 2.8 which depicts the situation of power according to the distance. The fig-2.7 and fig-2.8 has the same trend. In the fig-2.8 we see that there is two unusual points come at the distance 4.5 m and 5m. We did the experiment inside a room where lots of reflection comes from the wall and also there is a highest possibility to presence of another RF source. The unusual point in the curve may come from the external RF source at that distance or because of reflection of side wall or other substance.

From this analysis we found the minimum voltage the need to turn on the microcontroller in the WISP is 1.8V. According to voltage we can achieve a distance of around 7m but according to power we can achieve the distance less than 4m. If we want to drive any load which consumes more than 50uW at the distance more than 4m we need to use some driver circuit to provide the excess current. In this case we can use buffer as a driver circuit. Otherwise we can increase the input power to get the required distance and output power. So we have a tradeoff between input power, output power and distance.
FIGURE 2.7: PRACTICAL DISTANCE VS VOLTAGE CURVE

FIGURE 2.8: PRACTICAL POWER VS DISTANCE
CHAPTER 3

RFID COMMUNICATION PROTOCOL

INTRODUCTION:

HISTORY OF COMMUNICATION PROTOCOL

EPC GLOBAL GENERATION 1 CLASS 0

EPC GENERATION 1 CLASS 1

EPC CLASS 1 GENERATION 2
3.1 INTRODUCTION:

Passive Radio Frequency Identification (RFID) is a rising wireless technology that can give us small computer chip which can be remotely powered or information can be remotely taken from the chip by the interrogator (RFID reader). Many applications are developed based on this concept such as scanning product code from shop when it is ready to sell and also monitor patient in hospital. Like every communication, interrogator and RFID need some specification for their conversation of digital message. This specification technically known as protocol and especially for RFID communication it is known as RFID protocol. Every communication protocol has some important criteria like:

- **Medium**: It means the medium of communication like wire (if it is wire then is it twisted pair or optical fiber) or wireless (if it is wireless, is it vacuum or normal air media).

- **Message of Format**: This is called the construction of message. The vital portion of this criteria is that the construction should be in such a way that both device that are going to communicate can understand the message. Like every human language such as Swedish, English, Bengali etc. has different alphabet, grammar, syntax etc. digital message should have some criteria like how many bits are going to be sent or what encoding should followed by the transmitter and receiver.

- **Medium Access**: It means how the medium of communication can be access like simplex or duplex communication system. For a complex system, how many devices can communicate at a time or how many people can share the media at a time. If there is a queue for communications who will communicate first and who will come next.

- **Context and Interpretation**: The meaning of the message must be established by reference to a context in which the exchange takes place even though the communication executes without any error.

Every protocol has to follow these criteria. A medium range RFID protocol in USA operates within the range of 902-928MHz or 2.4 – 2.483GHz with a channel of 500 kHz and 1 MHz respectively. But in EU system operation is under 865MHz –868MHz with a channel of 200 kHz. Radio waves can get through the obstacles in three ways: Diffraction, Reflection & Direct Penetration.

3.2 HISTORY OF COMMUNICATION PROTOCOL:

A key aspect of communication protocol is being understandable in both ways i.e. the speaker and the listener should know the specific way of communication what they are using. Here we are talking about standard i.e. called communication standard. With a commercial prospect, the both ends of the link can be manufactured by the same vendor or manufactured product by vendor must be agreed on communication standard. The most famous standard are IEEE
(Institute of Electrical and Electronics Engineers), ANSI (American National Standards Institute), IETF (Internet Engineering Task Force) and specifically for RFID, EPC global Inc.

Most used RFID protocol is mentioned in the following table 3.1:

<table>
<thead>
<tr>
<th>Tag type</th>
<th>FREQUENCY</th>
<th>125-134kHz</th>
<th>5-7MHz</th>
<th>13.56MHz</th>
<th>303-433MHz</th>
<th>860-960MHz</th>
<th>2.45GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ISO 18000-2</td>
<td>iPico</td>
<td>ISO14443</td>
<td>EPC class 0</td>
<td>Ic code</td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
<tr>
<td>HiTag</td>
<td>DF/iPX</td>
<td>ISO 15693</td>
<td>TIRIS</td>
<td>EPC class 1</td>
<td></td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ISO18000-3</td>
<td></td>
<td>Intellitag</td>
<td></td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIRIS</td>
<td></td>
<td>Title 21</td>
<td></td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Icode</td>
<td></td>
<td>AAR S918</td>
<td></td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ISO 10356</td>
<td></td>
<td>Ucode</td>
<td></td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
<tr>
<td>Semi-passive</td>
<td></td>
<td></td>
<td></td>
<td>AAR S918</td>
<td></td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Title 21</td>
<td></td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EZPass</td>
<td></td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Intelleflex</td>
<td></td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Maxim</td>
<td></td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>ANSI 371.2</td>
<td>ISO 18000-7</td>
<td></td>
<td>RFCode</td>
<td></td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ISO 18000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>ISO 18000</td>
<td></td>
<td></td>
<td></td>
<td>ISO 18000-4 Alien BAP</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE: 3.1 SOME RFID AIR INTERFACE PROTOCOL [4]**

For example, the protocol named EPC class 0 developed for supply chain application and Title 21 was developed for automobile oil application and both are operating in the range of 902-928MHz within USA ISM band and use frequency modulation to send data from tag to reader but Title 21 uses 0.6 and 1.2MHz frequency offsets comparing to EPC class 0 which uses 2.2 and 3.3MHz. Most of the protocols mentioned in the table 3.1 are not compatible to each other. Because of the cost and complexity most of the tags understand only one of these protocols.

### 3.3 EPCGLOBAL GENERATION 1:

Two kind of protocol is developed under EPCglobal Generation 1 denoted as Class 0 and Class 1. This protocol can provide improved forward and reversed link coding, tag state management and medium access control. Sometimes class 0 and class 1 is separated from class 1 Generation 2 tag but all of them have some common theoretical and practical interest of solving the problems faced by UHF passive tag protocol.

### 3.4 EPC GLOBAL CLASS 0:

The class 0 tag can only be factory written and read only tag but some field re-writable tag are also available in commercial market. 64 bit and 96 bit Electronic Product Codes tags are built
with this protocol. The table 3.2 gives the details of EPC. From the description it is clear that the 96 bits and the larger EPC are not clearly defined.

<table>
<thead>
<tr>
<th>EPC Type</th>
<th>Header Size</th>
<th>First Bits</th>
<th>Domain Manager</th>
<th>Object Class</th>
<th>Serial Number</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 bit type I</td>
<td>2</td>
<td>01</td>
<td>21</td>
<td>17</td>
<td>24</td>
<td>64</td>
</tr>
<tr>
<td>64 bit type II</td>
<td>2</td>
<td>10</td>
<td>15</td>
<td>13</td>
<td>34</td>
<td>64</td>
</tr>
<tr>
<td>64 bit type III</td>
<td>2</td>
<td>11</td>
<td>26</td>
<td>13</td>
<td>23</td>
<td>64</td>
</tr>
<tr>
<td>96 bit and more</td>
<td>8</td>
<td>00</td>
<td>28</td>
<td>24</td>
<td>36</td>
<td>96</td>
</tr>
</tbody>
</table>

Table 3.2: Description of EPC code in Class 0 protocol [5]

Reader symbol of class 0 tag is shown in figure 3.1. The binary ‘0’ and binary ‘1’ is chosen in such a way that it will provide high average power to the tag, a random sequence of 0s and 1s will give the average power of about 65% of the peak CW power of the reader. The special symbol NULL is used to send free tree traversal and also induce certain stage changes in the tag. The average power during NULL transmission is about 25% of the CW power.

![Figure 3.1: Class 0 Reader Symbol, Depicted as Baseband Level](image-url)

FIGURE 3.1: CLASS 0 READER SYMBOL, DEPICTED AS BASEBAND LEVEL [4]
FIGURE 3.2: CLASS 0 TAG SYMBOL; [4]

Tag symbol is depicted in figure 3.2. This tag symbol are conceptually similar to the 20/40KHz frequency-shift-tag symbol of early UHF protocol but the actual frequency is higher and is not harmonically related. The tag protocol is designed for bit by bit transmission rather than packet transmission so that the tag IC can decode the reader bit and decide to response immediately after completion of the gap and the modulation occurs during the end of the reader symbol. During the CW portion of the reader symbol, the tag backscattering is performed and the reader symbol is performed during the low power gap as soon as power is stored.

In the figure 3.3 the medium access control of the reader is sketched. This is a kind of special binary tree traversal process by reader to access the medium. The reader starts the tree traversal path with a special null symbol. When the tag hears the null symbol, the tag replies the first bit EPC code (in the example we used 4 bit simplified EPC code). In a reply, if reader hear only 0 or only 1, it will echoes the same bit or if reader hear the 0 and 1 at a time, it will reply the either 0 or 1 and the selection is random. In a consequence, if the tag hears the same bit what it replies, it will send the next bit of the EPC code or ID and otherwise it will be muted. In this process the tag need only to detect one bit at a time and no need of memory and also the reader’s job is simple because it doesn’t need to record any data.

FIGURE 3.3: SIMPLIFIED CLASS 0 MAC SCHEME [4]
3.4.1 BENEFIT & DRAWBACKS:

- The reader and the tag can simultaneously transmit data i.e. it is full duplex protocol and the net rate of data is twice of reader data rate.
- To perform this system in large baseband transients of the reader receiver that results from modulation transmit leakage, it is necessary to operate the tag in relatively high frequency which makes the tag signal to shift 2-3MHz from the carrier that makes to filter out the slow transmit leakage and recover the tag signal.
- This high shifter signal of tag from the reader carrier gives the easiest way to make sensitive receiver for the class 0 tag and to get reasonable forward-link distance even in the old fashioned IC technology.
- Because of large spectral displacements, the tag could be expected to radiate unexpected power level outside of the allowed power level in RFID operation according to European Regulatory recommendation.
- The large spectral could push the tag signal in out of band where the operation is more interested in narrow band.
- The procedures of medium access control sometimes create the security thread. Because every time the reader echoes the bit and reader usually do it in high power, so it is really easy to incept the communication and get the information about the ID which may be more private information rather than public information.
- The receiver of reader should have to detect the tag signal in few microseconds immediate after reader symbol so that the frequency level of both symbol should have considerable amount of difference which can arise the out of band problem.
- When a bunch of tag present with a same starting bit set in a dense area, all the tag tried to backscattering the bit together. So the received power by the single tag can be influenced by the other tag and some tag will be shadowed and the depth of the shadow will vary with the tag modulation of antenna for the same state.
- The tag which are not programmed yet and containing the default ID can't be used in this procedure.
- To overcome the tag ID problem, the reader can chose to singulate depends on three kind of ID. Where ID0 is produced when the tree traversal begins; ID1 is the stored number in the Tag at the time of manufacturing and ID2 is generally an EPC code. If it is allowed to singulate between ID0 and ID1, most of the problem will be solved but though the new ID is random number, there is possibility to get a same ID for more than a single tag.

3.4.2 COMMANDS:

Some special commands are used to initiate some special command. SetNegotiationPage is used to select the singulation between three IDs; some commands used for reset the tag and forced the tag to be muted or dormant; Read command for read the tag or Kill command for kill the tag. The simplified flow diagram of the tag is shown in figure 3.4.
3.5 EPC GENERATION 1 CLASS 1:

In this protocol the tag is capable to send an ID to a reader and the tag id is rewriteable for one time. But in the case of commercial use the tag can be reprogrammable with ID. 64 and 96 bit tag are available in the market with this protocol. The tag symbol, the reader symbol and the procedure of medium access is totally different comparing to the class 0 protocol.

The reader symbol is shown in figure 3.5.
This symbol is also as pulse-duration-encoded amplitude shift keyed as class 0 symbols. But in class 1 symbol doesn’t contain the null symbol. The pulse width is shorter and this symbol provides slightly more power to the reader but the transmitted spectrum is slightly greater for same data rate comparing to the class 0 symbol. The raw reader data rate is about 70kbps. The transaction sequence of the reader transmission is shown in figure 3.6.

The transaction gap initiates the starting of the command and the gap is longer than any other RF unwanted interferer. After this gap CW will transmit from the reader for 64µs which is followed by the original reader command that will end with a binary ‘1’ named as EOF (end of frame). During the sync interval tag may be reflect the advisability of speaking up. After sync interval the binary ‘1’ initiates the time period for the tag to respond the command of the reader. The reader has the separate startup and synchronization bit sequence for every command which provide the tag to enter in inventory round at every transaction gap which is not available in class 0 protocol. Class 1 is a half-duplex protocol because the reader talks for some milliseconds and then it waits for tag responds. So the reader data rate is slower than the class 0 protocol.
The tag symbol is shown figure 3.7. The modulation technique used here is known as F2F (frequency double frequency) modulation which is different from class 0 in accords with the use of packetized approach. F2F is also known as Aiken Bi-phase. From the figure 2.7, it is clear that the state transition occurs at the edge of the every state. when binary ‘0’ is transmitted, a single state transition presents in the middle of the symbol but in the case of binary ‘1’, three additional transition occurs which means the double of the frequency of the original one. This is also a kind of frequency shift keying as like in all backscattering coding. Comparing to the class 0 protocol, the used frequency is much lower and the tag data rate is about 140kbps.

The way of accessing the medium is different from class 0 protocol. Any specific way is not available in class 1 protocol but it provides some special command to facilitate a binary tree approach to collision resolution. Depending on the distance of the tag in the read-zone, there is several ways to access the media.

The most simplified way to avoid the collision is Global Scroll mode in where ScrollAllID is a command that a reader can continuously transmits and when the tag hears this command, it will backscatter his ID and CRC. This process is very effective for one tag within the read-zone. If there is more than three tags within the read-zone, all tag will try to send the ID and CRC which creates the collision. The collision is higher when all the tag transmits in the same power level (in the case when all tags are in same distance). Because of this inefficiency of reading with ScrollAllID command, some modification has done with this command. The reader will transmits ScrollAllID command until it hears a successful reply (ID and valid CRC) from tag; it then sends back a Quiet command with that ID to make the tag quiet. The tag remains quiet with the proper ID until it loses the power. Each time the tag is read, it becomes quiet. So the tag population will be less and the collision becomes lower. This process is good for the population 6-8 tags. If all tags transmit their ID in different power level (all tags are not in the same distance from the reader), they can be read according to the higher to lower power level. In this case, the reader needs to send the whole EPC for the quiet command. So the payload is high and the
communication becomes slow. The speed is about three times slower than the normal Global Scroll mode.

The another way to access the medium is PingID command which has a different sequence than figure 3.6 and this sequence is shown in figure 3.8. This PingID provides a filter consisting of starting location and a bit pattern which is known as mask. The tag will reply the EPC according to the match of the mask to EPC in the specific position that is mentioned in the filter. And the tag whose EPC doesn’t match with the mask on specific bit will be quiet.

![Diagram](image)

**FIGURE 3.8: THE PING ID COMMAND IS FOLLOWED BY BIN MARKERS DEFINING THREE SLOTS [4]**

After this command the reader will send a lonely binary ‘1’ to define the eight reply slots known as bins. The tag will chose a bin to reply depending on the 3 bits after the mask bits of the ID. So the reader can reduce the next three bits of the ID after observing in which bin the tag replies.

So the tag reply contains the next 8 bits of the ID and pay load becomes low. However no error checking occurs, this is not entirely reliable. The PingID binary tree traverse is shown in a figure 3.9. The mixed of ScrollAllID and PingID can be used to get the tag id back in reader.
FIGURE 3.9: SIMPLIFIED BINARY TREE SEARCH BY PINGID COMMAND [4]

The state diagram of class 1 protocol is given in figure 3.10 which is simpler than class 0 protocol. The tag which becomes asleep (if it is read) can be again awake if it is needed to count all the tags in the read-zone by a command Talk issued by the reader. This protocol has some explicit programming command named as EraseID and ProgramID. The tag can be programmed directly from awake state. The VerifyID is an important command from the reader by which tag will reply its ID, CRC and password and this is very important in dealing with tag which is not programmed yet.
FIGURE 3.10: SIMPLIFIED STATE DIAGRAM OF CLASS 1 TAG [4]
The general arrangements of the reader command are shown in figure 3.11. All commands start with a preamble of 20 binary ‘0’ symbol. All commands are consists of 8 bits and protected by a single parity bit. Each block is protected by a parity bit. Most of the commands use mask and so that the length and starting location of the mask are also be in commands as well with data. The running speed of the whole process is of 70kbps for the reader and 140kbps for the tag. The important thing to notice for the implementation of this standard is:

- The reader will send the LSB first and the tag will reply MSB first.
- The 16 bit data ('0') are used to terminated the CRC calculation are actually sent over the air by the tag.
3.6 EPC CLASS 1 GENERATION 2 TAG

This protocol is also known as ISO 18000-6C. According to the ISO 18000-6B, class 0 and class 1 protocol, there is some unavoidable limitation which inspired to develop the new protocol like class 1 gen 2 protocol. The limitations are:

- Both EPCglobal protocol provides the binary tree based collision resolution which is unfriendly to the new tag.
- Lack of link level security during programming operation.
- All protocol has the difficulties to maintain the unique session with tags.
- The inflexibility between the reader and tag data rate.
- There is a lack of way to minimize the inference in tag and reader spectra.
- EPC global protocol is susceptible to phantom or ghost tag.

The new clues that make the gen 2 protocol are different from other is given below:

- Tag data rate is flexible
- Spectral control of minimizing interference in reader and tag spectra.
- Separate protocol control bit with explicit declaration of EPC length.
- Use of Aloha-based adaptive collision resolution.
- Variable length command for inventory speed improvement.
- Link cover coding for secure tag programming
- A compliance and interoperability testing procedure by EPC global.
- Explicit specification of memory maps & locks provision and programming procedures.

3.6.1 INTERROGATOR (READER) TO TAG COMMUNICATION:

By modulating RF carrier using DSB-ASK, SSB-ASK or PR-ASK with PIE encoding, the interrogator start communication with one or more tags. The interrogator uses fixed data and coding format to run the inventory round (The period initiated by Query command and terminated by a either a query command or select command). Tags should have demodulate the all three kind of modulation techniques which used by interrogator.

![Figure 3.12: PIE Symbol][6]
The R=>T (reader to tag) link use the PIE symbol for data encoding. The reference time interval between readers to tag signaling is known as Tari values which is also known as data-0. The construction of data 0 and data 1 is shown in figure 3.12 and the rise time, modulation depth, PW and fall time is shown in table 3.3. Tari values for the interrogator must be within the range of 6.25µs to 25µs. The R=>T RF envelope is shown in figure 3.13. In figure 3.13, A is the electric and magnetic strength which show the maximum amplitude of the RF envelope and the PW is measured at the 50% point of the pulse.

<table>
<thead>
<tr>
<th>Tari</th>
<th>Parameter</th>
<th>Symbol</th>
<th>Minimum</th>
<th>Nominal</th>
<th>Maximum</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25 µs to 25 µs</td>
<td>Modulation Depth</td>
<td>(A–B)/A</td>
<td>80</td>
<td>90</td>
<td>100</td>
<td>%</td>
</tr>
<tr>
<td></td>
<td>RF Envelope Ripple</td>
<td>M_h = M_t</td>
<td>0</td>
<td></td>
<td>0.05(A–B)</td>
<td>V/m or A/m</td>
</tr>
<tr>
<td></td>
<td>RF Envelope Rise Time</td>
<td>t_{r,10–90%}</td>
<td>0</td>
<td></td>
<td>0.33Tari</td>
<td>µs</td>
</tr>
<tr>
<td></td>
<td>RF Envelope Fall Time</td>
<td>t_{f,10–90%}</td>
<td>0</td>
<td></td>
<td>0.33Tari</td>
<td>µs</td>
</tr>
<tr>
<td></td>
<td>RF Pulsewidth</td>
<td>PW</td>
<td>MAX(0.265Tari, 2)</td>
<td>0.525Tari</td>
<td>µs</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 3.3: RF ENVELOPE PARAMETER [6]**

**ASK Modulation**

**PR-ASK Modulation**

The interrogator power up and power down waveform is shown in figure 3.14 where it is seen that when the power reach the 10% level, the power will rise monotonically up to M1 ripple limit and the power should not fall down below 90% during the time Ts where Ts means settling time. The interrogator shall not issues command before the end of the time Ts. When the interrogator power goes down below 90% then the power will fall down monotonically up to the power off limit Ms. Once the reader powered off it should wait at least 1ms before powering up. The parameter for power up and power down of the reader is shown in table 3.4 and table 3.5
3.6.2 R=>T PREAMBLE AND FRAME –SYNC:

In figure 3.15, the preamble and the frame sync for R=>T is shown. The first inventory round starts with preamble preceding by Query command and all other signaling will start with frame-
sync. A preamble is consist of a fixed length start delimiter, a data-0 symbol, an R=>T calibration (RTcal) symbol and a T=>R (tag to reader) calibration (TRcal) symbol.

- **RTcal**: The RTcal is equal to length of data-0 plus length of data-1. A tag shall measure the length of RTcal and calculate pivot= RTcal/2. The symbol which is shorter than pivot counts as data-0s and which are greater than pivot counts as data-1s by the tag. Tag shall county symbol invalid if it is longer than 4 RTcal. An interrogator shall transmit CW for a minimum 8 RTcal before changing the RTcal.
- **TRcal**: The backscattering link frequency is measured from TRcal and DR (divide ratio) in preamble and payload respectively of a Query command that initiates an inventory round.

\[
BLF = \frac{DR}{TRcal}
\]

(3.1)

The limit of the TRcal is \(1.1 \times RTcal \leq TRcal \leq 3 \times RTcal\).

The frame-sync is identical to a preamble minus the TRcal symbol.

---

**FIGURE 3.15**: R=>T PREAMBLE AND FRAME SYNC [6]
A tag communicates with interrogator using backscattering modulation in which the tag switches the reflection coefficient of the antenna between two states in accordance with the data being sent. For the backscattering the tag selects the modulation format and the reader selects the encoding & data rate by the query command in the inventory round. The backscattering uses the ASK or PSK modulation.

The data encoding for the tag is either FM0 format or Miller subcarrier.

- **FM0 baseband**: The transition has occurs at the end of the each bit period and for a zero bit it has an extra transition in the middle. The FM0 symbol and the sequence are shown in figure 3.16. The FM0 signaling always end with a dummy data-1 at the end of the transmission which is shown in figure 3.17. The basic function of FM0 baseband and the state diagram for FM0 is shown in figure 3.16. In the state diagram four possible states by the two phase of each of the FM0 state function labeled as S1-S4. The FM0 has two types of sequence in figure 2.20. It is because of the symbol of prior communication. FM0 preamble is shown in figure 2.24 where there is two kind of preamble depending on TRext bit mentioned in Query command that initiates the inventory round. In the case of writing to the tag memory from the reply of tag, the tag uses the extended preamble regardless of TRext bit. In figure 3.19, the “v” is showing the violation of FM0 encoding because of non-inversion.

![FM0 Basis Functions and Generator State Diagram](image-url)

**FIGURE 3.16: FM0 BASIS FUNCTION AND GENERATOR STATE DIAGRAM [6]**
FIGURE 3.17: FM0 SYMBOL AND SEQUENCE [6]

FIGURE 3.18: TERMINATING FM0 TRANSMISSION [5]

FIGURE 3.19: FM0 R=>T PREAMBLE [6]
**Miller subcarrier:** In figure 3.20 the basic function and the state diagram of Miller function is shown. In the miller encoding there will be no phase inversion between every symbol but there is phase inversion between two data-0 symbols and there is a phase inversion at the middle of the data-1 symbol as like as data-0 symbol of FM0 baseband. The transmitted waveform is the


![Miller Basis Functions](image)

**FIGURE 3.20: MILLER BASIS FUNCTION AND GENERATOR STATE DIAGRAM [6]**

base waveform multiplied by a square wave at M times the symbol rate. The miller subcarrier sequence for M=2, 4 & 8 is shown in figure 3.21. Miller sequence always end with a dummy data-1 at the end of the transmission which is shown in figure 3.22. The miller preamble is also dependent to TRext bit of Query command that initiates at the inventory round. If TRext =0, then the preamble is consist of $4M/BLF=4*M$ square cycles with Miller coded 010111 and if TRext =1, the preamble is consist of $16M/BLF=16*M$ square cycles with Miller coded 010111. So for example, if TRext =1 and the M=4, the number of square cycles is $16*4= 64$. The figure of the miler preamble is shown in figure 2.23.
Miller Subcarrier Sequences

\[ \begin{array}{cccc}
\text{M=2} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
\text{M=4} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
\text{M=8} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
\end{array} \]

FIGURE 3.21: MILLER SUBCARRIER SEQUENCE [6]

FIGURE 3.23: MILLER PREAMBLE [6]
3.6.4 SOME IMPORTANT ISSUES:

- Tag should support the R=>T Tari values in the range of 6.25µs to 25µs.
- Tag should support the T=>R link frequencies and the tolerance in table 3.6.
- Tag should not exceed the maximum settling time period for receiving the interrogator command shown in table 3.4.
- The tag vendor shall specify the free-space sensitivity and backscattering modulation power.
- The R=>T and the T=>R transmission starts with MSB first.
- The protocol uses two CRC types: CRC-16 and CRC-5.

<table>
<thead>
<tr>
<th>DR: Divide Ratio</th>
<th>TRcal(^1) (µs +/- 1%)</th>
<th>BLF: Link Frequency (kHz)</th>
<th>Frequency Tolerance FT (nominal temp)</th>
<th>Frequency Tolerance FT (extended temp)</th>
<th>Frequency variation during backscatter</th>
</tr>
</thead>
<tbody>
<tr>
<td>64/3</td>
<td>33.3</td>
<td>640</td>
<td>+ / - 15%</td>
<td>+ / - 15%</td>
<td>+ / - 2.5%</td>
</tr>
<tr>
<td></td>
<td>66.7</td>
<td>320 &lt; BLF &lt; 640</td>
<td>+ / - 22%</td>
<td>+ / - 22%</td>
<td>+ / - 2.5%</td>
</tr>
<tr>
<td></td>
<td>83.3</td>
<td>256</td>
<td>+ / - 10%</td>
<td>+ / - 15%</td>
<td>+ / - 2.5%</td>
</tr>
<tr>
<td></td>
<td>133.3 &lt; TRcal &lt; 200</td>
<td>150 &lt; BLF &lt; 265</td>
<td>+ / - 10%</td>
<td>+ / - 15%</td>
<td>+ / - 2.5%</td>
</tr>
<tr>
<td>8</td>
<td>200 &lt; TRcal &lt; 225</td>
<td>96 &lt; BLF &lt; 107</td>
<td>+ / - 5%</td>
<td>+ / - 5%</td>
<td>+ / - 2.5%</td>
</tr>
<tr>
<td></td>
<td>17.2 &lt; TRcal &lt; 25</td>
<td>320 &lt; BLF &lt; 465</td>
<td>+ / - 19%</td>
<td>+ / - 19%</td>
<td>+ / - 2.5%</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>320</td>
<td>+ / - 10%</td>
<td>+ / - 15%</td>
<td>+ / - 2.5%</td>
</tr>
<tr>
<td></td>
<td>25 &lt; TRcal &lt; 31.25</td>
<td>256 &lt; BLF &lt; 320</td>
<td>+ / - 12%</td>
<td>+ / - 15%</td>
<td>+ / - 2.5%</td>
</tr>
<tr>
<td></td>
<td>31.25</td>
<td>256</td>
<td>+ / - 10%</td>
<td>+ / - 10%</td>
<td>+ / - 2.5%</td>
</tr>
<tr>
<td></td>
<td>31.25 &lt; TRcal &lt; 50</td>
<td>100 &lt; BLF &lt; 256</td>
<td>+ / - 10%</td>
<td>+ / - 10%</td>
<td>+ / - 2.5%</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>100</td>
<td>+ / - 7%</td>
<td>+ / - 7%</td>
<td>+ / - 2.5%</td>
</tr>
<tr>
<td></td>
<td>50 &lt; TRcal &lt; 75</td>
<td>107 &lt; BLF &lt; 160</td>
<td>+ / - 7%</td>
<td>+ / - 7%</td>
<td>+ / - 2.5%</td>
</tr>
<tr>
<td></td>
<td>75 &lt; TRcal &lt; 200</td>
<td>40 &lt; BLF &lt; 107</td>
<td>+ / - 4%</td>
<td>+ / - 4%</td>
<td>+ / - 2.5%</td>
</tr>
</tbody>
</table>

Table 3.6: T=>R link frequencies [6]

3.6.5 TAG MEMORY:

The tag has four kind of memory:

**Reserved Memory:** The Kill and Access password is stored in this memory if it is implemented in the tag. The memory address of Kill and Access password is 00h to 1Fh and 20h to 3Fh respectively.

**EPC memory:** It contains the StoredCRC at 00h to 1Fh, a storedPC at 10h to 1Fh, a code (such as EPC) at 20h and if the tag implements XPC then either one or two words beginning at 210h.

**TID memory:** It contains the ISO/IEC 15963 allocation class at 00h to 07h which consists of 8 bits.
**User memory**: it is an optional memory.

The logical memory map is shown in figure 3.24. Command that access the memory has a MemBank parameter that selects the bank. MemBank is defined as follows:

- **00<sub>2</sub>** → Reserved
- **01<sub>2</sub>** → EPC
- **10<sub>2</sub>** → TID
- **11<sub>2</sub>** → Users

### 3.6.6 COMMANDS:

There are basically three kinds of commands which are given below:

- **Select Commands**: `Select` (mandatory)
- **Inventory Commands**: `Query`, `QueryAdjust`, `QueryRep`, `ACK` and `NAK`. Here all commands are mandatory.
- **Access Commands**: In this class there are some mandatory command and also some optional commands. `Req_RN`, `Read`, `Write`, `Kill` and `Lock` are mandatory commands. The optional commands are `ACCESS`, `BlockWrite`, `BlockErase` and `BlockPermalock`.  

---

Page 44 of 80
The handshaking signal of basic inventory round and access of a single tag is shown in the next page figure 3.25.

**FIGURE 3.25: THE HANDSHAKING SIGNAL OF INVENTORY AND ACCESS COMMAND**
CHAPTER 4

COMMUNICATION BETWEEN WISP AND UHF READER

INTRODUCTION:

LINK FREQUENCY

ENCODING MISMATCH

HOW THIS TRANSMITTING CODE WORKS
4.1 INTRODUCTION:

According to the Gen 2 EPC protocol the communication between the reader and the RFID starts with the inventory command from the reader side (because this protocol known as reader talks first protocol). The wisp firmware is designed to communicate with a special reader called speedway impinj reader and the reader what we are using for the communication is not able to communicate with the WISP. Now the challenge comes to make it possible to talk with this wisp. The possible problem that can disturb the inventory round is given below:

1. Link Frequency
2. Encoding mismatch

To find out the Link Frequency we need to find out the characteristics of the Reader.

4.2 CHARACTERISTICS OF THE READER:

The RFID reader used in this experiment is from the company named CREDIPASS and the model no. is CPR 303EU. The specification of the reader is given in Appendix VII. The inventory packet from the reader according to the protocol should have the following parameters shown in table 4.1.

<table>
<thead>
<tr>
<th>Command</th>
<th>DR</th>
<th>M</th>
<th>TRext</th>
<th>Sel</th>
<th>Session</th>
<th>TARGET</th>
<th>Q</th>
<th>CRC-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>#no of bits</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1000</td>
<td>0: DR = 8</td>
<td>00:M=1</td>
<td>00: No Pilot Tone</td>
<td>00: All</td>
<td>00:S0</td>
<td>0:A</td>
<td>1:B</td>
<td>0-15</td>
</tr>
<tr>
<td>1: DR = 64/3</td>
<td>01:M=2</td>
<td>01:S1</td>
<td>01:S2</td>
<td>10:S2</td>
<td>11:S3</td>
<td>10:SL</td>
<td>11:SL</td>
<td>128 µs</td>
</tr>
</tbody>
</table>

Table 4.1: the bit pattern of the inventory round

We connect the WISP with reader through air interface and then observe inventory packet on the oscilloscope connecting the receiving pin of wisp to the oscilloscope. The bit pattern is given in figure 4.1. After comparing the bit pattern of figure 4.1 and figure 2.20 we got the value of TARI, TRcal and RTcal which are given in table 4.2.

<table>
<thead>
<tr>
<th>TARI</th>
<th>TRcal</th>
<th>RTcal</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.8 µs</td>
<td>128 µs</td>
<td>66.8 µs</td>
</tr>
</tbody>
</table>

Table 4.2: the value of TARI, TRcal and RTcal

We found the following bit pattern
Comparing the bit pattern with the table 4.1 and table 4.2 we got the parameter of the reader which is given in table 4.3.

![Image of a graph showing two series, Series1 and Series2, with data points at various frequencies.](image)

**FIGURE 4.1: THE INVENTORY ROUND PACKET FROM THE READER.**

<table>
<thead>
<tr>
<th>#no of bits</th>
<th>Command</th>
<th>DR</th>
<th>M</th>
<th>TReg</th>
<th>Sel</th>
<th>Session</th>
<th>TARGET</th>
<th>Q</th>
<th>CRC-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>0: DR = 8</td>
<td>01: M=2 Miller 2 coding</td>
<td>00: All</td>
<td>00:S0</td>
<td>0:A</td>
<td>0000</td>
<td>00001</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3: the parameters of the RFID reader

4.3 LINK FREQUENCY:

Link Frequency means the data rate between the RFID reader and the WISP. From the oscilloscope it is found that the link frequency of the reader (Uplink) i.e. the data rate from the reader to the wisp is about 40 KHz. According to the equation 3.1 and using the data from the table 4.2 and 4.3, the data rate from the reader to the WISP (Downlink) become 63 KHz. The downlink is very low comparing to the Impinj reader (Downlink is 256 KHz) and the uplink is the same as the Impinj reader.
The clock for the uplink data communication is same as the default firmware. But the clock for the downlink needs to modify for setting up the downlink communication. The clock rate must be higher than the receiving and transmitting data rate so that the microcontroller can do the calculation before sending and receiving data bits. The wisp firmware has two functions which are known as SEND_CLOCK and RECEIVE_CLOCK. With this clock we can fixed the clock rate of receiving and transmitting data of WISP. The MSP430 microcontroller has DCO (Digital Controlled Oscillator) which is used to set the clock frequency for the receiving and transmitting data. MCLK (master clock) and SMCLK (subsystem master clock) is supplied from the DCO.

Basic Clock System Control Register1 (BCSCTL1) is used to control the clock. The pattern of the BCSCTL1 is given in table 4.4. [7]

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>XT2OFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XT2 Oscillator is on if XT2OFF=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XT2 Oscillator is off if XT2OFF = 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XTS = 0; low frequency mode of LFXT1 oscillator.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XTS = 1; high frequency mode of LFXT1 oscillator.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIVA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIVA = 00; ACLK is divided by 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIVA = 01; ACLK is divided by 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIVA = 10; ACLK is divided by 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIVA = 11; ACLK is divided by 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSELx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>With this four bits sixteen different frequency range of DCO.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There is another control register named as DCOCTL (DCO control register). The bit format of this control register is given in table 4.5. [7]:

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCOx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>These 3 bits will divide the nominal frequency which is selected by RSELx by 8 steps and separated by approximately 10%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>These 5 bits selects the switching between the frequency of DCOx and DCOx +1. And if the DCOx =1112, there is no effect of MODx bit because it is already selected higher frequency level.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The clock rate is about 10 times higher than the data rate. The code is given in Appendix IV.

4.4 ENCODING MISMATCH

The firmware already designed for the WISP is following the Miller 4 coding. The reader which used in this experiment is designed to receive the miller 2 encoded data. This is the most challenging part to convert the miller 4 coding to miller 2 coding.

There is a several control register from the MSP430 micro-controller used to design the miller-2 coding. The detail of the control register is given below.

*TACTL (Timer A control register):* It is a 16 bits control register for timer A register (TAR). The control bits are shown in below [7].

<table>
<thead>
<tr>
<th>Bits</th>
<th>15-10</th>
<th>9-8</th>
<th>7-6</th>
<th>5-4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused</td>
<td>TASSELx</td>
<td>IDx</td>
<td>MCx</td>
<td>Unused</td>
<td>TACLR</td>
<td>TAIE</td>
<td>TAIFG</td>
<td></td>
</tr>
<tr>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td>w-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.6: TACTL register.

<table>
<thead>
<tr>
<th>Bits</th>
<th>7-6</th>
<th>5-4</th>
<th>3</th>
<th>2-1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELMx</td>
<td>DIVMx</td>
<td>SELx</td>
<td>DIVSx</td>
<td>DCOR</td>
<td></td>
</tr>
<tr>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.7: BCSCTL2 register.
SELMx: used for selecting MCLK source.
00: DCOCLK
01: DCOCLK
10: XT2CLK if XT2 oscillator is available or LFXT1CLK or VLOCLK.
11: VLOCLK or LFXT1CLK

DIVMx: 00: MCLK is divided by 1
01: MCLK is divided by 2
10: MCLK is divided by 4
11: MCLK is divided by 8

SELx: used for selecting SMCLK source.
0: DCOCLK
1: XT2CLK if XT2 oscillator is available or LFXT1CLK or VLOCLK.

DIVSx: 00: SMCLK is divided by 1
01: SMCLK is divided by 2
10: SMCLK is divided by 4
11: SMCLK is divided by 8

DCOR: Selection of DCO resistor.
0: Internal resistors
1: External resistors

\( TACCR0 \) (Timer A Capture/Compare Register): The timer A will count up to the value stored in the TACCR0 if the timer A operate in up count mode. If TACCR0 value is N, timer A will count up to (N+1) and then immediately becomes zero.

\( TACCTL0 \) (Timer A Compare Control Register 0): There is different mode of output signal. In this experiment the output mode is selected as toggle mode. In the toggle mode, Output signal will be toggle when timer A values is equal to TACCR0 value.

4.5 HOW THIS TRANSMITTING CODE WORKS:

According to our analysis the link frequency is \(~63\) kHz and the time period is \(~16\) µs. According to Miller-2 coding waveform in figure 2.8, the short pulse length is \(8\) µs and the long pulse length is \(16\) µs. The SMCLK is serving the Timer A from the source of DCOCLK. According to the figure 2.8, we have to transmit \(16M/BLF\) where every \(M/BLF\) contains two cycles of short pulse.

The constant value in TACCR0 can be calculated in the following way which toggles the output of the time A. SEND CLOCK macro gives the clock of \(~750\) KHz. SMCLK time period is then \(1.3\) µs.

\[ 1.3 \, \mu s \times 6 = \sim 8 \, \mu s. \]

\[ 1.3 \, \mu s \times 12 = \sim 16 \, \mu s \]
So the value in TACCR0 is 5 for the short pulse and 11 for the long pulse. Then the output of the Timer A will toggle in every 6 (5+1) cycles of SMCLK for short pulse and 12 (11+1) for long pulse.

In the code we need to load the TACCR0 counter by the value 5 and continue the loop up to finishing the 16M/BLF sequence according to figure 2.8. After sending the 16 M/BLF, the sequence of preamble of 010111 will come to be sent. For transmitting 1 we need to load the counter by the value of 11. To continue the sequence of 0 and 1, we need to synchronous the clock so that the proper value for the TACCR0 will be loaded in proper time. To synchronous the cycle, we use lot of NOPs instruction in the middle of the code. The code is given in APPENDIX 3. The exact pulse shape for the code is given in the figure 4.2. In figure 4.3 gives the pulse shape of acknowledgement according to figure of 3.3. The code of the transmitting RF is given in Appendix V. The flow diagram of the transmitting code is given in Appendix I.

![FIGURE 4.2: TRANSMITTING SIGNAL IN MILLER-2 CODING.](image-url)
4.6 IMPLEMENTATION OF ACCESS COMMAND:

There is lot of commands such as READ, ACCESS, WRITE, BLOCKWRITE, according to Gen2 protocol. Except QUERY command, no command was implemented in WISP protocol. Access command is a pre-command for all kind of READ, KILL, WRITE command. The ACCESS command is implemented in this protocol. According to the Gen 2 protocols the ACCESS command format is look like the table 4.8 and figure 4.9 and the experimental result of the table 4.8 and table 4.9 is given in the figure 4.4 and 4.5. In the tag memory, the handle (16bits) is copies to an array and again sent it back to the reader with the proper CRC-16 calculation following the transmitting code format. The code is given in Appendix VI

<table>
<thead>
<tr>
<th>No. of bits</th>
<th>Command</th>
<th>Password</th>
<th>RN</th>
<th>CRC-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>11000110</td>
<td>(1/2 access password) * RN16</td>
<td>Handle</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 4.8: Access Command (Reader to tag)
<table>
<thead>
<tr>
<th>RN</th>
<th>CRC-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Bits</td>
<td>16</td>
</tr>
<tr>
<td>Description</td>
<td>handle</td>
</tr>
</tbody>
</table>

Table 4.9: Tag reply to Access command

FIGURE 4.4: EXPERIMENTAL RESULT OF FIGURE 4.4.
FIGURE 4.5: EXPERIMENTAL RESULT OF FIGURE 4.5
CHAPTER 5

CONCLUSION

SUMMARY

FUTURE PROSPECT
5.1 SUMMARY:

The communication link between the WISP and the reader for the inventory round is successfully implemented and tested in this thesis paper with Miller-2 encoding according to EPC Class1 Gen2 protocol. The link rate of the RFID tag is decreased by the factor of 2, 4 and 8 depending on the Miller factor [9]. While WISP platform operates in Miller-2 coding the link rate becomes faster than Miller-4 coded platform, the inventory round becomes fast. The fast inventory round is more important to design the application where it is needed to detect the tag in a very fast way.

The ACCESS command of the Gen2 protocols is successfully implemented in WISP platform. This Access command is needed to run the command like Read, Write and BlockWrite command. The packet formation of handshaking signal for the inventory round is also tested properly with the Miller-2 criteria.

With Miller-2 encoded becomes frailer to errors because of lack of number of pulses. When the application designed with WISP platform becomes more sophisticated, it is better to use miller-4 coding. So the trade-off between speed and error depends on the different application.

The reader used in this project is a commercial reader. Vendors for the commercial reader provide the performance information about the reader with some limitation. With this limitation it is really difficult to interpret the reader configuration. Most of the reader is actually black box while reader has a lot of tuning parameter that can impact the performance of the reader. It is really hard to know what is happening inside the MAC and Physical layer of the reader.

5.2 FUTURE PROSPECT

The Write, BlockWrite, Kill, Lock commands of the Gen2 protocol can be implemented with WISP platform. But if it is tried with existing reader the Write command can’t be implemented because the reader is used to send only BlockWrite command.

Also the comparative speed and error analysis can be done with different encoding (miller 2, miller 4, miller 8, FM0).

Some security application is possible to design with WISP platform. It is possible to secure the cardboard box using the conductive ink and WISP platform. The conceptual picture of this application is shown in figure 5.1. The wisp will be connected with conductive link inside of the box. The information stored in the WISP should have relation of the properties of conductive ink such as impedance. Before sealing the box the wisp will be loaded with specific information. After sealing the box the information can be read by the reader. If any unauthorized person opens the box, the conductive link will be cut down results change of impedance that will also change the information of the WISP platform when it will be read again by the reader. By seeing the read data user can identified that the box is in the initial condition after the sealing or not.
FIGURE 5.1: THE CONCEPTUAL PICTURE OF THE WISP APPLICATION
The flow diagram of the transmitting code is given in the next page.
PCB layout of WISP (Both Side)
Details Schematic of WISP (continue)
Details Schematic of WISP (continue)
Details Schematic of WISP (continue)
APPENDIX IV

Modified code of send clock and receive clock

```c
#define SEND_CLOCK \
  BCSCTL1 = XT2OFF + RSEL2 + RSEL1 ; \
  DCOCTL = 0; // 673 kHz (666 KHz wanted)
/*
  BCSCTL1 = XT2OFF + RSEL2 ; \
  DCOCTL = DCO2 + DCO1 + DCO0 ; // 600 KHz
*/

/*
  BCSCTL1 = XT2OFF + RSEL3 + RSEL0 ; \
  DCOCTL = DCO2 + DCO1 ; // 3.0 MHz
*/
//BCSCTL1 = XT2OFF + RSEL3 + RSEL1 ; \
//DCOCTL = 0;
#define RECEIVE_CLOCK \
  BCSCTL1 = XT2OFF + RSEL3 + RSEL1 + RSEL0; \
  DCOCTL = 0; \
  BCSCTL2 = 0; // Rext = ON
```
Modified code of transmitting code for miller-2

```c
void sendToReader(volatile unsigned char *data, unsigned char numOfBits)
{
    SEND_CLOCK;

    TACTL &= ~TAIE;
    TAR = 0;
    // assign data address to dest
    dest = data;
    // Setup timer
    P1SEL |= TX_PIN; // select TIMER_A0
    // P1DIR |= TX_PIN; // already set.
    TACTL |= TACLR; // reset timer A

    TACTL = TASSEL1 + MC0; // up mode

    // TACTL = TASSEL1 + MC0 + ID0; // up mode
    // We divide SMCLK in two
    TACCR0 = 5; // this is 1 us period( 3 is 430x12x1)
    TAR = 0;
    TACCTL0 = OUTMOD2; // RESET MODE

#if MILLER_4_ENCODING
    BCSCTL2 |= DIVM_0;
    // BCSCTL2 |= DIVM_1;
#endif

    /***************************************************************************/
    // 4bit data: 0101, 0000, 1111, 0000, 0101, 0000, 0101, 0000
    /***************************************************************************/
    //<------------------------- The below code will initiate some set up --------------->/
```
//asm("MOV #05h, R14");
//asm("MOV #02h, R15");
bits = TRext;       // 6 cycles
asm("CMP #0001h, R5");  // 1 cycles
asm("JEQ TRextIs_1");   // 2 cycles
asm("MOV #0004h, R9");   // 1 cycles
asm("JMP otherSetup");   // 2 cycles

// initialize loop for 16 M/LF
asm("TRextIs_1:");
asm("MOV #000fh, R9");    // 2 cycles    *** this will change to right value
asm("NOP");

//
asm("otherSetup:");
bits = numOfBits;                // (3 cycles). This value will be adjusted. if numOfBit is constant, it takes 2 cycles
asm("MOV #0bh, R14");     // (2 cycles) R14 is used as timer value 11, it will be 2 us in 6 MHz
asm("MOV #05h, R15");       // (2 cycles) R15 is used as timer value 5, it will be 1 us in 6 MHz
asm("MOV @R4+, R7");      // (2 cycles) Assign data to R7
asm("MOV @R4+, R7");      // (2 cycles) Assign decimal 16 to R13, so it will reduce the 1 cycle from below code
asm("MOV R13, R6");       // (1 cycle)
asm("SWPB R7");           // (1 cycle) Swap Hi-byte and Low byte
asm("NOP");
asm("NOP");

// new timing needs 11 cycles
asm("NOP");
//asm("NOP");     // up to here, it make 1 to 0 transition.
//<-----------------------1 us -----------------------------
//asm("NOP");     // 1
//asm("NOP");     // 2
//asm("NOP");     // 3
//asm("NOP");     // 4
//asm("NOP");     // 5
//asm("NOP");     // 6
//asm("NOP");     // 7
//asm("NOP");     // 8
//asm("NOP");     // 9
// <---------- End of 1 us --------------------------------

// The below code will create the number of M/LF. According to the spec,
// if the TRext is 0, there are 4 M/LF. If the TRext is 1, there are 16 M/LF
// The upper code executed 1 M/LF, so the count(R9) should be number of M/LF - 1
//asm("MOV #000fh, R9");   // 2 cycles    *** this will change to right value
asm("MOV #0001h, R10");;  // 1 cycles

// The below code will create the number base encoding waveform., so the number of count(R9) should be times of M
// For example, if M = 2 and TRext are 1(16, the number of count
should be 32.
asm("M_LF_Count:");
asm("NOP"); // 1
asm("NOP"); // 2
asm("NOP"); // 3
asm("NOP"); // 4
asm("NOP"); // 5
asm("NOP"); // 6
asm("NOP"); // 7
asm("NOP"); // 8
asm("NOP"); // 9
asm("NOP"); // 10
asm("NOP"); // 11
asm("NOP"); // 12
asm("NOP"); // 13
asm("NOP"); // 14
asm("NOP"); // 15
asm("NOP"); // 16
  // asm("NOP"); // 17
asm("CMP R10, R9"); // 1 cycle
asm("JEQ M_LF_Count_End"); // 2 cycles
asm("INC R10"); // 1 cycle
asm("NOP"); // 22
asm("JMP M_LF_Count"); // 2 cycles
asm("M_LF_Count_End:");
  // this code is preamble for 010111, but for the loop, it will only
send 01011
asm("MOV #5c00h, R9"); // 2 cycles
asm("MOV #0006h, R10"); // 2 cycles
  // this should be counted as 0. Therefore, Assembly DEC line should be
1 after executing
asm("Preamble_Loop:");
asm("DEC R10"); // 1 cycle
asm("JZ last_preamble_set"); // 2 cycles
asm("RLC R9"); // 1 cycle
asm("JNC preamble_Zero"); // 2 cycles ... up to 6
  // this is 1 case for preamble
asm("NOP");
#if USE_2132
asm("MOV R14, TA0CCR0"); // 4 cycle ... 10
#else
asm("MOV R14, TACCR0"); // 4 cycle ... 10
#endif
asm("NOP");
asm("NOP");
asm("NOP");
#if USE_2132
asm("MOV R15, TA0CCR0"); // 4 cycle ... 19
#else
asm("MOV R15, TACCR0"); // 4 cycle ... 19
#endif
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP"); // .. 22
asm("JMP Preamble_Loop"); // 2 cycles .. 24

// this is 0 case for preamble
asm("preamble_Zero:");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("JMP Preamble_Loop"); // 2 cycles .. 24
asm("last_preamble_set:");
asm("NOP"); // 4
asm("NOP");
asm("NOP"); // TURN ON
asm("NOP");
#if USE_2132
asm("MOV.B R14, TA0CCR0"); // 4 cycles
#else
asm("MOV.B R14, TACCR0"); // 4 cycles
#endif
asm("NOP");
asm("NOP");
asm("NOP");
#if USE_2132
asm("MOV.B R15, TA0CCR0");
#else
asm("MOV.B R15, TACCR0");
#endif
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
//asm("NOP");
//<--------------------- end of initial set up

Page 71 of 80
```c
#include <avr/io.h>

#define USE_2132

void transmit_data(unsigned char data) {
    asm volatile(
        "
        // this is starting of loop
        "
        "loop:
        "
        "asm("LOOPAGAIN:");
        "
        "asm("DEC R5");  // 1 cycle
        "
        "asm("JEQ Three_Cycle_Loop_End");  // 2 cycle
        "
        "//<------------------loop condition ------------------
        "
        "asm("NOP");  // 1 cycle
        "
        "asm("RLC R7");  // 1 cycle
        "
        "asm("JNC bit_is_zero");  // 2 cycles ..7
        "
        "// bit is 1
        "
        "asm("bit_is_one:");
        "
        "#if USE_2132
        "
        "asm("MOV R14, TA0CCR0");  // 4 cycles ..11
        "
        "#else
        "
        "asm("MOV R14, TACCR0");  // 4 cycles ..11
        "
        "#endif  // 4 cycles ..11
        "
        "asm("DEC R6");  // 1 cycle ..12
        "
        "asm("JNZ bit_Count_Is_Not_16");  // 2 cycle ..14
        "
        "// This code will assign new data from reply and then swap bytes.
        "
        "After that, update R6 with 16 bits
        "
        "asm("MOV @R4+, R7");
        "
        "#if USE_2132
        "
        "asm("MOV R15, TA0CCR0");  // 4 cycles .. 20
        "
        "#else
        "
        "asm("MOV R15, TACCR0");  // 4 cycles .. 20
        "
        "#endif  // 4 cycles .. 20
        "
        "asm("MOV R13, R6");  // 1 cycle .. 22
        "
        "asm("MOV R15, TACCR0");  // 4 cycles .. 20
        "
        "asm("MOV @R4+, R7");
        "
        "asm("SWPB R7");  // 1 cycle .. 21
        "
        "asm("MOV R13, R6");  // 1 cycle .. 22
        "
        "// End of assigning data byte
        "
        "asm("JMP LOOPAGAIN");  // 2 cycle .. 24
        "
        "seq_zero:");
        "
        "asm("NOP");  // 1 cycle .. 3
        "
        "#if USE_2132
        "
        "asm("MOV R15, TA0CCR0");  // 4 cycles .. .7
        "
        "#else
        "
        "asm("MOV R15, TACCR0");  // 4 cycles .. .7
        "
        "#endif
        "
        "// bit is 0, so it will check that next bit is 0 or not
        "
        "asm("bit_is_zero:");  // up to 7 cycles
        "
        "asm("DEC R6");  // 1 cycle .. 8
        "
        "asm("JNE bit_Count_Is_Not_16_From0");  // 2 cycles .. 10
    ");
}
```

// The main loop code for transmitting data in 6 MHz. This will transmit data in real time.
// R5(bits) and R6(word count) must be 1 bigger than desired value.
// Ex) if you want to send 16 bits, you have to store 17 to R5.
// bit count is 16
asm("DEC R5"); // 1 cycle .. 11
asm("JEQ Thirteen_Cycle_Loop_End"); // 2 cycle .. 13
// This code will assign new data from reply and then swap bytes.

After that, update R6 with 16 bits
asm("MOV @R4+,R7"); // 2 cycles 15
asm("SWPB R7"); // 1 cycle 16
asm("MOV R13, R6"); // 1 cycles 17
// End of assigning new data byte
asm("RLC R7"); // 1 cycles 18
asm("JC nextBitIs1"); // 2 cycles .. 20
// bit is 0
#if USE_2132
asm("MOV R14, TA0CCR0"); // 4 cycles .. 24
#else
asm("MOV R14, TACCR0"); // 4 cycles .. 24
#endif
// Next bit is 0 , it is 00 case
asm("JMP seq_zero");

// <<----------this code is 00 case with no 16 bits.
asm("bit_Count_Is_Not_16_From0:"); // up to 10 cycles
asm("DEC R5"); // 1 cycle 11
asm("JEQ Thirteen_Cycle_Loop_End"); // 2 cycle ..13
asm("NOP"); // 1 cycles ..14
asm("NOP"); // 1 cycles ..15
asm("NOP"); // 1 cycles ..16
asm("NOP"); // 1 cycles ..17
asm("RLC R7"); // 1 cycle .. 18
asm("JC nextBitIs1"); // 2 cycles .. 20
#if USE_2132
asm("MOV R14, TA0CCR0"); // 4 cycles .. 24
#else
asm("MOV R14, TACCR0"); // 4 cycles .. 24
#endif
asm("JMP seq_zero"); // 2 cycles .. 2

// whenever current bit is 0, then next bit is 1
asm("nextBitIs1:"); // 20
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP"); // 24
asm("NOP");
asm("NOP");
asm("NOP");
asm("NOP");
asm("JMP bit_is_one"); // end of bit 0 .. 7
asm("bit_Count_Is_Not_16:"); // up to here 14
asm("NOP");
#if USE_2132
asm("MOV R15, TA0CCR0");       // 4 cycles   .. 20
#else
asm("MOV R15, TACCR0");       // 4 cycles   .. 20
#endif
asm("NOP");
asm("NOP");
asm("NOP");
asm("JMP LOOPAGAIN");       // 2 cycle   .. 24

// below code is the end of loop code
asm("Three_Cycle_Loop_End:");
asm("JMP lastBit");      // 2 cycles   .. 5
asm("Thirteen_Cycle_Loop_End:");
asm("NOP");      // 1
asm("NOP");      // 2
asm("NOP");      // 3
asm("NOP");      // 4
asm("NOP");      // 5
asm("NOP");      // 6
asm("NOP");      // 7
asm("NOP");      // 8
asm("NOP");      // 9
asm("NOP");      // 10
asm("NOP");      // 11 ..24
asm("NOP");      // 12
asm("NOP");      // 13
asm("NOP");      // 14
asm("JMP lastBit");

/*******************************************************************************
*   End of main loop
*********************************************************************************/
// this is last data 1 bit which is dummy data
asm("lastBit:");
asm("NOP");
asm("NOP");
#if USE_2132
asm("MOV.B R14, TA0CCR0");   // 4 cycles
#else
asm("MOV.B R14, TACCR0");   // 4 cycles
#endif
asm("NOP");
asm("NOP");
#if USE_2132
asm("MOV.B R15, TA0CCR0");
#else
asm("MOV.B R15, TACCR0");
#endif
asm("NOP");
asm("NOP");
// experiment
asm("NOP");
//TACCR0 = 0;

TACCTL0 = 0;  // DON'T NEED THIS NOP
RECEIVE_CLOCK;

}
Implemented code for ACCESS COMMAND

/// this below handle_access code for the acess command

inline void handle_access(volatile short nextState)
{
    TACCTL1 &= ~CCIE;
    TAR = 0;

    while ( TAR < 300 );
    TAR = 0;
    sendToReader(&queryReply[0], 33);

    /*
    Behaviour_____________
    */
    state = nextState;
}
Reader specification:

1.1 Reader Define

The CPR303EU is high performance RFID reader that reliably operated in outdoor circumstances. The I/O structure of the CPR303EU enables inputting sensor signal and outputting controller. With internal antenna and simple construction, CPR303EU provides a seal against adverse environmental conditions. It is designed based on ISO18000-6 so that it supports major tag protocols. The CPR303EU can be operated not only in stand-alone mode but also in remote control mode by adjusting internal selection switch which enables wide application range. The most suitable application range of this reader is parking-lot application, vehicle and human access control.

1.2 Operation Specifications

- ISO18000 6A, 6B, 6C singular which is satisfied a standard or it will recognize multiple Tag and there is a possibility which it will write a data in Tag memories.
- External interface support : RS-232, TCP/IP
- To use LED, the operation state identification is possible from the external.
- User Mode Possibility.
- Firmware upgradable by Bootloader
1.3 Communication Specifications

<table>
<thead>
<tr>
<th><strong>External Interface</strong></th>
<th>Serial</th>
<th>9600bps / 38,400bps / 115,200bps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ethernet</td>
<td>TCP/IP</td>
</tr>
<tr>
<td><strong>Air Protocol</strong></td>
<td>Tag Air Protocol</td>
<td>ISO 18000 – 6A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ISO 18000 – 6B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ISO 18000 – 6C</td>
</tr>
</tbody>
</table>

1.4 Environment Specifications

<table>
<thead>
<tr>
<th><strong>Operating Temp.</strong></th>
<th>−30 °C ~ 40 °C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Storage Temp.</strong></td>
<td>−30 °C ~ 70 °C</td>
</tr>
<tr>
<td><strong>Humidity</strong></td>
<td>10% ~ 95%</td>
</tr>
</tbody>
</table>
REFERENCES


