AN EXPERIMENTAL STUDY OF EPCGLOBAL CLASS-1 GENERATION-2 ANTI-COLLISION PROTOCOL FOR RFID SYSTEMS

A Thesis by

M. H. Maheesha H. De Silva

Bachelor of Science, Wichita State University, 2006

Submitted to the Department of Electrical Engineering and Computer Science
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

August 2010

AN EXPERIMENTAL STUDY OF EPCGLOBAL CLASS-1 GENERATION-2 ANTI-COLLISION PROTOCOL FOR RFID SYSTEMS

The following faculty members have examined the final copy of this thesis for form and content, and recommend that it be accepted in partial fulfillment of the requirement for the degree of Master of Science with a major in Electrical Engineering.

_____

Vinod Namboodiri, Committee Chair

_____

Edwin Sawan, Committee Member

_____

Krishna Krishnan, Committee Member

DEDICATION

To my loving wife for her endless support and patience;
my ever loving parents, brother, and sister for the sacrifices they had to make in life for me;
and all my admirable teachers for their priceless knowledge and grateful support

ACKNOWLEDGEMENTS

I am extremely thankful to my admirable thesis advisor, Dr. Vinod Namboodiri, for his dedicated support, encouragement, and supervision. He always made time for me in spite of his busy schedule and offered me helpful guidance on my academics, career, and life. I also take this opportunity to thank members of my committee for their time and effort.

I would like to extend my gratitude to Dr. Ravi Pendse for his valuable advice and suggestions throughout my carrier at Wichita State University. It has been my privilege to work under him as a graduate research assistant.

I take enormous pleasure in recognizing, with endless thanks, all who provided support for MATLAB and all who helped me in numerous other ways. Without them, this thesis would not have been possible.

ABSTRACT

Radio frequency identification (RFID) is used to identify, track, and manage tagged animate or inanimate objects automatically using wireless communication technology. RFID is similar to existing barcode identification, but it has additional features. RFID has the capability of scanning multiple objects at the same time. This improves productivity by reducing the time taken to identify objects. RFID has the capability to read through opaque material without requiring line of sight, thus saving time in processing that would otherwise require upward-facing objects. RFID is extremely appropriate for applications that require tags to be read at large distances. RFID readers and tags come in various sizes and forms, thus permitting this type of technology to be used in a broad variety of situations. Some tags are blast-proof, some tags are the size of lunch boxes, and some are smaller than a grain of rice. Also, RFID tags can be reprogrammable, thus reducing cost.

As RFID technology continues to grow rapidly, different issues and challenges are presented. A serious concern faced by RFID technology is the collisions that occur during communication. This is considered one of the immense challenges in RFID development because collisions limit system performance significantly. Collisions bring extra delay, a waste of bandwidth, and extra energy consumption to the interrogation process of RFID. Delays that arise due to collisions in RFID systems create significant issues and challenges to applications that require high inventory speed. Therefore, RFID system designers and researchers need to simulate these different environments before deployment to correctly identify various factors, such as the number of RFID readers needed, where to place these readers, etc.

The simulator developed in this research is called the RFID Simulator. It was developed completely from scratch to evaluate the performance of Slotted Aloha and EPCglobal Class-1

Generation-2 protocols for RFID systems. The RFID Simulator was designed to replicate a real-life RFID environment. It can be used to imitate hardware and has the capability to calculate the delay to any number of RFID tags, which is not possible with real-life RFID systems. As a result, the performance of RFID systems can be improved significantly. The integrity of the simulator was verified by comparing its results with mathematical analysis and experimental results. The RFID Simulator is a complete, all-in-one package, designed with the ability to be extended to a commercial RFID simulator, which will help immensely in the future development of RFID.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

CHAPTER 1

INTRODUCTION

1.1 What is RFID?

Radio frequency identification (RFID) is used to identify, track, and manage the tagged animate or inanimate objects automatically using wireless communication technology. RFID is similar to existing barcode identification, but it has additional features, such as communication without line of sight, large storage capacity, greater read range, capability of being reprogrammable, etc.

At some point in our lives, most of us have made a payment for goods by swiping a credit card, opened a door using a badge, or used the non-stop fast lanes on a toll road. If so, then we have already used RFID technology without even noticing it. Other RFID applications include, but are not limited to, supply chain tracking, animal tracking devices, vehicle tracking, and wrist/ankle bands for infant security. In short, anything that can be tagged is capable of being identified using RFID.

1.2 Why RFID?

Numerous methods are used to recognize objects, animals, and people. So, why is RFID advantageous?

RFID has the capability of scanning multiple objects at the same time. This improves productivity by reducing the time taken to identify objects. Some applications are required to have high inventory speeds. Hundreds of objects moving on a conveyor belt or applications that read tags on cars on a freeway require a very high read speed. RFID has the capability to read through opaque material without requiring line of sight, thus saving time in processing that would otherwise require upward-facing objects. RFID is extremely appropriate for applications

that require tags to be read at large distances. RFID readers and tags come in various sizes and forms, thus permitting RFID technologies to be used in a broad variety of situations. Some tags are blast-proof, some tags are the size of lunch boxes, and some are smaller than a grain of rice. RFID tags can be reprogrammable, which reduces cost. In today's world, RFID components can be purchased at a lower price, which facilitates in generating more revenue [1].

1.3 Current RFID Issues

As RFID technology grows rapidly, it also produces issues and challenges. One of the concerns regarding RFID is the possible threat to privacy rights. If the RFID tag is not killed (disabled) at the point of sale, there is a possibility of secret tracking and secret inventorying, which could be a risk to consumer privacy. Since RFID does not require line of sight, the unique identification that is attached to the item can be read and tracked by any party, even by rogue agents without the consumer's knowledge. Contactless credit cards have data that are linked to the banking information of people carrying the cards, and data in a tag of medicine may be linked to a patient's medical details. Stealing such vital information is very dangerous for the owner. This personal information may even be sold to a third party.

If data in a tag is not password protected or encrypted during communication, there is greater potential for the stored information in the tag to be easily stolen by malicious readers. This creates a security issue. RFID tags have the capability to be written and rewritten. If these tags are not properly protected, they can be a security liability since malicious readers can change the information stored in these tags and might even kill RFID tags.

When implementing RFID in the supply chain, each item is tagged using RFID, and the details related to each tagged item are located in a remote data center, which belongs to the respective manufacture. Therefore, every item bought by the consumer has to query its

manufacturer's database via the Internet in order to receive the data of that particular item. As each and every item gets tagged, the amount of data produced increases. Since possibly everything in this world can be tagged, including humans and animals, the amount of data that could be generated is unimaginable. This would certainly overpower all existing infrastructure mechanisms, and the Internet and network infrastructure may collapse unless required modifications are made.

1.4 Problem Description

A serious setback faced in RFID technology is the number of collisions during communication. This is considered one of the immense challenges in RFID development because collisions limit a system's performance significantly. Tag collision occurs where the reader cannot identify the data of a tag when multiple tags respond simultaneously. Collisions can also occur when two or more readers are operating at close range. Tag collisions bring extra delay, waste of bandwidth, and extra energy consumption to the interrogation process of RFID. Delays that arise due to collisions in RFID systems create significant issues and challenges to applications that require a high inventory speed. Hundreds of objects moving by on a conveyor belt and applications that read tags on cars on a freeway require a very high read speed. For example, if the RFID system in a supermarket is delayed due to collisions, only a fraction of the products would be identified by the reader from a cart full of products. This would create massive chaos in the inventory, bring huge financial losses to the companies, and likely cause the companies to put an end to this type of technology. Therefore, RFID system designers and researchers need to simulate these different environments before deployment to correctly identify various factors, such as the number of RFID readers needed, where to place these readers, etc.

But it is very difficult to find a free RFID simulator that can be used to identify RFID tag read times.

1.5 Contribution

As explained in section 1.4, a serious issue in the RFID technology is the number of collisions during communication. A simulator, RFID Simulator, was developed in this research completely from scratch to evaluate the performance of the Slotted Aloha and EPCglobal Class-1 Generation-2 (also referred to as Class 1 Gen 2) protocols for RFID systems. The RFID Simulator was designed to replicate a real-life RFID environment, could be used to imitate hardware, and has the capability to calculate the delay to any number of RFID tags, which is not possible with real-life RFID systems. The delay time to identify RFID tags using Slotted Aloha and EPCglobal Class-1 Generation-2 protocols can be obtained from the RFID Simulator. A mathematical model was developed to predict the RFID tag read time using the Slotted Aloha protocol. The results of the RFID Simulator were compared with the mathematical analysis to confirm the integrity of the simulator. An experimental study was performed to calculate the RFID tag read times using the EPCglobal Class 1 Gen 2 protocol. The testbed was constructed to simulate a real-world RFID system. The read time for EPCglobal Class-1 Generation-2 RFID tags was obtained using the testbed and the results were compared with the RFID Simulator results.

1.6 Organization of Thesis

This thesis presents the development of an RFID simulator to calculate the tag identification time using Slotted Aloha and EPCglobal Class-1 Generation-2 protocols. It is structured as follows: Chapter 1 presents a foreword to RFID technology, why RFID is used, and current RFID issues. Chapter 2 provides a literature survey of RFID, including RFID history,

components, applications, standards, collision types, and anti-collision protocols. Chapter 3 presents a detailed description of the Slotted Aloha anti-collision protocol. It also provides a mathematical model to identify RFID tag read times using the Slotted Aloha protocol. Then it gives a detailed description of the RFID Simulator and Slotted Aloha performance tests and results. Chapter 4 provides a thorough explanation of the EPCglobal Class-1 Generation-2 protocol, as well as performance tests and results using the RFID Simulator and a comparison of simulation results between the Slotted Aloha and EPCglobal Class-1 Generation-2 anti-collision protocols. Chapter 5 presents the practical implementation and compares the RFID Simulator and RFID testbed results for the EPCglobal Class-1 Generation-2 protocol. Chapter 6 concludes the thesis and presents ideas on future research work in the RFID anti-collision area.

CHAPTER 2

LITERATURE SURVEY

2.1 ABCs of RFID

Radio frequency identification is used to identify, track, and manage the tagged animate or inanimate objects automatically using wireless communication technology. Possibly everything in this world can be tagged, and anything that can be tagged is capable of being identified using RFID.

RFID tags and readers exchange data using electromagnetic waves. Electromagnetic waves are formed in the air using electric and magnetic components. Electromagnetic waves are classified in the order of increasing frequency: radio waves, microwaves, infrared, visible light, ultraviolet, x-rays, and gamma rays. Radio waves have a better capability to penetrate other materials such as cloth, wood, plastic, cardboard, etc., and they vary more gradually with a longer wavelength than other electromagnetic waves. But radio waves have difficulty passing through opaque materials such as metal, graphite, sodium, and liquids. The frequency of the radio wave is the main factor that decides the level of limitation that radio waves experience when penetrating opaque materials. Even though low frequency (LF) and high frequency (HF) waves can pass through opaque materials more efficiently than ultra high frequency (UHF) waves, UHF RFID tags have turned out to be the standard for the RFID supply chain since the low frequency and high frequency RFID tags are very large and economically high priced compared to ultra high frequency RFID tags [2].

The major components of an RFID system are RFID readers, RFID tags, and middleware. Tags are very tiny radio transponders, which can store and transmit data. The data that is stored in the tag has an identification number to recognize the object. RFID tags broadcast their data

when they are in the surrounding area of a reader. The reader gathers the tag data details and sends it to a host computer, which stores the data and uses it in an application program. Such application programs include an inventory system, a warehouse management system, or a database. Analysts use the data to learn, assess, and perform improvements, which decrease the time involved and expose different hidden dependencies and correlations.

2.2 History of RFID

Research and progress in radio frequency electronics, information technology, and materials science areas have made RFID commercially feasible. Antenna systems and radio frequency electronics used by RFID readers and tags have become feasible as a result of radio frequency electronic research and growth. Improvements in the area of information technology have made possible the use of the interrogator and host computer, the networking of RFID readers, and networking of RFID systems. Cheap RFID tags, as low as $0.05 per tag, have been able to be produced due to developments in materials science technology [3].

The primary technologies used in RFID are radio broadcasting and radar. The first known use of radio frequency identification is thought to have been the "identification, friend, or foe" system, also known as IFF, a system that allowed the British military to identify whether the arriving aircrafts were the enemy or not during World War II. The IFF system utilized coded recognition signals sent by friendly aircraft to identify them from hostile aircraft since the majority of the bombing missions were done at night. Even today, many armed forces in the world use an improved IFF system [4].

The paper entitled "Communications by Means of Reflected Power," published in the *Proceedings of the IRE* by Stockman in October 1948, is the closest article to the birth of RFID technology [3]. Stockman revealed that a remote transmitter can be powered using the

electromagnetic energy of radio waves, and that the remote transmitter can power itself and produce a radio transmission utilizing the electromagnetic energy produced from the received radio signals. This finding is the groundwork for present-day passive RFID tag technology. The growth of small and flexible RFID tags has been made possible because of developments in circuit etching [4].

In the past, RFID applications were primarily used for security and surveillance. These applications increased in the late 1960s and early 1970s. One of the most fundamental applications of RFID is electronic article surveillance (EAS). EAS tags are used in compact disc cases, books, and clothing [4].

RFID technology turned out to be more functional when the Intel Corporation pioneered the 4004 microprocessor in 1971, which was the world's initial single-chip microprocessor. Implementation of the microprocessor produced a technological revolution and thus resulted in smaller and faster integrated circuits. The rise of this novel technology allowed engineers to build complicated RFID products that could not have been considered a few years prior. RFID tags and readers became more sophisticated as microprocessors became smaller [4].

Lezin and Wilson were able to successfully insert RFID tags beneath the skin of dairy cows in 1978. This allowed the implanted cows to be monitored for potential health issues, ovulation cycles, and feeding patterns. Since the time when Lezin and Wilson tagged the first cow, many animals have been tagged, including fish, monkeys, cats, and dogs. Today, these tags can store not only a pet's identification number but also its unique identifier, name, address, and vaccination record [4].

The first major organization to employ RFID was the United States Department of Defense (DoD) because of possible logistics rewards. Initial DoD field tests tagged goods at the

shipping-container level. Massive offerings by companies like Gillette to the Massachusetts Institute of Technology Auto-ID Center assisted in ground-breaking RFID in the retail market. The retail world rapidly became aware of RFID after Wal-Mart required its primary 100 suppliers to supply RFID-tagged containers by the end of year 2006. Numerous other companies, like Best Buy, pursued the same, shortly after Wal-Mart's mandate [4]. The main motivation for these companies was the minimized costs of labor and storage, and decrease of product loss. Inventory became a much easier job with the implementation of RFID technology.

Sterzer of RCA filed a U.S. patent (number 4,001,822) called "Electronic license plate for monitor vehicles" in May 1974. In 1991, the Raytheon Company constructed the first entirely electronic toll road, Highway 407, in Toronto, Canada. RFID tags were installed in vehicles that used the 407 Express Toll Way, and RFID readers were positioned along the toll road. When a vehicle goes by these readers, the readers ask for the tag's identification number, which is linked to the car and a bank account. As the tag reply for the reader's request, the toll amount is debited automatically from the related bank account. In 1991, Oklahoma started the initial hybrid electronic toll way, which used RFID technology for express lane tolls. All other tolls were gathered using the typical tollbooths [4].

2.3 Frequency and Wavelength

Frequency and wavelength are the major characteristics of radio waves. Each frequency is exclusively related to a particular wavelength. A lower frequency corresponds to a longer wavelength, and a higher frequency corresponds to a shorter wavelength. The frequency bands used by RFID systems and their associated wavelengths are outlined in Table 1.

From Table 1, it can be seen that low frequency waves are much longer than other frequency waves. Longer waves have the ability to go around barriers but require more power.

Shorter waves require less energy but are blocked by opaque substances. The antenna is the deciding factor of the frequency of the radio wave.

TABLE 1 [2]

FREQUENCY BANDS AND ASSOCIATED WAVELENGTHS USED BY RFID SYSTEMS

Frequency Band	Frequency Range	Wavelength
Low Frequency	9–135 kHz	2300 m
High Frequency	13.553–15.567 MHz	22 m
Amateur Radio Band	430–440 MHz	69 cm
Ultra High Frequency	860–960 MHz	33 cm
Microwave Frequency	2.4–2.4835 and 5.8 GHz	12 cm

2.3.1 Low Frequency RFID Systems

As outlined in Table 1, the range for low frequency is 9–135 kHz. Low frequency RFID systems typically use a frequency range of 125–134 kHz. Low frequency communications are standard around the globe and are administered by the International Organization for Standardization (ISO) specification 18000-2. The oldest RFID systems in existence are the low frequency systems. Radio waves in these systems are capable of penetrating opaque substances. Low frequency tags have a read range of approximately 20 inches and the capability to store up to 60 characters. Since tag read rates are very slow, these low frequency RFID systems are mainly used for access control, animal tagging, and vehicle immobilizers [2].

2.3.2 High Frequency RFID Systems

As outlined in Table 1, the range for the high frequency is 13.553–15.567 MHz. High frequency communications are administered by ISO/IEC (International Electrotechnical

Commission) specification 18000-3, ISO/IEC specification 15693, and ISO/IEC specification 14443, parts A and B. This technology is steadier, and the RFID tags are less expensive than the low frequency tags. These systems are mainly used in libraries, smart cards, luggage control, access control, apparel management systems, and biometric identification systems. DHL, one of the major shipping companies, has successfully implemented item-level tagging using 13.56 MHz tags, and it is prepared to tag more than one billion packages shipped each year [2].

2.3.3 Amateur Radio Band RIFD Systems

As outlined in Table 1, the range for the amateur radio band is 430–440 MHz. This frequency range is described as the "optimal frequency for global use of Active RFID." The amateur band has the capability of propagating around obstructions, for example vehicles, containers, etc., since its wavelength is about one meter. Another advantage of amateur radio band is that it requires very low power. That is, for communication of 100 m, amateur radio band uses only 1 mW, compared to 100 mW or more used in an ultra high frequency system. Amateur radio band RFID systems have been utilized by the U.S. Department of Defense for equipment tagging for more than ten years [2].

2.3.4 Ultra High Frequency RFID Systems

The range for ultra high frequency is 860–960 MHz. Ultra high frequency communications are administered by ISO/IEC specification 18000-6 and EPCglobal Gen 2 specification. UHF RFID tags are enormously utilized in electric toll collections, asset management applications, and supply chain applications for pallet and box tagging since they are very inexpensive to build. UHF tags have the capability to store 8,000 characters, have a read range of four to five meters, and are available as both active and passive tags. The main issue

with UHF tags is the contradictory frequency applications around the globe [2]. Table 2 outlines the ultra high frequency regulations for RFID in some countries.

TABLE 2 [5]

UHF REGULATIONS FOR RFID IN SOME COUNTRIES

Country	Frequency	Power	Technique
Australia	920–926 MHz	4W eirp	
	918–926 MHz	1W eirp	
Canada	902–928 MHz	4W eirp	FHSS
China	840.5–844.5 MHz	2W erp	FHSS
	920.5–924.5 MHz	2W erp	FHSS
France	865.6–867.6 MHz	2W erp	
Germany	865.6–867.6 MHz	2W erp	
India	865–867 MHz	4W eirp	
Italy	865.6–867.6 MHz	2W erp	
Japan	952–954MHz	4W eirp	LBT
	952–955MHz	20mW eirp	LBT
New Zealand	864–868 MHz	4W eirp	
Russian Federation	865.6–867.6 MHz	2W erp	
United Kingdom	865.6–867.6 MHz	2W erp	
United States	902–928 MHz	4W eirp	FHSS

Note: The "Power" column specifies the maximum power available to RFID applications. It is expressed either as eirp (effective isotropic radiated power) or erp (effective radiated power) and 2 Watts erp is equal to 3.2 Watts eirp. The "Technique" column specifies the reader-to-tag communication methods: frequency hopping spread spectrum (FHSS) and listen-before-talk (LBT).

To alleviate these inconsistencies in radio frequency distributions, the EPCglobal Gen 2 standard requires readers that are capable of reading tags across the whole UHF band. Different countries have different read rates since they use different bandwidths and communication methods. Since the United States has a wider bandwidth, the UHF tag read rate in United States is about 1,600 tags per second, while in Europe, it is 600 tags per second. The frequencies are divided into 200-kHz channels due to regulations. This creates 15 channels in Europe and 200 channels in North America, since North America has a wider bandwidth. Having more channels considerably decreases channel conflicts, thus allowing readers in North America to work faster than readers in Europe [2].

2.3.5 Microwave Frequency RFID Systems

As outlined in Table 1, the frequency of a microwave is 2.45 GHz. Microwave frequency communications are administered by ISO/IEC specification 18000-4. This same frequency is also used for cordless phones, medical equipment, and microwave ovens, thus creating the possibility for interference among these devices and RFID systems. Microwave tags have a faster read rate than even UHF tags but are more vulnerable to degradation from solid substances. Microwave frequency RFID tags have the capability to store up to 16,000 characters, and microwave active tags have a read range of 100 meters, while passive tags have a read range of 10 meters. Microwave RFID tags are frequently employed for real-time asset tracking and in electronic toll-collection applications [2].

Table 3 summarizes the characteristics of the different frequencies mentioned above.

TABLE 3 [2]

FREQUENCY CHARACTERISTICS SUMMARY

	LF	HF	Amateur Band	UHF	Microwave
Frequency	9–135 kHz	13.553–15.567 kHz	430–440 MHz	860–930 MHz	2.4—2.4835 GHz and 5.8 GHz
Typical applications	Aeronautical and marine communication	Smart cards, personal identification	Baby monitors, amateur radio		Microwave ovens, cordless telephones, 802.11 wireless computer networks
Standards	ISO/IEC 18000 Part 2	ISO/IEC 18000 Part 3, ISO 15693, ISO 14443 Parts A and B	ISO/IEC 18000 Part 7	ISO/IEC 18000-6, EPCglobal Gen-1 and Gen-2 standards	ISO/IEC 18000-4
RFID applications	Animal tagging, access control, vehicle immobilizers	Access control, payment ID, item level tagging, luggage control, biometrics, library books, laundries, apparel, pharmaceuticals	Active tags identifying containers, vehicles, other RTLS applications	Supply chain (case and pallet level), asset management, and access control	Security, access control, work tracking for factory automation, RTLS
Opaque materials	Not susceptible	Somewhat susceptible	Somewhat susceptible	Very susceptible	Very susceptible
Read rates	Slow		Fast	Fast	Very fast
Read range	20 inches	1 meter	30 meters	4-5 meters	10 meters

2.4 RFID System Components

An RFID system consists of various components that work collectively to identify, track, and make use of data and information. The major components of an RFID system are RFID readers, RFID tags, and middleware. Tags are very tiny radio transponders, which can store and transmit data. Data that is stored in the tag includes an identification number to recognize the object. RFID tags broadcast their data when they are in the surrounding area of a reader. The reader gathers the tag data details and sends it to a host computer, which stores the data and uses it in an application program. Figure 1 shows a simplified representation of an RFID system.

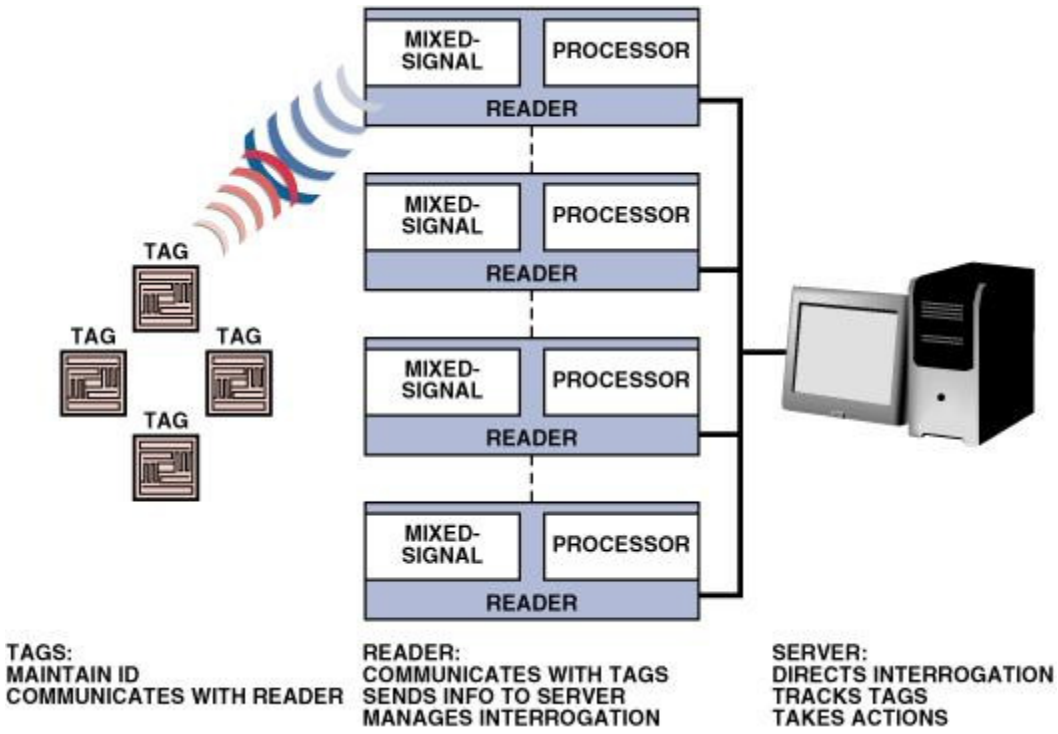


Figure 1: Simplified Representation of an RFID System [6]

2.4.1 Tags

RFID tags are tiny objects that can be attached to merchandise, animals, or people to identify and track using radio waves. Tags are comprised of an integrated circuit (IC) and an antenna, which are the necessary components, and memory, the optional element. An integrated circuit consists of a microprocessor, memory, and a transponder. The microprocessor is used to process the details received from the reader and to access the memory to supply the unique identification number for the tag. The purpose of the antenna is to communicate with the reader and to expand the communication range. Different antenna types are designed to suit various environments and different tag applications. Memory is used to remember details sent by the readers to the tag. RFID tags are primarily categorized into active tags, passive tags, semi-active tags, and semi-passive tags. Read-only tags and read-write tags are another categorization of RFID tags [4].

2.4.1.1 Active Tags

Active tags have a battery to provide the required power. They do not need to be within the reader power field to be detected since they continuously beep. One of the advantages of active tags over passive tags is that they can be read from a longer distance since they have higher signal power due to the built-in battery. The battery life of active tags depends on the number of beeps in a given period—the greater the number of beeps, the lesser the battery life. The disadvantages of actives tags are the cost and size due to the internal battery [4]. An active tag is shown in Figure 2.

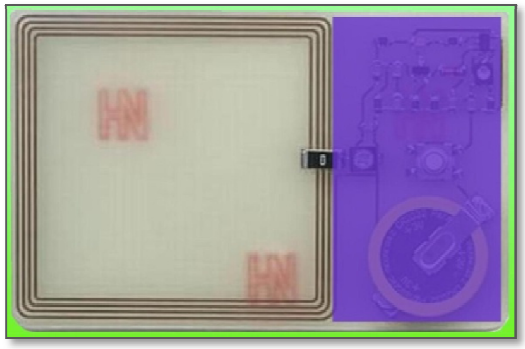


Figure 2: RFID Active Tag [7]

2.4.1.2 Passive Tags

Passive tags are activated with the power supplied by the radio wave of the reader, and they do not have internal power supplies. When a passive tag moves into the range of the radio frequency wave field of the reader, it utilizes that energy to power itself up and talk with the reader. Inexpensive to build, small in size, and no need for an internal power source are the advantages of passive tags. The disadvantage of passive tags is that they have a limited read range. Different antenna types are designed to suit various environments and different tag applications [4]. A passive tag is shown in Figure 3.

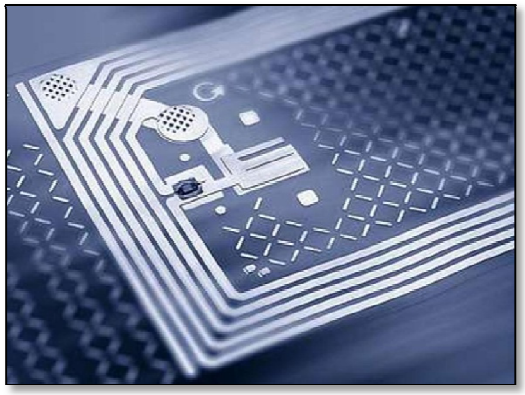


Figure 3: RFID Passive Tag [8]

2.4.1.3 Semi-Active Tags

A semi-active tag is a combination of an active tag and a passive tag. The main reason to manufacture semi-active tags is that the use of active tags is restricted by the life of the batteries that power them, and passive tags have a limited read range. Additionally, the frequent beeping of active tags severely decreases the life of the battery. As a result, active tags do not last for a sufficient amount of time. The passive part of a semi-active tag becomes energized when it approaches the electromagnetic field of the reader and activates the active part of the tag to send RFID signals. The advantages of the semi-active tag are that it has a larger read range and greater battery life. The battery is utilized only when activated by the passive component of the tag, and after a preset amount of time, the battery goes into sleep mode [4].

2.4.1.4 Semi-Passive Tags

Semi-passive tags have a built-in battery but are not used for RFID communication with a reader. The battery is used only to power the internal circuit and peripheral functionality of the tag. Therefore, the tag utilizes all the energy gathered from the radio wave of the reader to broadcast its information. As a result, semi-passive tags have a longer read range than passive tags [4].

2.4.1.5 Read-Only and Read-Write Tags

RFID tags also can be categorized as read-only tags and read-write tags. Read-only tags are programmed with their unique identifier at the time of initial setup or at the time they are manufactured. These tags can only be read by the reader, and the communication between the tag and the reader is unidirectional. Read-write tags allow the reader to read data from the tag as well as write data to the tag any number of times. In other words, read-write tags are reprogrammable and have a memory to store the data sent by the reader. The use of read-only or read-write tags is dependent on the application in which they will be used. For example, if tags are used to identify goods in a store, then typically read-only tags will be used. Otherwise, malicious readers will be able to change the information, especially the price of the product, stored in these tags [4].

2.4.2 Readers

The bridge that connects the RFID tag and the RFID middleware is the RFID reader. The three main components of an RFID reader are the antenna, the radio frequency electronics module, and the controller electronics module. The antenna and radio frequency electronics module is used for communication with the RFID tag, and the controller electronics module is used for communication with the middleware. The functions of an RFID reader are to read data from an RFID tag, write data to the tag, pass on data to and from the RFID middleware, power up the tag, use anti-collision procedures to guarantee concurrent read-write communication with many tags, authenticate tags to stop fraud or illegal access to the system, and encrypt data to guard the integrity of data [3]. RFID readers are mainly categorized according to their communication interface and mobility as serial readers, network readers, stationary readers, and handheld readers [4].

2.4.2.1 Serial Readers

Serial readers utilize a Recommended Standard 232 (RS-232) serial port to transmit information or commands performed by the user or the application of the host system. The main advantage of serial readers is that the serial port connections are very reliable. Two disadvantages are a lower information transfer speed and limited cable length [4].

2.4.2.2 Network Readers

Network readers have the capability of connecting to wired or wireless host systems, which make them, network devices. Network readers have no cable-length limitations. However, the connection is not very reliable, as in serial readers [4].

2.4.2.3 Stationary Readers

Stationary readers are typically mounted on walls, gateways, or any appropriate structure in the read area. Sometimes, these readers are mounted on moving objects like forklifts and trucks. Stationary readers contain external antennas. An example of a stationary reader type is the RFID printer, which is capable of printing bar codes and writing on its RFID tag [4].

2.4.2.4 Handheld Readers

Handheld readers have built-in antennas and the capability of functioning as handheld units [4]. A handheld RFID reader is shown in Figure 4.

Figure 4: RFID Handheld Reader [9]

2.4.3 RFID Middleware

RFID middleware is used to manage the flow of data between RFID readers and enterprise applications. RFID middleware moves the information stored in a tag from the RFID reader to the appropriate enterprise system in a tag-read process, and moves the information from the enterprise system to the appropriate RFID reader and eventually to the correct tag in a tag-write process [3]. Figure 5 shows the relationship between RFID readers, RFID middleware, and enterprise applications.

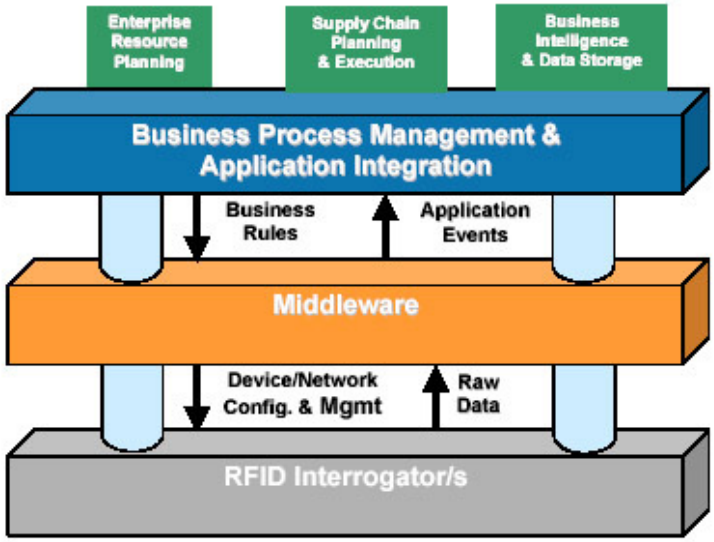


Figure 5: RFID System and Enterprise Applications [10]

The four major functions of RFID middleware are data collection, data routing, process management, and device management.

In the data collection process, the main responsibilities of RFID middleware are the extraction, aggregation, smoothing, and filtering of information that is received from many readers in an RFID system. RFID middleware extracts the necessary data from a large amount of raw data that has been gathered by RFID readers for the managerial process in the enterprise systems. This filtering process of RFID middleware protects the enterprise systems from

becoming overwhelmed by the flow of data. When Wal-Mart shifted to item-level tagging, two terabytes of raw data were produced each second [3].

In the data routing process, RFID middleware assists in the integration of RFID networks with enterprise systems by sending information to the proper enterprise systems within the institution. Some of the information gathered by RFID readers may be input to a warehouse management system to keep track of inventory, and the rest of the information may be sent to other applications in the enterprise system; RFID middleware decides which data goes where [3].

RFID middleware can be utilized to generate events according to company policies. When a purchase is made on an organization website, the enterprise system accountable for initiating the delivery passes the purchase order on to the RFID middleware. Then the RFID middleware locates the exact dock door where the pallet is sitting and writes the delivery details on its tag. RFID middleware is also used to administer unauthorized shipments and unanticipated inventory, such as low-stock items and out-of-stock items [3].

RFID middleware is also utilized for device management. Large companies have hundreds of thousands of different kinds and brands of RFID readers all over their network, and RFID middleware is used to monitor and coordinate these RFID readers very efficiently. RFID middleware also allows for remote management of RFID systems [3].

2.5 Barcodes vs. RFIDs

A barcode is an automatic identification technology that uses laser light as the data carrier. Therefore, it is also referred to as optical technology. The barcode scanner reads the numbers and characters that are placed in a pattern of vertical bars, squares, spaces, and dots; converts the pattern into information; and broadcasts that information to a host computer for processing. Barcodes are primarily used to identify books, documents, products, shipments,

equipment, tools, etc. The major advantages in using barcodes are fast data collection, more accuracy than manual data entry, less cost, no international restrictions, no privacy issues, and human readability, since the numbers and characters are printed at the bottom of the barcode [2].

RFID technology is similar to barcode identification technology, but it has more advantages. RFID tags are reprogrammable. That is, the data on the tag can be changed, or new data can be added at any time. However, with barcodes, data cannot be modified once they are printed. RFID read-write tags also have the ability to record environmental data like radiation, moisture, temperature, or even the time of tampering. RFID tags can be programmed to perform calculations and logical operations. An active RFID tag has the capability to operate as a small computer, whereas a barcode is simply a fixed-storage method. RFID tags can be identified without human involvement since they do not require line of sight for detection. However, barcode scanners require a top quality line of sight to read labels. Only one label can be read at a time with a barcode, but with RFID, many tags can be read simultaneously. RFID tags also have a greater read range, higher read rate, and larger storage capacity compared to barcodes. RFID tags have a greater survivability than barcodes. That is, RFID tags have the capability to function when buried in dirt; covered with mud, grease, or snow; or even painted over. Barcode data is not protected. Anyone can easily scan a barcode or read the printed characters. But RFID provides greater data security using passwords and encryption to protect data from unauthorized access [2][3].

2.6 Applications of RFID

RFID has the ability to identify, locate, track, and monitor objects and people automatically. As a result, RFID applications have been developed in every industry and service. This section illustrates some of the common applications for RFID.

2.6.1 Toll Roads

Toll roads are one of the most widespread RFID applications used daily by millions of people. The RFID reader installed in the fast lane reads the RFID tag data attached to the windshield of the cars. Therefore, the vehicle does not have to stop at the toll booth to pay the toll. The reader then passes the RFID tag data along with a unique identification number to an RFID-enabled system. The RFID system correlates the tag identifier with the correct vehicle using a database and finally bills the toll amount for that particular vehicle [4].

2.6.2 Newborn Infant Tracking

A major issue that many hospitals face is the possibility of inadvertently mismatching newborns with their parents. This issue has raised major legal liabilities and generated enormous emotional pain for parents. By using RFID technology, hospitals offer a better method to guarantee that the right newborn baby is given to the right parent. Protected wristbands with RFID tags are affixed to the baby and the mother. Then, unique identifiers of the baby's and mother's RFID tags are linked together in the database. This process ensures that each baby is placed with the correct mother by matching the unique identifiers of the RFID tags [4].

2.6.3 Gasoline Payment

RFID has created a more convenient method to pay for gas. The RFID reader installed at gas pumps reads the consumer's unique RFID tag identifier and transfers the data to the back-end information system. The information system then matches the unique tag identifier with the database, relates it to the person at the gas pump, and supplies all the details to finish the sale. Customers can update their profiles beforehand to indicate what credit card they want used and whether they want a receipt at the end of the sale [4].

2.6.4 Attraction Park Children Positioning

A major issue facing parents is the possibility of losing their children in public places. A number of attraction parks have set up a child tracking system using RFID to overcome this issue. A bracelet with an RFID tag is affixed to the child's wrist, and RFID readers are placed tactically around the park grounds in order to provide full coverage so that the child can be tracked easily. These details are well protected by the information system and are only given to the person in charge of the child [4].

2.6.5 Public Transportation

Public transportation by bus is one of the major applications of RFID. Conventional paper tickets are now replaced by contactless smart cards. Passengers are not longer required to have the exact change or money since smart cards have the ability to be loaded with a large amount of money. The customer no longer needs to know the exact price of the fare, as the system automatically debits the correct price from the card. Also, customers can use the same card for all transportation routes. Even if ticket prices change, the prepaid contactless cards remain valid, and the transportation company does not need to print new tickets. Since no tickets are sold on the bus, the driver will be less distracted and not be required to carry cash in the vehicle. Customers are even able to get discount rates since accurate data is obtained automatically [11].

2.6.6 Ticketing

RFID systems are commonly used for airline ticketing purposes. The Lufthansa "ticketless flying" project replaced the conventional paper ticket and boarding pass with new contactless smart cards. Passengers book their flights using an RFID-enabled unique card number. Then an electronic ticket, associated with the customer's personal data along with the

flight details, is generated and saved in the system. At the terminal, the customer only needs to have the smart card on his/her person, even without removing it from a wallet or briefcase, in order to obtain a printed, detailed receipt. Even at the boarding gate, the customer only needs to present the smart card. This convenient method saves considerable time for customers as well as airlines, and reduces labor-intensive management and verification tasks for the airlines [11].

2.6.7 Access Control

RFID is used to automatically check authorization for individuals to gain access to certain premises. When a person uses his/her RFID-enabled card, the RFID reader sends the card details to a central database. Based on permissions for that particular user in the database, access is either granted or denied. The database administrator has the capability to grant or deny access for each individual, for each entrance to each premise. Any modifications to an individual's access permission can be done by a single entry in the central database. The administrator can even remotely perform the required changes, thus saving a considerable amount of time. Temporary access cards can be disabled automatically after a given period, thus preventing security issues involving unauthorized access. Hotels use RFID-enabled cards to grant permission to enter rooms, gyms, lounges, etc., and also to use in hotel restaurants. The same access-control RFID card can also be used for time keeping since it logs time when a person scans the card [11]. Vehicle companies have used RFID in car keys so that when the owner nears the vehicle, the doors automatically open and the vehicle starts. In addition, the vehicle automatically locks as soon as the owner steps away from it [12]. A most unusual RFID application for access control is where a person implanted an RFID chip in his hand to permit him access to his home and office, and to even start his vehicle without using a key [13].

2.6.8 Animal Identification

RFID has allowed the rancher to obtain the exact number of his/her cows automatically at a distance by using a mobile phone instead of counting each cow on the ranch. An RFID tag, affixed to the animal's ear, stores details about the animal—breed, pedigree, birth date, vaccination details, and temperature profile. RFID readers placed on the farm read data from these tags and transfer it to a mobile phone, laptop, or personal digital assistant using a wireless connection. These animals are tracked from the ranch to auction, distribution, and even butchering. This method also assists ranchers in obtaining the history of sick animals within a short period of time. RFID-tagged meats allow consumers to get details about the rancher, nutrition facts, cooking methods, etc. Today, many animals are tagged, including cats and dogs. Tags can store not only a pet's identification number but also its unique identifier, name, address, and vaccination records [4].

2.6.9 Vehicle Immobilization

Almost all new manufactured vehicles are now equipped with RFID-based anti-theft systems. A vehicle immobilizer is a security gadget installed in a vehicle to make it difficult for unauthorized users to have access to the vehicle. Typical anti-theft systems sound the horn and flash the headlights when the vehicle has been illegally accessed. However, the RFID anti-theft system immobilizes the engine, prohibits the flow of fuel, impedes the electrical supply, and stops all other moving parts, when it detects an unauthorized access. RFID readers that are installed in vehicle door locks and the ignition read the RFID tagged keys and ensure that the valid key is used to open the door and start the engine. Otherwise, readers will prevent mobilization of the vehicle and send an alert to the owner of the vehicle regarding unauthorized access [4].

2.6.10 Health Care

Many health care organizations are already using RFID technology due its enormous ability to improve performance, responsiveness, and accountability of health care systems. All patients, visitors, and staff are given a smart card with an embedded RFID chip. RFID readers are installed at various locations to track the position of these people. This method saves a considerable amount of time in finding a person's exact location. Another method is to affix an RFID bracelet or wristband to the patient. The patient's name, birth date, gender, surgical information, medications, and billing details are all stored in this bracelet or wristband. The eventual benefit of RFID bracelets is to decrease the risk of misidentifying patients and to access the patient's correct records quickly. Another method is to implant an RFID chip in a patient's arm. This allows recognizing the patient's full identity and medical records in an emergency situation, even outside the hospital premises [4].

2.6.11 Asset Tracking and Management

The major advantage of RFID technology is that it does not require line of sight for communication. As a result, RFID applications have become the main technique to identify and track objects, animals, and even humans. Some of these RFID applications involve the tracking of various items including the following: airline baggage; shipping containers; library books; toxic waste; manufacturing and assembly items; pallets in a container and items in a pallet; inventory items in stores, hospitals and warehouses; prisoners; children at schools; and babies at homes.

2.7 RFID Standards

Standards guarantee that products have at least certain features, such as quality, interoperability, reliability, and safety. Standards can be divided into technology standards and

application standards. Technology standards specify the fundamentals of how things work. For example, RFID technical standards specify how frequency, communication protocols, and data transfer works. Conversely, application standards classify how a technology is used. At present, there are numerous groups working towards standardization of RFID. The main organizations creating standards in the RFID field are the International Organization for Standardization and EPCglobal Inc.

2.7.1 International Organization for Standardization

The world's largest organization that develops and publishes international standards is the International Organization for Standardization. The ISO headquarters, located in Geneva, Switzerland, is a combination of the national standards organizations of 161 countries. It is a non-governmental organization that creates a link between private and public divisions [14]. The ISO primarily works in the development of RFID technological standards [15]. Table 4 shows several ISO standards for RFID.

TABLE 4 [16]

SEVERAL ISO STANDARDS FOR RFID

ISO/IEC 14443	Identification cards -- Contactless integrated circuit(s) cards -- **Proximity cards** Part 1: Physical characteristics Part 2: Radio frequency power and signal interface Part 3: Initialization and anticollision Part 4: Transmission protocol
ISO/IEC 15693	Identification cards -- Contactless integrated circuit(s) cards -- **Vicinity cards** Part 1: Physical characteristics Part 2: Air interface and initialization Part 3: Anticollision and transmission protocol
ISO/IEC 18000	RFID for Item Management Part 1: Defines the foundation for all air interface definitions in the ISO/IEC 18000 series. Part 2: Parameters for air interface communications below 135 kHz Type A (FDX): 125 kHz Type B (HDX): 134.2 kHz Part 3: Parameters for air interface communications at 13.56 MHz Part 4: Parameters for air interface communications at 2.45 GHz Passive tag operating as an interrogator talks first Battery assisted tag operating as a tag talks first. Part 6: Parameters for air interface communications at 860 MHz to 960 MHz Type A and type B with the primary difference being the anti-collision algorithm used. Type C - also know as EPCglobal Class 1 Gen 2. Part 7: Parameters for active air interface communications at 433 MHz

Note: ISO/IEC indicates a combined ISO and IEC publication.

2.7.2 EPCglobal

EPCglobal is the most powerful organization in RFID. In 1999, a group of vendors and manufacturers decided to create a worldwide, industry-driven standard for RFID, known as the Electronic Product Code (EPC) [17]. EPC is the unique identification number (UID) for each RFID tag around the globe. EPC assists in identifying the manufacturer, product, and serial number of tagged objects [17]. Figure 6 shows an Electronic Product Code.

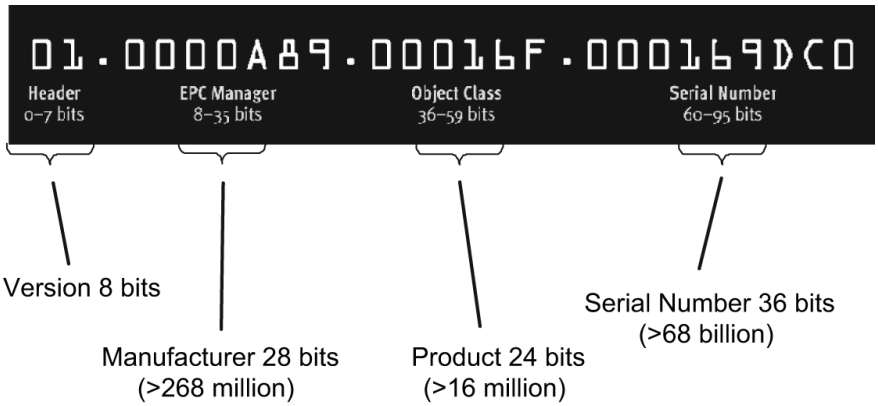


Figure 6: Electronic Product Code [18]

The group financed the research of Auto-ID Labs at the Massachusetts Institute of Technology (USA), University of Cambridge (United Kingdom), University of St. Gallen & ETH Zurich (Switzerland), Fudan University (China), Information and Communications University (South Korea), University of Adelaide (Australia), and Keio University (Japan) [19]. The Auto-ID Center produced the original ideas for implementing the "Internet of Things" [2]. They realized that research work needed to be standardized to make it successful. As a result, EPCglobal was formed in 2003 [17] to develop standards for the EPC to uphold the use of RFID. EPCglobal is an organization comprised of industry leaders such as Cisco Systems, LG Electronics, DHL Solutions, Lockheed Martin Corporation, Sony Corporation, Wal-Mart Stores Inc., and so on [20][21]. Some of the EPCglobal standards are listed below [22]:

- EPC Tag Data Standard
- EPC Tag Data Translation Standard
- Class-1 Generation-2 UHF Air Interface Protocol Standard
- Reader Protocol Standard
- Application Level Events (ALE) Standard
- Object Naming Service (ONS) Standard

- EPCglobal Certificate Profile Standard
- Reader Management Standard
- Drug Pedigree Standard
- EPCIS Standard

EPCglobal has defined a class structure for RFID tags, which consists of five classes, as shown in Table 5.

TABLE 5 [2][4]

EPCGLOBAL TAG CLASSIFICATIONS

Class	Range	Description
0	< 3m	Read-only UHF passive tags. Offers basic RF passive ability. Number programmed into tag during production. Class 0 tags added after other classes were published.
1	< 3m	UHF or HF passive tags. Field programmable Write-Once-Read-Many (WORM). Newer standard is UHF Generation-2.
2	< 3m	Passive read-write tags. Most flexible type of passive tag with data-encryption functionality. Data can be added to the tag by a qualified reader at any point in the supply chain.
3	< 100 m	Read-write with on-board sensors and batteries. Can record parameters like temperature, pressure, and motion, by writing into the tag's memory. Provides broadband communication. Either semi-passive or active.
4	< 300 m	Read-write active tags with integrated transmitters and on-board batteries. Like miniature radio devices that can communicate with other tags and readers.
5	Unlimited	Fundamentally readers with on-board batteries and AC/DC connection support. Can provide power to other tags and communicate with devices other than readers.

Because no agreements regarding RFID standards existed between ISO and EPCglobal, EPCglobal presented its UHF Class-1 Generation-2 standard to be included as an ISO standard, and the ISO incorporated this standard as ISO 18000 in June 2006. As a result, the EPCglobal Class-1 Gen-2 standard became equivalent to the ISO/IEC 18000-6 type C standard (ISO 18000-6C) [15]. Table 6 shows the RFID standards used for each radio frequency.

TABLE 6 [16]

RFID STANDARDS APPLIED TO EACH RADIO FREQUENCY

Frequency / Standard	LF 125/134.2 kHz	HF 13.56 MHz	Amateur 433 MHz	UHF 860 - 960 MHz	Microwave 2.45 GHz
ISO	ISO 11784 ISO/IEC 18000-2A ISO/IEC 18000-2B	ISO/IEC 14443 ISO/IEC 15693 ISO 18000-3	ISO 18000-7	ISO 18000-6A ISO 18000-6B ISO 18000-6C	ISO 18000-4 ISO/IEC 24730-2
EPCglobal				Class 0 Class 1 Class 1 Gen 2	

2.8 Aloha Protocol

Aloha is one of the oldest mechanisms used to resolve the channel-share problem. In the 1970s, Abramson and colleagues at the University of Hawaii developed the Aloha system to interconnect the terminals of the campuses of the University of Hawaii. Pure Aloha and Slotted Aloha are its two variants. There is no time synchronization with Pure Aloha. But time is divided into distinct slots with Slotted Aloha [23].

2.8.1 Pure Aloha Protocol

The fundamental idea of Pure Aloha is that users transmit immediately whenever there is data to be sent. But when more than one user transmits at the same time, the frames become

damaged due to collisions. Ultimately, these simultaneous transmissions result in packet loss. The sender has the capability to discover whether a frame was damaged by listening to the medium for acknowledgement from the receiver. If the transmitted frame was destroyed due to collision, the sender waits a random amount of time and retransmits the frame. To avoid the same frames colliding multiple times, the user's waiting time was set to a random value [23].

2.8.2 Slotted Aloha Protocol

Roberts implemented a technique in 1972 to double the capacity of the Aloha system. This new technique is known as Slotted Aloha. He divided time into distinct intervals, and each interval corresponds to a single frame. Users are not allowed to transmit whenever they have data. Instead, they must wait until the beginning of the next time slot. Slotted Aloha has cut the vulnerable period in half by restricting the possible transmissions into time slots [23]. With Slotted Aloha, frames either collide fully or not at all. If the transmitted frame was destroyed due to collision, the sender waits a random amount of slots and retransmits the frame at the beginning of another slot [24]. Figure 7 shows the vulnerable period for Pure Aloha and Slotted Aloha.

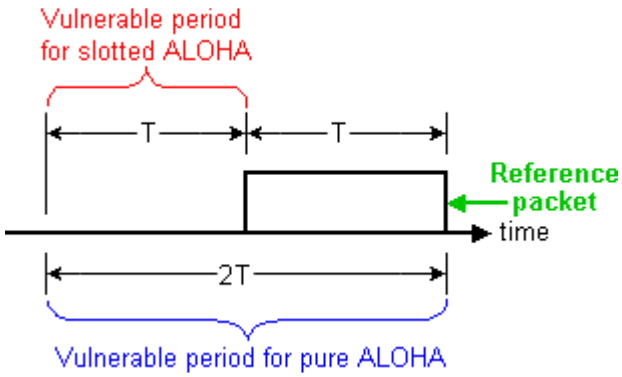


Figure 7: Aloha Vulnerable Period [25]

2.9 RFID Collisions

An RFID system consists of many RFID tags and readers. RFID has the capability of scanning multiple objects simultaneously. However, readers should be able to identify each

individual tag separately from the hundreds of tag responses in order to read the data stored in them. When several tags reply to a reader query at the same time, contradictory communication signals might cause interference, or collision. Collisions happen when readers and tags functioning on the same channel transmit simultaneously. If there is only one reader and only one tag in the RFID system, then they are able to communicate without any collisions. But if a single RFID reader is communicating with multiple RFID tags or more than one reader is operating in a closer range, then there will be collisions, and the end result will be a failed transmission. The three collision types that exist in an RFID system are tag-tag collisions, reader-tag collisions, and reader-reader collisions [26].

2.9.1 Tag-Tag Collisions

Tag-tag collisions occur when multiple RFID tags reply to the same RFID reader at the same time. Interference occurs at the RFID reader since the reader is receiving many replies from the tags simultaneously. Therefore, the reader is unable to read the tag responses due to collisions. The ultimate result is that the reader is not able to identify all tags in its interrogation region [26].

2.9.2 Reader-Tag Collisions

Reader-tag collisions occur when signals sent by the RFID reader collide with the replies sent by the RFID tags in its interrogation region. Reader-tag collisions can also occur when the RFID tag response of another RFID reader interferes with the radio signal of a nearby RFID reader [26].

2.9.3 Reader-Reader Collisions

Reader-reader collisions happen when two or more RFID readers operate in very close proximity. Queries sent by an RFID reader to tags in its region collide with queries sent by a

nearby RFID reader to the tags in its region. Reader-reader collisions can also occur when an RFID tag hears queries from multiple readers at the same time. For example, when an RFID tag receives queries from an RFID reader in its zone as well as queries from an RFID reader in a nearby zone, then reader-reader collisions occur at the receiving tag. As a result, the RFID tag is unable to reply to queries from any of the readers [26].

2.10 RFID Anti-Collision Protocols

The method that readers and tags use to avoid collisions is known as an anti-collision protocol or algorithm. Anti-collision protocols allow multiple items to be identified in the same shared frequency simultaneously. An anti-collision protocol is an essential part of a successful RFID system since it permits the RFID reader to identify and communicate with all RFID tags in the reader zone. These protocols should perform a quick and accurate identification of all tags present in the area because the tagged objects might leave the reading zone of the RFID reader before becoming completely identified. The main anti-collision protocols presently standardized by the ISO are Aloha Anti-Collision Protocol (ISO 18000-6A), Binary Tree Anti-Collision Protocol (ISO 18000-6B), and EPCglobal Class-1 Generation-2 Anti-Collision Protocol (ISO 18000-6C) [15].

Having surveyed the present literature on RFID, the evaluation of the performance of the Slotted Aloha protocol for RFID will be evaluated in the next chapter.

CHAPTER 3

PERFORMANCE EVALUATION OF SLOTTED ALOHA ANTI-COLLISION PROTOCOL FOR RFID SYSTEMS

3.1 Slotted Aloha Anti-Collision Protocol

The purpose of the RFID reader is to recognize all RFID tags in its area in the least amount of time. Nevertheless, delays happen as a result of collisions at the RFID reader when several tags respond at the same time. Aloha-based protocols are one of the main techniques used to minimize collisions.

As described in Chapter 2, Slotted Aloha is a variant of Pure Aloha, and it cuts the vulnerable period or collision interval in half by restricting possible transmissions into time slots. As a result, Slotted Aloha provides a greater throughput than Pure Aloha. The synchronization of all tags is managed by the RFID reader. Therefore, Slotted Aloha is a stochastic, reader-driven time division multiple access (TDMA) anti-collision method [11]. The Slotted Aloha anti-collision protocol is explained below using a practical example.

Assume that there is a single RFID reader with three available time slots and five RFID tags in the reader zone. The RFID reader broadcasts a REQUEST command at repeated intervals. The REQUEST command synchronizes all tags in the zone, notifies tags of the available number of slots, and asks the tags to send their serial numbers (or tag IDs) to the reader in one of the available three time slots. As soon as the tags receive the REQUEST signal, each tag randomly selects a time slot from the three available slots and sends its serial number during the selected time slot. If the reader does not receive any replies from the tags, it waits and sends the REQUEST command again. In this scenario, since there are five tags in the reader zone and only three time slots available, then definitely two or more tags will select the same time slot. When

two or more tags reply to the reader simultaneously in the same time slot, collisions occur at the reader. As a result, the reader will not be able to identify the serial number of any tags, and these tags must be read again. If the reader cannot identify a single tag due to collisions, it waits and resends the REQUEST command. If there was a time slot that only one tag selected, then that particular tag will be able to send its serial number to the reader without any errors. Once the reader receives a tag serial number without any errors, it sends the SELECT command to that particular tag. The SELECT command has the serial number of the tag, and the tag with this serial number is cleared to perform read and write commands. All other tags continue to react to the REQUEST commands from the reader until they are identified successfully [11]. This process is shown in Figure 8.

Reader → Tags	**REQUEST**	Time Slot 1	Time Slot 2	Time Slot 3	**SELECT 10111010**
Tags → Reader		**Collision**	**Collision**	**Success**	
Tag 1			**10110010**		
Tag 2		**10100011**			
Tag 3		**11011011**			
Tag 4			**11110101**		
Tag 5				**10111010**	

Figure 8: Slotted Aloha Anti-Collision Protocol [11]

As the number of tags increases in the reader zone, more collisions will occur, and throughput will decrease severely. Increasing the reader's number of available time slots will overcome this issue. But this maneuver decreases the performance of the anti-collision algorithm since it must listen for all feasible tags for the period of all the time slots, even though there is

only one tag in the reader zone. An improved version of the Slotted Aloha protocol, known as the Dynamic Slotted Aloha was introduced to mitigate this issue [11].

3.2 Dynamic Slotted Aloha Anti-Collision Protocol

The throughput of the Slotted Aloha protocol becomes maximized when the number of time slots of the reader is equal to the number of tags in the zone. With the Dynamic Slotted Aloha protocol, the number of available slots varies. Like the Slotted Aloha, the Dynamic Slotted Aloha broadcasts a REQUEST command at repeated intervals, synchronizing all tags in the zone, notifying tags of the available number of slots, and asking tags to send their serial numbers to the reader in one of the available time slots. But with the Dynamic Slotted Aloha, if the reader receives several replies, then it increases the available number of time slots in the REQUEST commands that follow until a single tag can be identified. To increase performance, once the serial number of a tag is identified without any errors, the reader then sends a BREAK command to block other tags from sending signals that may collide with the data exchange of the identified tag [11].

3.3 Mathematical Analysis

An error-free or collision-free data packet communication has a throughput (S) of "one" for the communication period since all packets will be read successfully by the reader. On the other hand, in all other situations, the throughput is zero, since the reader will not be able to identify a single packet due to collisions. For example, if the packets were nearly collided or fully collided, still the reader would not be able to identify it. The relationship between the average throughput (S) of a communication channel and the offered load (G) for the Pure Aloha anti-collision protocol is [11]

$$S = G \cdot e^{(-2G)} \tag{3.1}$$

From equation (3.1), the maximum throughput of 18.4% at offered load G = 0.5 was obtained with the Pure Aloha anti-collision protocol.

$$S = G \cdot e^{(-2G)}$$

When G = 0.5,

$$S = (½) \cdot e^{(-2*0.5)}$$

$$S = (½) \cdot e^{(-1)}$$

$$S = 1/(2e)$$

$$S = 0.184$$

$$S = 18.4\%$$

With Pure Aloha, more than 80% of the channel capacity is wasted, since the maximum throughput is 18.4%. Pure Aloha is suitable when there are a small number of tags in the reader zone. As the number of tags increases, the offered load increases, and the number of collisions between individual tags instantaneously increases sharply.

The Slotted Aloha protocol cut the vulnerable period in half by restricting the possible transmissions into time slots. Thus, the throughput (S) of the Slotted Aloha anti-collision protocol with the offered load (G) is [11]

$$\boldsymbol{S = G \cdot e^{(-G)}} \qquad (3.2)$$

Using equation (3.2), the maximum throughput of the slotted Aloha anti-collision procedure is 36.8% at offered load G = 1.

$$S = G \cdot e^{(-G)}$$

When G = 1,

$$S = 1 \cdot e^{(-1)}$$

$$S = 1/e$$

$$S = 0.368$$

$$S = 36.8\%$$

Figure 9 shows throughput versus offered load graphs for Pure Aloha and Slotted Aloha anti-collision protocols.

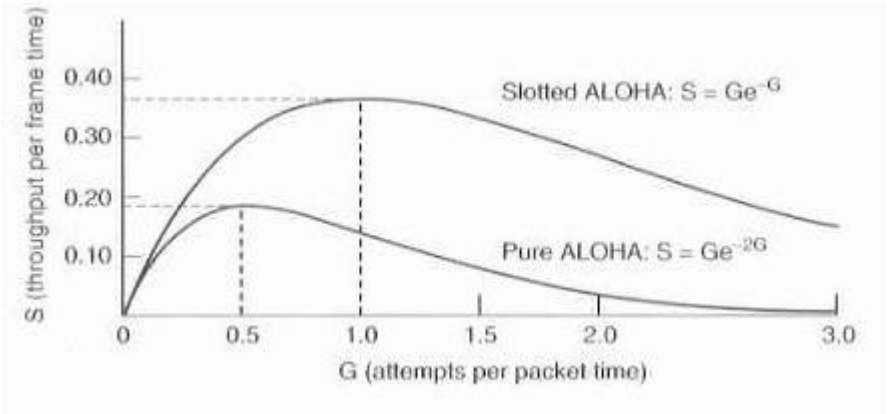
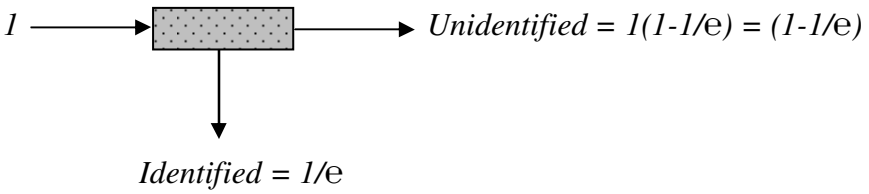


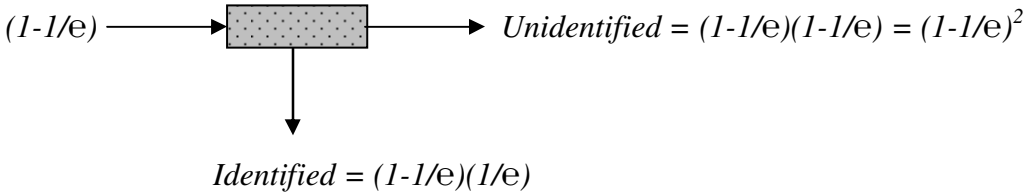
Figure 9: Throughput vs. Offered Load for Pure Aloha and Slotted Aloha [27]

Now the overall number of time slots or the total time needed for the RFID reader to identify all RFID tags (λ) within the reader area is examined. As explained previously, when the size of the frame or the number of time slots is equivalent to the total number of RFID tags, the Slotted Aloha anti-collision protocol offers maximum throughput of $1/e$ or 0.368. Therefore, the fraction of tags in each round left unidentified is $1-1/e$. If the total number of rounds needed to recognize all the RFID tags is r_i [28], then

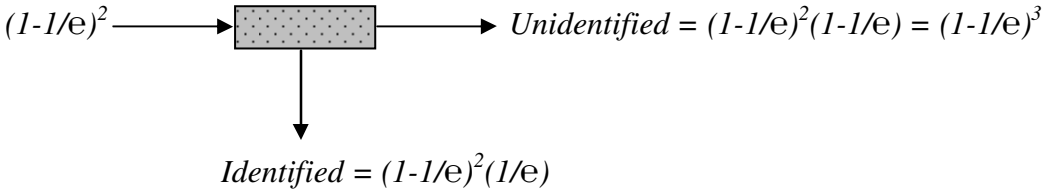
Round r_1:



Round r_2:



Round r_3:

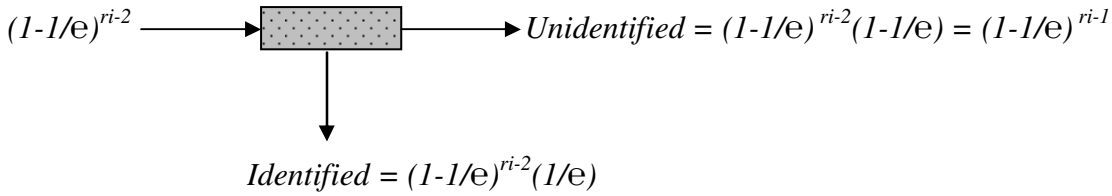


…….. …….. ……..

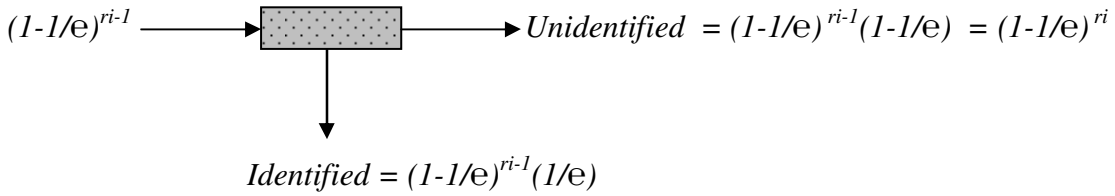
…….. …….. ……..

…….. …….. ……..

Round r_i-1:



Round r_i:



Therefore, $\lambda.(1-1/e)^{ri}$ tags are left to be identified from the initial total λ number of tags. Hence, the number of RFID tags identified after r^{th} round is $\lambda[1-(1-1/e)^{ri}]$. If β is the fraction of RFID tags that needs to be identified, then

$$\lambda[1-(1-1/e)^{ri}] \geq \lambda\beta$$

$$\lambda - \lambda(1-1/e)^{ri} \geq \lambda\beta$$

$$-\lambda(1-1/e)^{ri} \geq \lambda\beta - \lambda$$

$$-\lambda(1-1/e)^{ri} \geq \lambda(\beta-1)$$

$$\lambda(1-1/e)^{ri} \leq \lambda(1-\beta)$$

Therefore, the minimum number of rounds needed, such that, the anticipated number of unidentified RFID tags is less than or equal to $(1-\beta)$

$$\lambda(1-1/e)^{ri} = \lambda(1-\beta)$$

$$(1-1/e)^{ri} = (1-\beta)$$

Therefore, the total number of rounds

$$\boldsymbol{r_i = log_{(1-1/e)}\,(1-\beta)} \qquad (3.3)$$

From equation (3.3), the total number of rounds (r_i) required to identify all the RFID tags with the Slotted Aloha protocol is independent from the total number of RFID tags (λ) to be identified. Therefore [28],

$$r_i = R, \ \forall_i \text{ where } R \text{ is a constant for a given value of } \beta$$

The total number of time slots, $T(\lambda)$, needed to identify all λ RFID tags is the sum of unidentified RFID tags or the total number of frame slots per round. Initially, the number of unidentified tags is equal to the total number of tags. Therefore,

$$T(\lambda) = \lambda + \lambda(1-1/e) + \lambda(1-1/e)^2 + \ldots\ldots + \lambda(1-1/e)^{ri-1} + \lambda(1-1/e)^{ri}$$

$$T(\lambda) = \lambda[1 + (1-1/e) + (1-1/e)^2 + \ldots\ldots + (1-1/e)^{ri-1} + (1-1/e)^{ri}]$$

$$T(\lambda) = \sum_{k=0}^{r_i} \lambda(1-1/e)^k$$

$$T(\lambda) = \lambda \sum_{k=0}^{R} (1-1/e)^k$$

The sum of the geometric series is

$$T(\lambda) = \lambda \left[\frac{1-(1-1/e)^{R+1}}{1-(1-1/e)}\right]$$

$$T(\lambda) = \lambda.e\,[1-(1-1/e)^{R+1}] \qquad (3.4)$$

Note that $(1-1/e) \approx 0.63212$. Therefore, $(1-1/e)^{R+1}$ is negligible, since the power of a decimal number is very small. Rewriting equation (3.4) becomes

$$T(\lambda) \approx \lambda.e\,[1]$$

$$T(\lambda) \approx \lambda.e$$

Therefore, an RFID reader requires $e.\lambda$ time slots to identify λ number of RFID tags with the Slotted Aloha protocol.

$$\boldsymbol{T(\lambda) = e.\lambda} \qquad (3.5)$$

The advantage of this mathematical analysis is the ability to predict the RFID read time for any number of tags. This is helpful when designing RFID systems. Before implementing a

live network, the designer can predict the number of tags that will be identified by a single reader, the number of readers needed, the area a reader will be able to cover depending on the read time, etc.

3.4 Simulation

The RFID Simulator was developed completely from scratch to evaluate the performance of the Slotted Aloha and EPCglobal Class-1 Generation-2 protocols for RFID. The RFID Simulator was designed to replicate a real-life RFID environment. It is extremely difficult to locate a free RFID simulator to perform research or to execute experiments in this area. The delay time to identify RFID tags using Slotted Aloha and EPCglobal Class-1 Generation-2 protocols was obtained from the simulation results, and the performance was compared. The RFID Simulator can be used to imitate hardware and has the capability to calculate the delay to any number of RFID tags. The following sections explain the simulator's features, tests performed, and simulation results.

3.4.1 RFID Simulator

The RFID Simulator replicates a real-life RFID environment and is used to calculate the RFID tag read time for Slotted Aloha and EPCglobal Class-1 Generation-2 protocols. The simulator provides two options for the EPCglobal Class-1 Generation-2 protocol. The first option for the Class 1 Gen 2 protocol provides the time to identify all tags when the RFID reader knows the total number of tags in the range beforehand. The second option for the Class 1 Gen 2 protocol provides the time to identify all tags when the RFID reader does not know beforehand the total number of tags in the range. The RFID Simulator greatly assists RFID system designers since it can be used to mimic hardware. It calculates the delay to any number of RFID tags, which is not possible with real-life RFID systems. RFID system designers can determine the

time it takes to read tags, and according to the delay, they can decide how many readers need to be placed in the system so they can detect all tags without missing any of them. To confirm the integrity of RFID Simulator, the author compared the results of the simulator with mathematical analysis and experimental results. RFID Simulator was designed with the intention of making it scalable and extendable for future users wishing to extend its functionality.

The RFID Simulator provides a graphical user interface (GUI). Figure 10 shows a screenshot of the RFID Simulator.

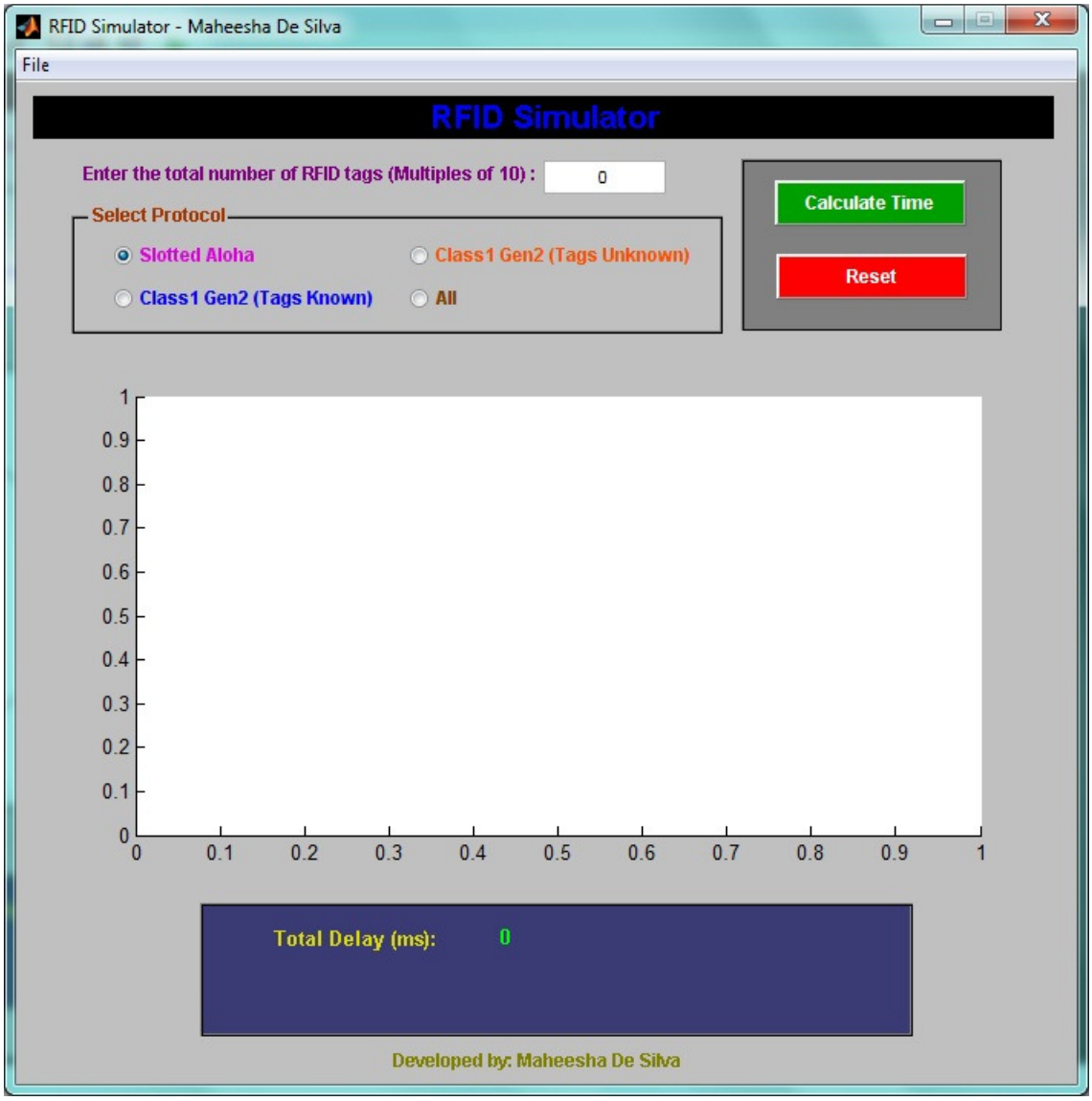


Figure 10: RFID Simulator

The user enters the total number of RFID tags and selects the protocol. The user has the option to select either the Slotted Aloha protocol or the EPCglobal Class-1 Generation-2 protocol with the RFID reader knowing beforehand the number of tags in the range, or the EPCglobal Class-1 Generation-2 protocol with the RFID reader not knowing beforehand the number of tags in the range, or all protocols. The simulator then runs the specified protocol or protocols to identify the number of RFID tags and displays the total time or delay that is needed to identify all specified tags. The simulator also displays a graph with the number of tags and delays to offer a better visualization of the results to the user. When the user selects "All" as the protocol, the simulator displays graphs for each protocol to allow the user to compare the results. The RFID Simulator was developed using the MATLAB high-level language software.

3.4.2 Why MATLAB?

The RFID Simulator was developed using MATLAB version 7.7.0.471 (R2008b). MATLAB is an advanced technical computing language that provides an interactive environment in an all-in-one package. It has the ability to solve technical computing issues faster than other programming languages. MATLAB provides many prebuilt functions, tools to create custom graphical user interfaces, graphs, and many more features. In addition to help and support provided by MathWorks, many aid documents published by other sources for MATLAB are available on the Internet. This is a great advantage for users who plan to develop, modify, and extend MATLAB codes. Since MATLAB can be used for signal and image processing, communications, control design, modeling, and various other applications, most colleges, schools, and universities around the world already use MATLAB. Therefore, users of this simulator do not need to purchase MATLAB software specifically to run it.

3.4.3 RFID Simulator Features

The RFID Simulator is a complete, all-in-one package that was designed with the intent of being extended to a commercial RFID simulator. Since the RFID Simulator was developed using MATLAB, users have the ability to run it on any Windows machine. The graphical user interface provides a simple, convenient access to the users. User inputs and results are all placed in one GUI for easy navigation and use. The RFID Simulator also has a command window for users who want to examine what is specifically happening in the code. Results of the simulation are presented as graphs, thus making them clear and easy to read. All parameters, variables, and functions of the code are named in such a way as to allow users to read, understand, and modify it easily. Each and every line of the code is well documented. Only a few RFID simulators that can simulate an RFID system environment are available. And most of them are not free of charge. Therefore, one of the main features of the RFID Simulator developed in this thesis is that it is free. It offers the user an easy and cost efficient opportunity to experiment and design RFID implementations. Since the source code is also provided, users can modify it according to their requirements. Each simulation scenario is run for as many as 1,000 trials, and the average value is used to obtain the most accurate reading. Users have the option to either increase or decrease the number of trials. The progress bar feature provides users with the current status or progress of the simulation. Figure 11 shows a screenshot of the RFID Simulator with the progress bar.

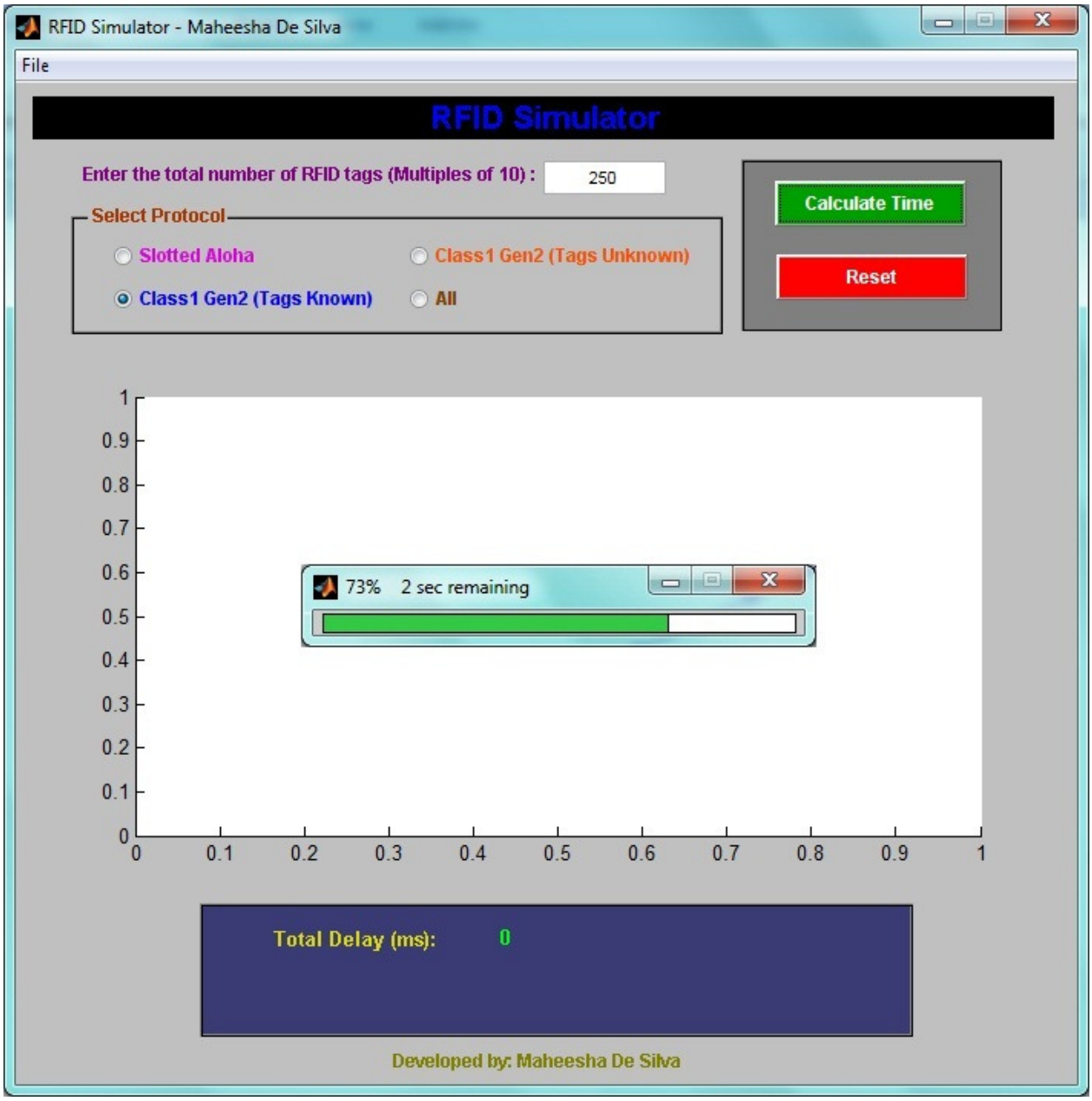


Figure 11: RFID Simulator with Progress Bar

The Reset button clears all inputs and results, and sets the GUI to the initial state. The user of the RFID Simulator also has the benefit of printing the inputs and results directly from it and to operate the simulation using only the keyboard. Another feature of the RFID Simulator is error checking. If the user inputs a non-numeric character for the number of RFID tags, the simulator detects it, displays an error message, and clears the input. Since users have the option

to determine the delay in order to read any number of RFID tags, as a technique to reduce the processing weight on the PC, users are limited to entering only multiples of 10 as the input values for the number of RFID tags. For example, if a user enters the value 100 for the number of RFID tags, the simulation will run for not only 100 tags but also for 0, 10, 20, 30… 80, 90, 100 tags, and on each occasion it will also run for 1,000 trials. If the user inputs a value other than a multiple of 10 for the number of RFID tags, the simulator detects it, displays an error message, and clears the input. Figure 12 shows the error messages displayed when the user inputs an invalid value for the number of RFID tags.

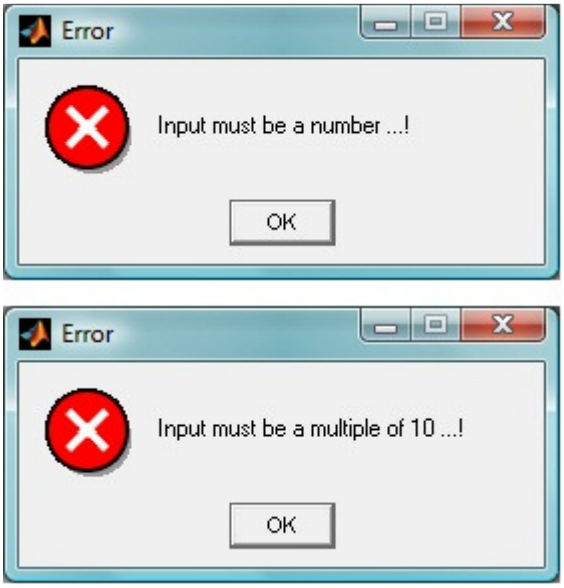


Figure 12: RFID Simulator Error Messages

As a precautionary feature, the RFID Simulator displays a confirmation popup menu when the user clicks on Close in the simulator menu, which is shown in Figure 13.

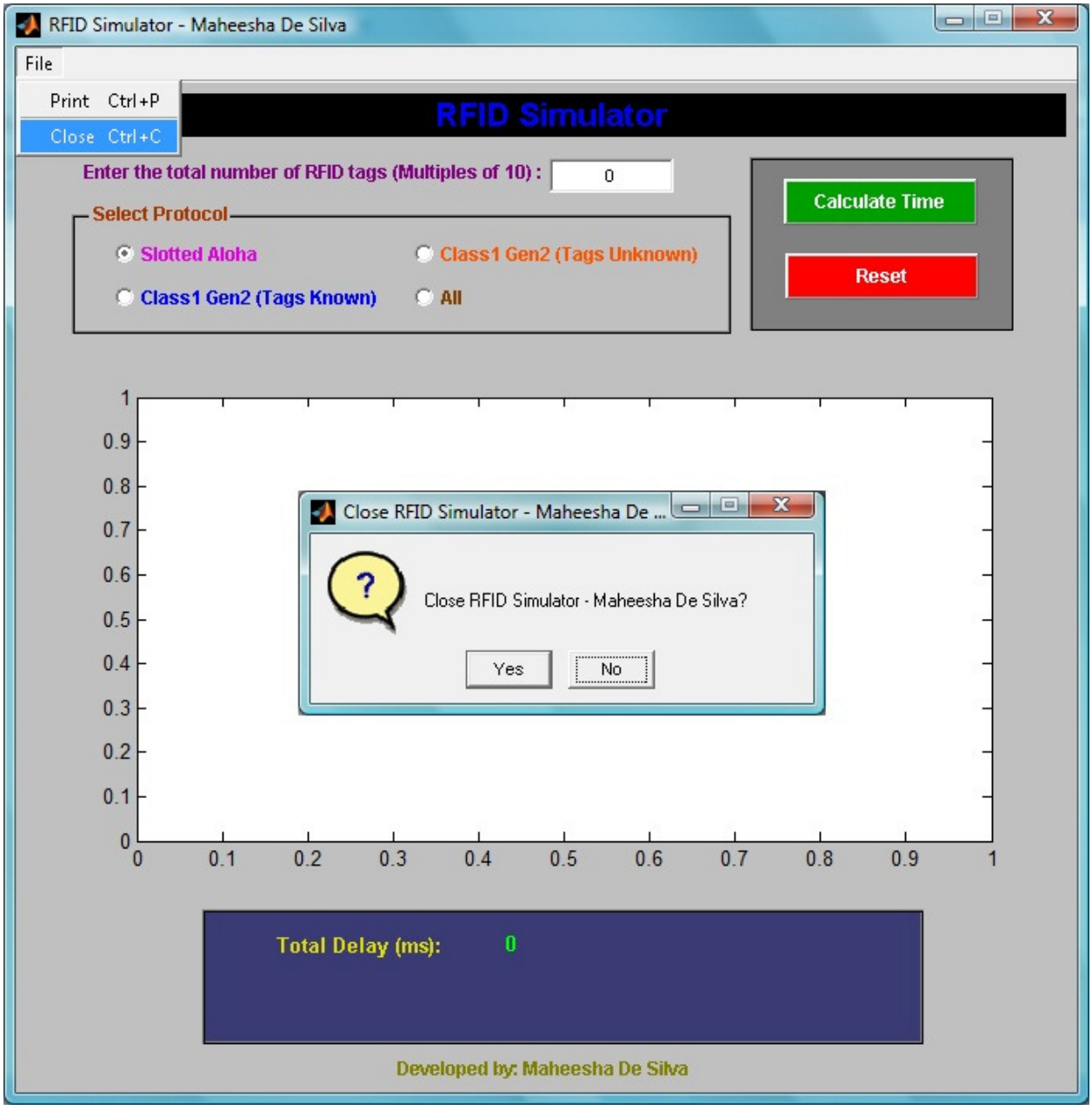
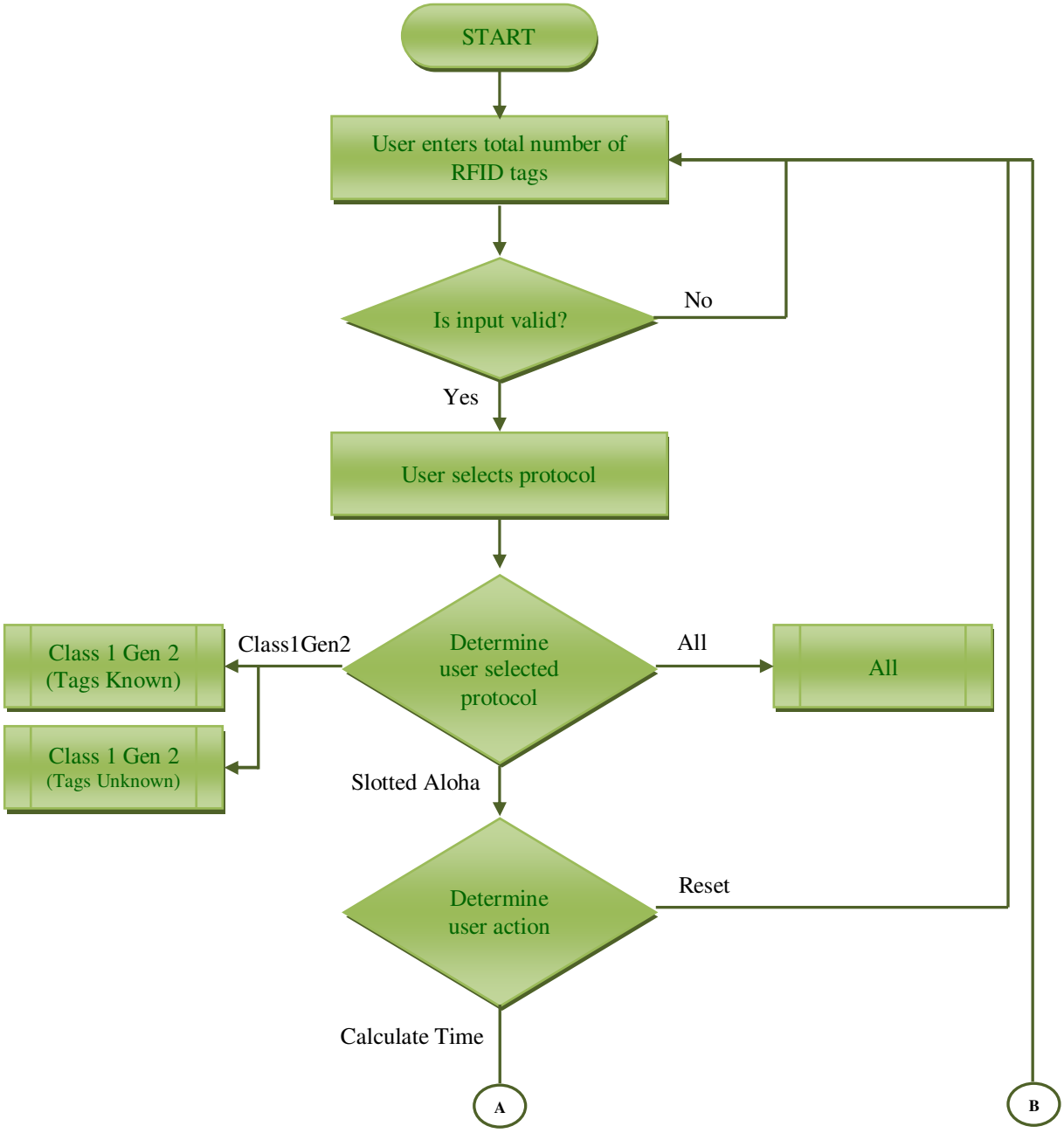
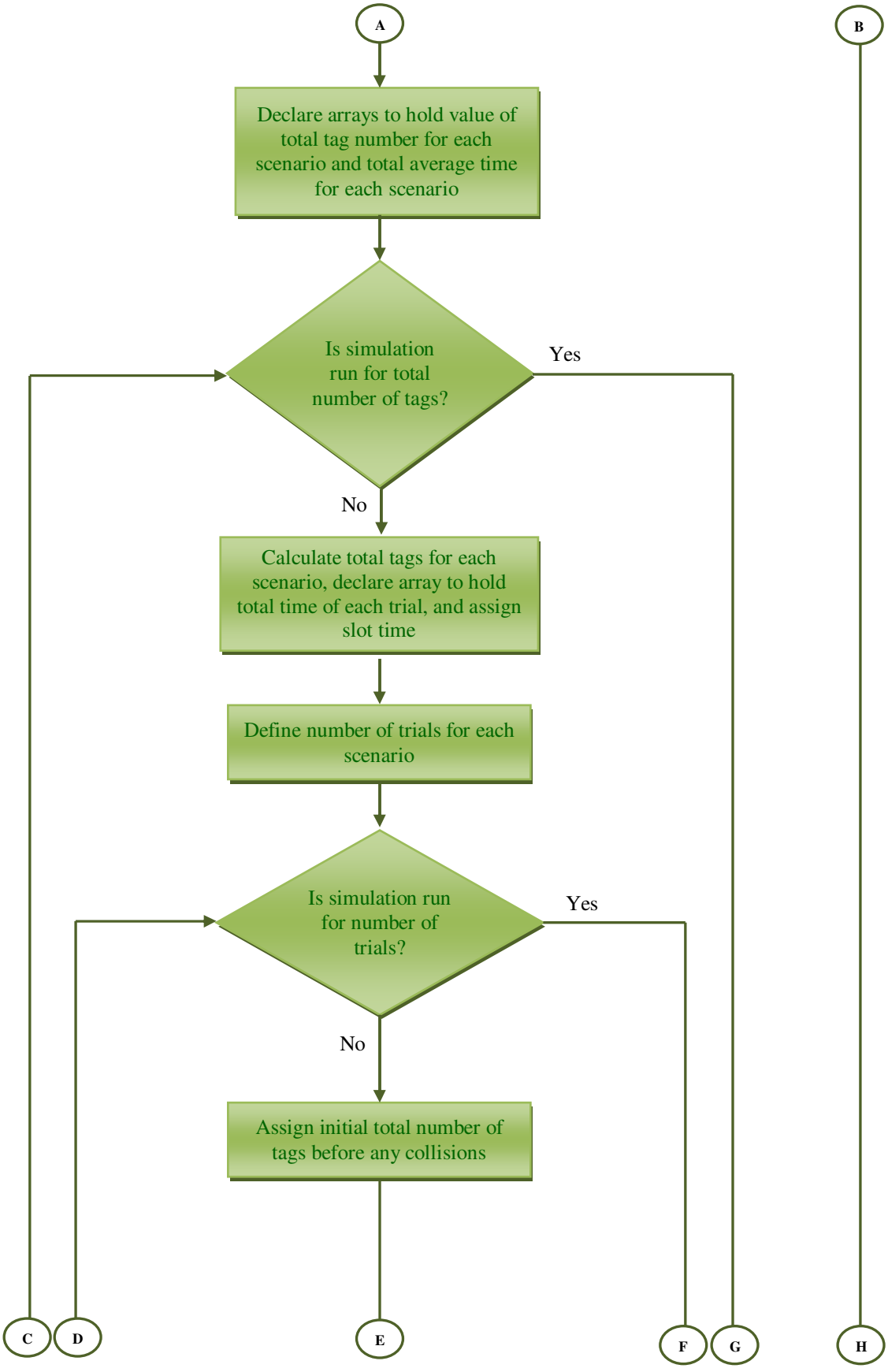


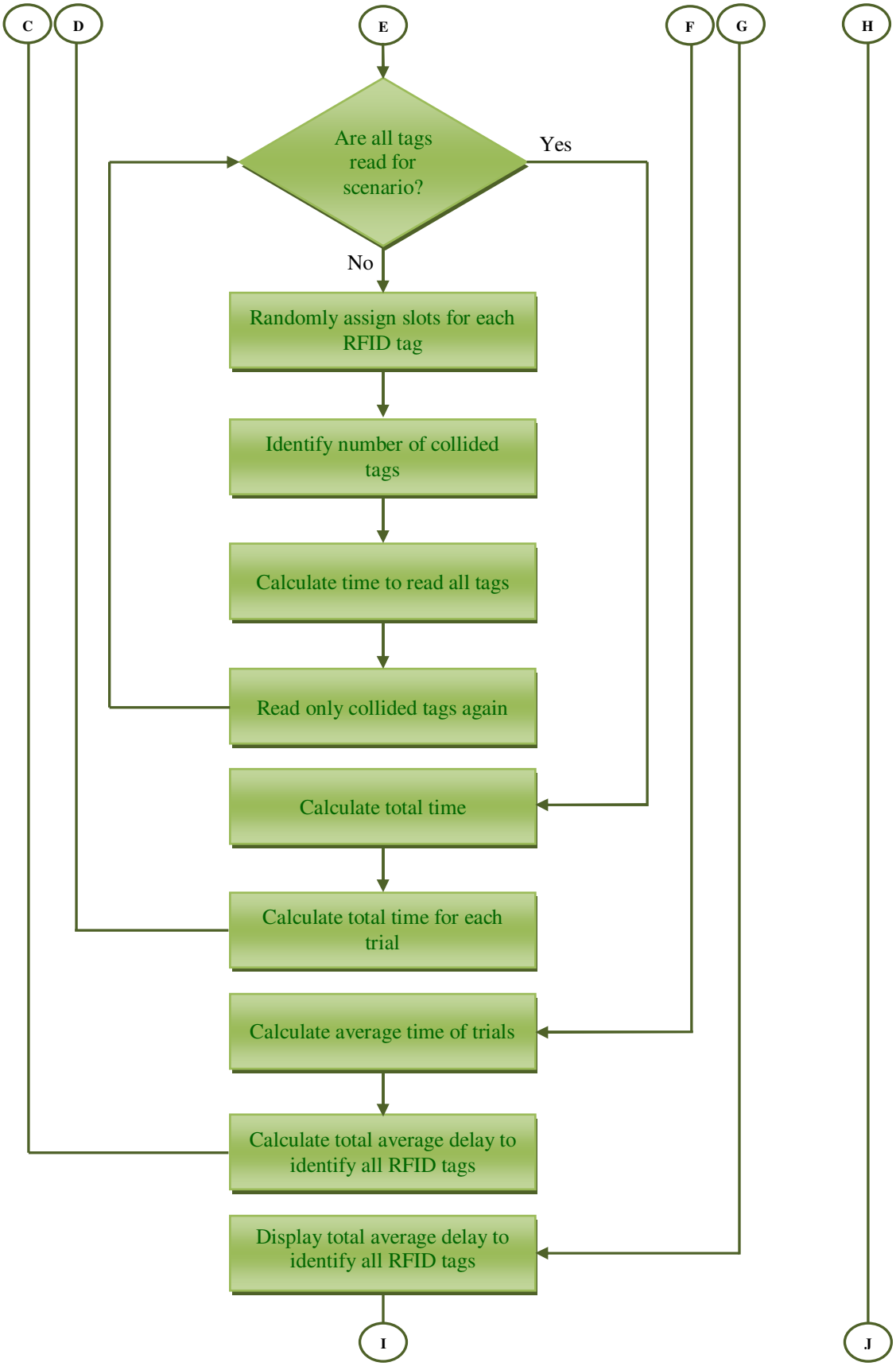
Figure 13: Close RFID Simulator Confirmation Message

3.4.4 Flow Chart

The flow chart for the Slotted Aloha protocol is shown in Figure 14, in order to provide users with a fundamental idea of how the RFID Simulator works for this protocol. This makes it easy for users who plan to modify or extend the functionality of the simulator to understand the code.




A
B
Declare arrays to hold value of total tag number for each scenario and total average time for each scenario
Is simulation run for total number of tags?
Yes
No
Calculate total tags for each scenario, declare array to hold total time of each trial, and assign slot time
Define number of trials for each scenario
Is simulation run for number of trials?
Yes
No
Assign initial total number of tags before any collisions
C
D
E
F
G
H


C
D
E
F
G
H
Are all tags read for scenario?
Yes
No
Randomly assign slots for each RFID tag
Identify number of collided tags
Calculate time to read all tags
Read only collided tags again
Calculate total time
Calculate total time for each trial
Calculate average time of trials
Calculate total average delay to identify all RFID tags
Display total average delay to identify all RFID tags
I
J

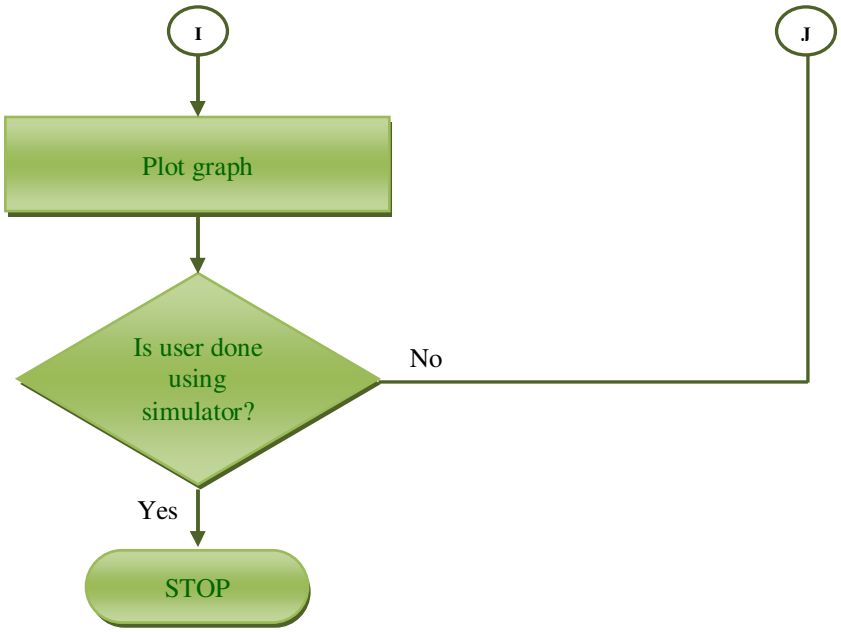


Figure 14: Slotted Aloha Protocol Simulator Flow Chart

3.4.5 Code

The entire source code for the RFID simulator is included in Appendix A. All parameters, variables, and functions of the code are named in such a way as to allow the users to read, understand, and modify the code easily. Each and every line of the code is well documented.

3.4.6 Slotted Aloha Performance Tests and Results

With the Slotted Aloha protocol, the RFID reader knows the number of tags in the field in advance. Therefore, the RFID reader adjusts the number of time slots depending on the number of RFID tags that must be identified. This will make the size of the frame equal to the total number of tags that must be identified. The user enters the total number of RFID tags that must be identified, and the simulation will run until all RFID tags are read successfully.

The delay time to identify RFID tags using the Slotted Aloha protocol was obtained using simulation results from the RFID Simulator. To illustrate how the RFID Simulator works, a sample MATLAB Command Window output for the Slotted Aloha protocol with 25 RFID tags is shown in Figure 15.

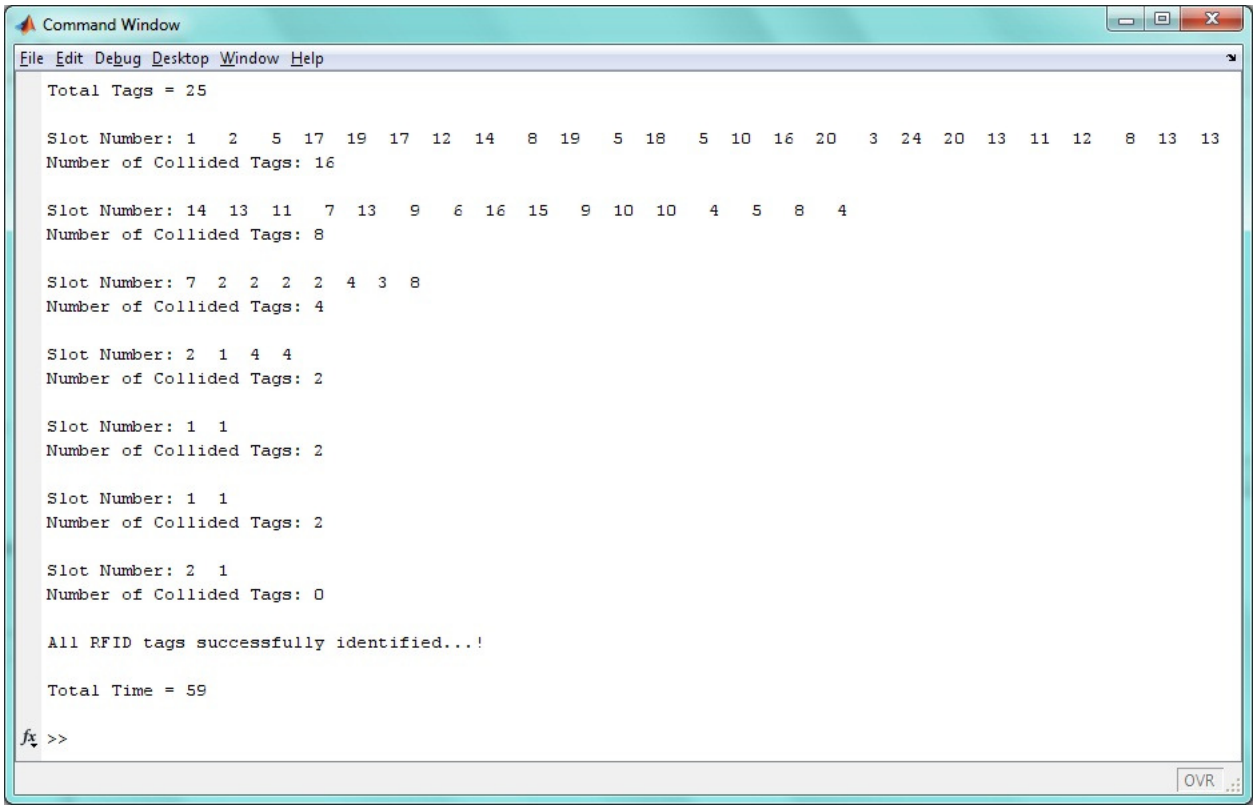


Figure 15: MATLAB Command Window Showing Output of Slotted Aloha Protocol

Tags randomly select a slot to respond to the RFID reader query. If more than one tag responds to the same time slot, there will be a collision at the reader. As a result, these collided tags will not be able to be identified and must be read again. This simulation was run for the following number of tags: 50, 100, 150, 200, 250, 300, 350, 400, 450, and 500. The simulator ran each simulation scenario for 1,000 times or trials, and the average value was taken as a method to obtain the most accurate value. The slot time was taken as 1 ms for the Slotted Aloha protocol simulations. Measured RFID tag read times with the Slotted Aloha protocol are shown in Table 7.

TABLE 7

SIMULATION RESULTS FOR SLOTTED ALOHA PROTOCOL

Total Tags	**Delay (ms)** Slot Time = 1 ms
50	131.622
100	266.072
150	402.269
200	538.775
250	673.147
300	810.550
350	945.227
400	1080.380
450	1218.740
500	1350.300

Figure 16 shows the graph generated by the RFID Simulator and math model for identifying 500 RFID tags using the Slotted Aloha protocol. The total number of RFID tags identified is shown along the x-axis and the total delay or time that it took to identify all RFID tags is shown along the y-axis.

From Figure 16, the equation of the RFID Simulator graph can be written as:

$$y = mx$$

$$m = y/x$$

Therefore, by substituting the values,

$$m = 1350.3/500$$

$$m = 2.7006$$

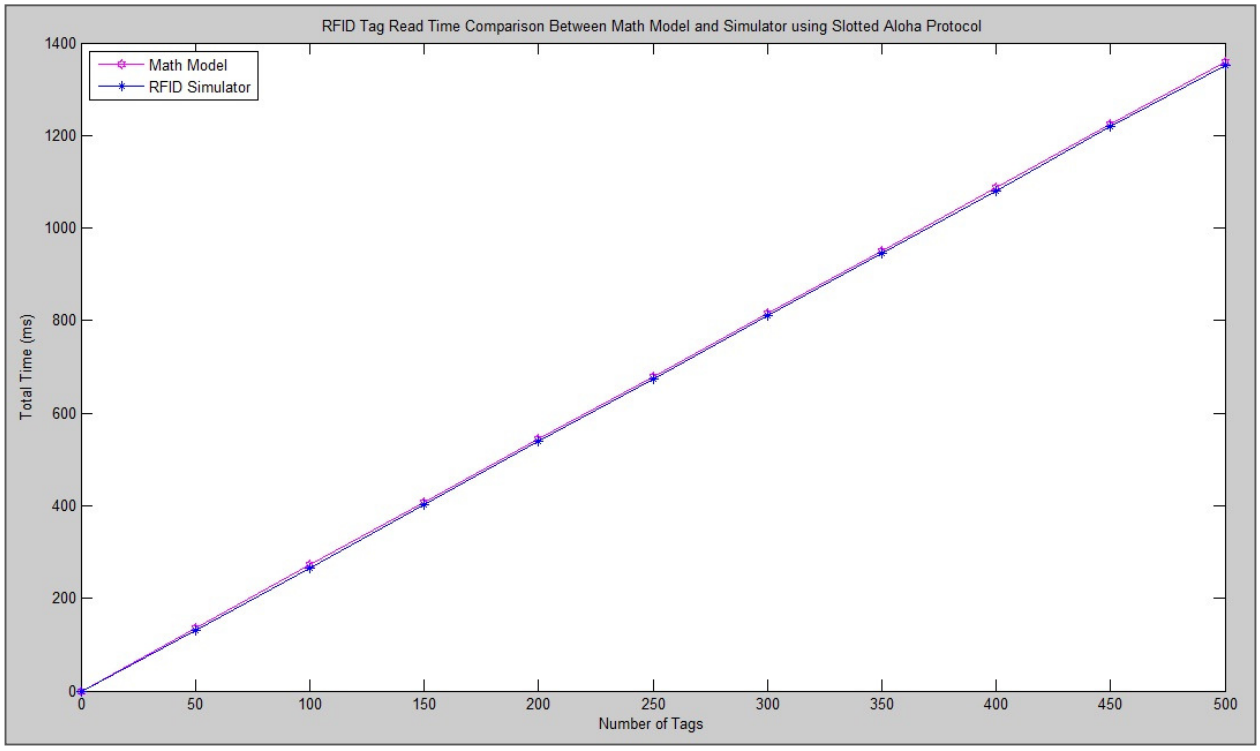


Figure 16: Time to Read 500 RFID Tags Using Slotted Aloha Protocol

The following mathematical model was generated using the simulation results given above to predict the RFID tag identification time for a various number of RFID tags using the Slotted Aloha protocol:

$$y = 2.7006\,x \tag{3.6}$$

The numeric value of e is equal to 2.71828. Therefore, equation (3.6) can be written as

$$y \approx e\,x$$

According to section 3.3, the total number of RFID tags (x-axis) can be denoted as λ, and the total time taken to read the tags (y-axis) can be denoted as $T(\lambda)$. Therefore,

$$x = \lambda$$

$$y = T(\lambda)$$

Hence, the mathematical model from the simulation can be written as:

$$\boldsymbol{T(\lambda)} = \mathrm{e}.\boldsymbol{\lambda} \tag{3.7}$$

The mathematical model shown in equation (3.7) was derived using the RFID Simulator results and is equal to the theoretical result shown in equation (3.5) from section 3.3. This confirms the integrity of the RFID Simulator.

In this chapter, the performance of the Slotted Aloha protocol was evaluated for RFID. In the next chapter, the performance of the EPCglobal Class-1 Generation-2 anti-collision protocol will be evaluated.

CHAPTER 4

PERFORMANCE EVALUATION OF EPCGLOBAL CLASS-1 GENERATION-2 ANTI-COLLISION PROTOCOL FOR RFID SYSTEMS

4.1 EPCglobal Class-1 Generation-2 Protocol

Another variation of the Aloha anti-collision protocol is the EPCglobal Class-1 Generation-2 protocol, also known as EPC Gen 2, Class 1 Gen 2, ISO 18000-6C, or Q algorithm. UHF tags are the least expensive and most widely used tags in RFID systems. As a result, the EPC Gen 2 protocol was developed to increase the performance and reliability of UHF tags.

The Hardware Action Group (HAG) of EPCglobal developed EPC Gen 2 to overcome the problems of Class 0 and Class 1 Gen 1 protocols. The manufacturer programmed the Class 0 tag, and the field programmable Class 1 Gen 1 tag specifications were not interoperable, even though they were open standards. Class 1 Gen 2 was able to retain high performance in order to control memory access and offer interoperability, and was designed to meet the regulatory standards [4].

4.1.1 Advantages of Class 1 Gen 2

Class 1 Gen 2 has many advantages compared to previous protocols. One of its main advantages is the establishment of a single UHF specification, which solved the issue of incompatible standards. As a result, a single EPC Gen 2 reader is able to identify all compliant tags. Class 1 Gen 2 was designed to operate in the entire UHF range of 860–960 MHz to avoid contradictory frequency applications around the globe. Therefore, products that are manufactured according to the EPC Gen 2 standard work in any region of the world. This protocol utilizes frequency and power to meet the terms of the main regional regulations. EPC Gen 2 provides greater security by including password protection, password authorization, and complex

encryption. This new protocol also provides better tag read and write speeds and the capability to work in dense reader surroundings [2].

4.1.2 New Features of Class 1 Gen 2

The Class 1 Gen 2 protocol consists of many new features: faster operating speeds, powerful tag identifying techniques, dense reader operating modes, reader sessions, improved security and privacy, and ability to extend to higher-function systems. These are explained in more detail below [2].

4.1.2.1 Faster Operating Speeds

The Class 1 Gen 2 protocol provides faster and very flexible operating speeds. It has four communication speeds. The Class 1 Gen 2 system has the ability to read more than 1,600 tags per second at maximum speed. The reader can switch to other reading speeds if there is a great deal of interference in the environment. The readers can slow down the reading speed to 100 tags per second to attain precise and full reads in a noisy environment. Class 1 Gen 2 also has the ability to set aggressive targets for the fast programming of tags. It can write at least five tags per second and can set a target of 30 tags per second. This provides the capability to use Class 1 Gen 2 for high-speed applications [2].

4.1.2.2 Powerful Tag Identifying Techniques

RFID tags situated at the outer border of the reader region are hard to read reliably. These tags obtain only a small, uneven power from the reader. Class 1 Gen 2 provides new features to overcome this issue.

One major feature is the Q algorithm. The Q algorithm makes use of short, straightforward query and acknowledgment commands between the readers and tags, so a tag

which is receiving only small, uneven power can still be identified [2]. The Q algorithm is explained in detail in section 4.1.3.2.1.

Class 1 Gen 2 also provides a sleep mode once the tags have been identified. As a result, readers can focus more on the tags that are hard to read in the read area. When the reader performs a new count, it sends a wake-up or activation command to all tags, whereupon the tags wake-up and are ready to be identified [2].

Class 1 Gen 2 also introduced a dual-state symmetric inventory method. With this method, the reader can start a new tag count without tags taking the time to respond to the reader's wake-up command. The tag alters its state each time it is identified. Tags move from "A" state to "B" state automatically once they are counted by the reader. As the reader counts each "B" state tag, it then moves back to the "A" state tags. This procedure stops when all tags in the read region have been identified [2].

4.1.2.3 Dense Reader Operating Modes

The dense reader operating modes of Class 1 Gen 2 alleviate reader interference. When readers operate in close range of each other, their signals may interfere in numerous ways. A nearby reader may unintentionally alter the state of a tag in another reader's read range or may be communicating on the same frequency as tags in another reader area, which may destroy the fairly weak tag signals. Also, nearby reader's signals may stimulate a tag to transmit in the neighboring reader's range. The signals of this tag may collide with the signals of another tag that is already communicating with a reader in that zone. As a result, the reader is not able to identify any of the tags. Older protocols used conventional methods to permit dense readers to operate efficiently. Since there is a broad bandwidth in the United States, the Federal Communications Commission mandated frequency hopping as the solution. In other regions

where available bandwidth is narrower, readers use the technique listen-before-talk to avoid collisions; however, this technique is much slower [2].

Class 1 Gen 2 has many features to improve dense reader operations. It utilizes the available bandwidth very efficiently by using the Q algorithm, which identifies tags more effectively. Class 1 Gen 2 also provides different operation modes for dense reader situations. If the number of simultaneously active readers is equal to the number of channels, the reader automatically changes itself to the dense reader mode. This feature is essential to prevent readers from interfering with each other. Reader communications are performed in channels separate from tag communications in the dense reader mode to avoid readers from superseding tag communications, and tag replies are isolated into side channels to make them well heard. These are known as *Miller subcarrier* rates. Depending on the overall environmental interference circumstances, the reader has the ability to specify side channels with different widths. It is easier for the reader to identify the tag when the side channels are narrower. Class 1 Gen 2 also offers an FM signaling mode. FM signaling mode has quicker reads than the Miller subcarriers, but it is more susceptible to noise [2].

4.1.2.4 Reader Sessions

Another feature of the Class 1 Gen 2 protocol is that a tag can be read simultaneously using more than one reader. This is done by using reader sessions known as S0, S1, S2, and S3 [15]. The four logical sessions are assigned to readers for communication, and up to four separate readers can identify the same tag concurrently by using its own session with the tag without any interference and without waiting for any one reader to finish its identification. The RFID middleware or the reader itself can administer the monitoring and allocation of sessions [2].

4.1.2.5 Enhanced Security and Privacy

Class 1 Gen 2 allows the user to configure 32-bit passwords. The kill command that shuts down a tag permanently can be guarded using passwords. Passwords are also used to protect the tag's memory. To prevent an eavesdropper from identifying the EPC number of a tag, the Q algorithm never transmits the entire ID of a tag on air at once as an enhanced security feature. Class 1 Gen 2 also uses optional handle-based commands to safely control RFID tags. Each tag generates a random 16-bit number on its handle when it is inventoried by the reader. The reader then uses this handle to improve the safety of write, read, and erase commands [2].

4.1.3 Tag Identification

The RFID reader sends commands to RFID tags in its read zone to gather information from the tags. All commands are sent only by the reader to the tags. The reader has the ability to choose a subset of tags from the reader range to communicate, list the chosen tags, and read/write data from every tag using these commands. A reader manages the tags in its range, also known as the tag population, using the three basic operations: Select, Inventory, and Access. Figure 17 shows the reader/tag operations and the different states of the tag.

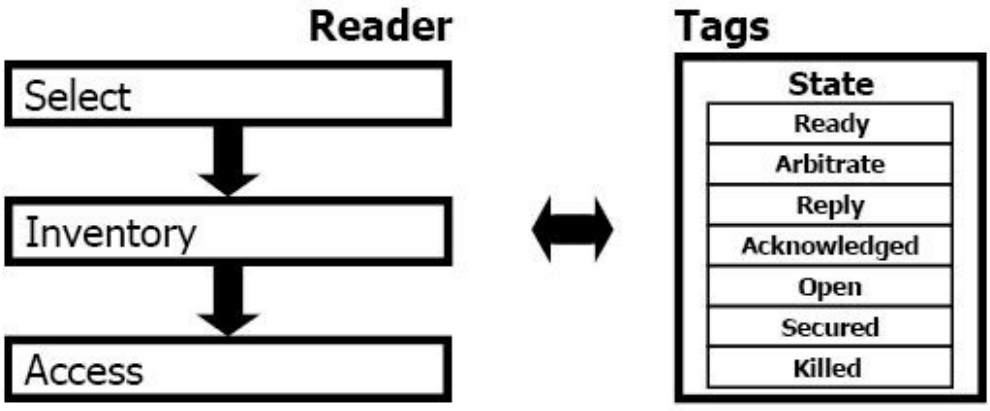


Figure 17: Reader/Tag Operations and Tag State [29]

4.1.3.1 Select Command

In certain circumstances, users do not need to obtain data from all tags in the reader range. In these situations, the user has the ability to pick a subset of the tag population by using the Select command. The Select command allows the reader to inventory and access the selected tags. The Select command indicates the data mask with the memory location, and the RFID tag compares it with the data at the indicated location on the tag. If it matches, the tag is selected to participate in the inventory [4]. Still, the reader does not know how many tags will be participating in the inventory. Only the tags know whether they need to participate in the inventory or not.

4.1.3.2 Inventory Commands

Inventory is the process of identifying tags. The reader uses inventory commands to identify tags in the selected population once it has selected a subset from the tags. The RFID reader initiates an inventory by broadcasting a query command to the selected tags in one of the four sessions. Tags reply with a 16-bit random number, and the reader identifies a single tag reply. Then the reader asks for the PC, EPC, and 16-bit cyclic redundancy check (CRC) from the tag. The inventory process includes the commands Query, QueryAdjust, QueryRep, ACK, and NAK [4]. The inventory process is explained in detail in section 4.1.3.2.1 under the Q algorithm. Figure 18 shows the tag identification process and timing relationship in the Class 1 Gen 2 protocol.

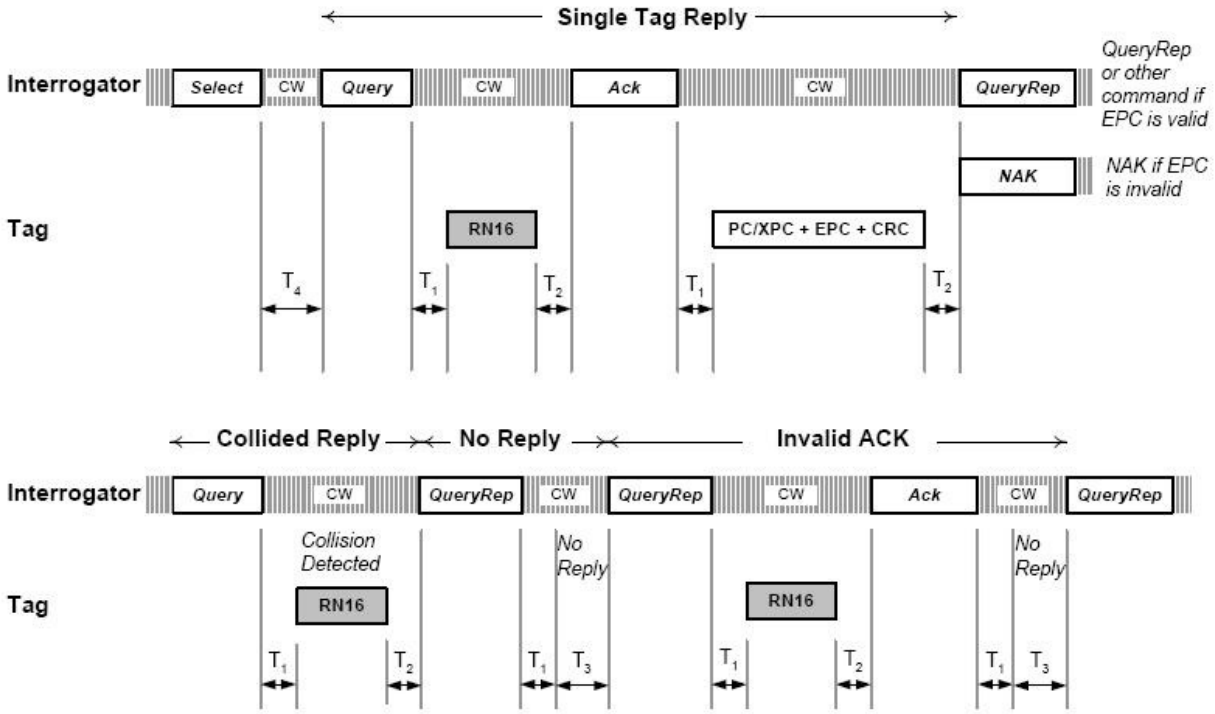


Figure 18: Tag Identification Process and Timing Relationship in Class 1 Gen 2 Protocol [29]

Notes: T_1: Time from reader transmission to tag response.
T_2: Time from tag response to reader transmission.
T_3: Time a reader waits, after T_1, before it issues another command.
T_4: Minimum time between reader commands.

4.1.3.2.1 Q Algorithm

The Q algorithm is based on the random Slotted Aloha algorithm. The fundamental principle of the Q algorithm is to assign a reading slot number dynamically by exchanging Q values between the reader and tags.

The reader begins an inventory by transmitting a Query command to the tags with the session number and the Q parameter. The initial value of Q is 4. The Q parameter value is limited to integers between 0 and 15. Tags that are participating in the identification process select a random value between 0 and 2^Q-1 and store this value in the slot counter. The random number assists in reducing the number of tag collisions. Tags that randomly select the value 0 for the slot counter move to the Reply state and reply immediately with a 16-bit random number

(RN16) to the reader; the remaining tags go to an Arbitrate state. The random number gives a basic cryptography key and is also utilized as a handle to recognize the tag. If there was a collided reply or no replies from the tags, the reader uses the QueryAdjust command to modify the Q value used in the initial Query command. It informs the tags to either increase or decrease the current Q value. If only one tag replied, the current Q value will not be changed and the reader uses the QueryRep command to inform all tags to decrement their current slot counter by 1. When the counter reaches 0, the tag starts the communication. If more than one tag responds simultaneously, there will be collisions at the reader. Therefore, the reader will not acknowledge these tags, and the slot counter value 0 of the tags that responded will be decremented to 32767 or 2^{15}-1 or 7FFFh. If the tag response was successful, the reader acknowledges the tag by transmitting the ACK command to the tag after it receives the 16-bit random number. The tag then replies to the reader's ACK command by sending its PC, EPC, and CRC value. If the reader needs to reset a tag to the Arbitrate state, it sends the NAK command to the tag [15][30][4].

The Q value is adjusted by the reader according to no reply, collided reply, and successful reply. Q_{fp} indicates the floating-point depiction of Q. The reader rounds the Q_{fp} value to the nearest integer and assigns this value for Q in the Query command [15][30].

$$\boldsymbol{Q = round\ (Q_{fp})} \tag{4.1}$$

If only one tag has selected the value 0 for the slot counter, then the reader will receive a successful reply, and the current value of Q and Q_{fp} will remain unchanged. But if there was a collided reply, then more than one tag has selected value 0 for the slot counter. In this case, the value of Q is considered too small or there are several remaining unidentified tags. Therefore, Q_{fp} is incremented by the value of c, which is a real number. But still the maximum value of Q_{fp} is 15. The reader will not receive any replies if no tags have selected the value of 0 for the slot

counter. This means that the value of Q is too large or there are a small number of remaining unidentified tags. Therefore, Q_{fp} is decremented by the same value of c. The value zero will be assigned to Q_{fp} if it receives a negative value [15][30].

The values for c are $0.1 < c < 0.5$. The reader will use small values of c when Q is large and large values of c when Q is small. The Q value will be incremented by 1 or decremented by 1 or remains unchanged after each Q update since $c < 1$. The new value of Q is sent to the tags by the reader using the QueryAdjust command [15][30].

There are also several disadvantages of using the Q algorithm. Many commands need to be exchanged between the reader and RFID tags since the value of Q is updated after every reading slot. This process adds considerable overhead and increases power consumption. Since the Q algorithm dynamically changes the reading slot number, it requires some time to converge to the best value, and this also decreases the reading efficiency [31]. Figure 19 shows the flow chart of the Class 1 Gen 2 Q algorithm.

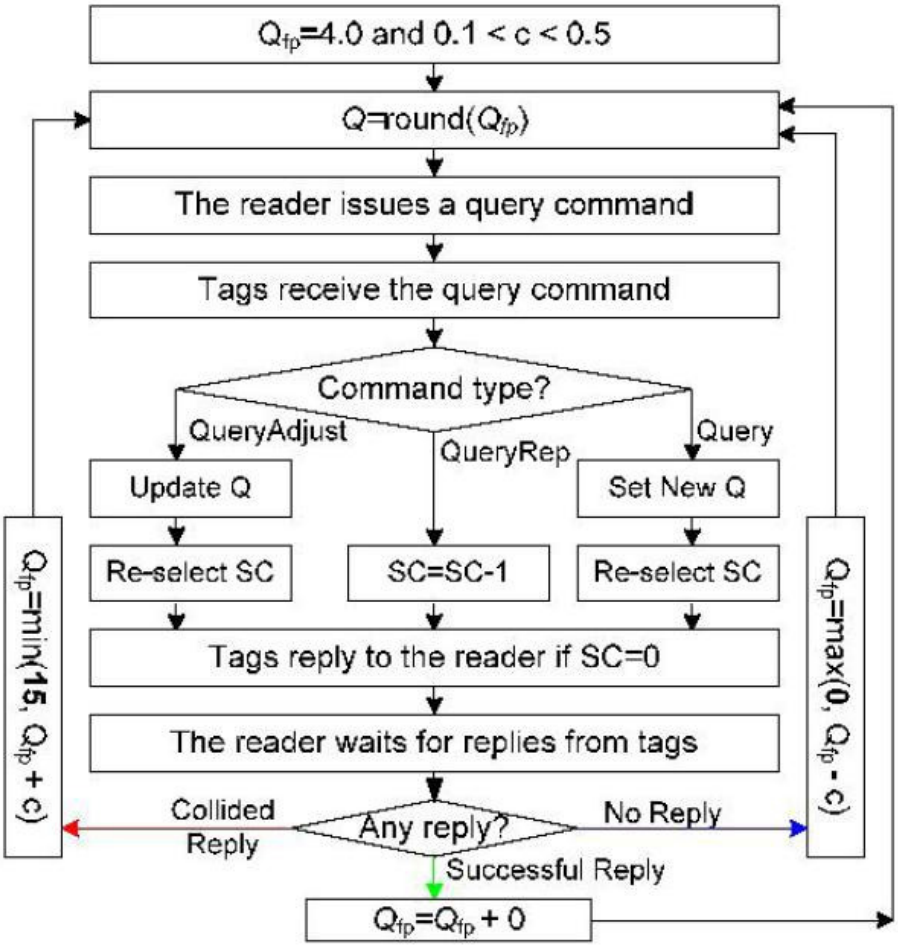


Figure 19: Flow Chart of Class 1 Gen 2 Q Algorithm [30]

4.1.3.3 Access Commands

Access is the process of communicating with RFID tags, for example, reading data from a tag and writing data to a tag. Once a tag is uniquely identified by the reader, it uses access commands to access the user memory bank of the tag. The access commands comprise Req_RN, Read, Write, Kill, Lock, Access, BlockWrite, and BlockErase [4]. Figure 20 shows the tag inventory and access of a single tag by the RFID reader.

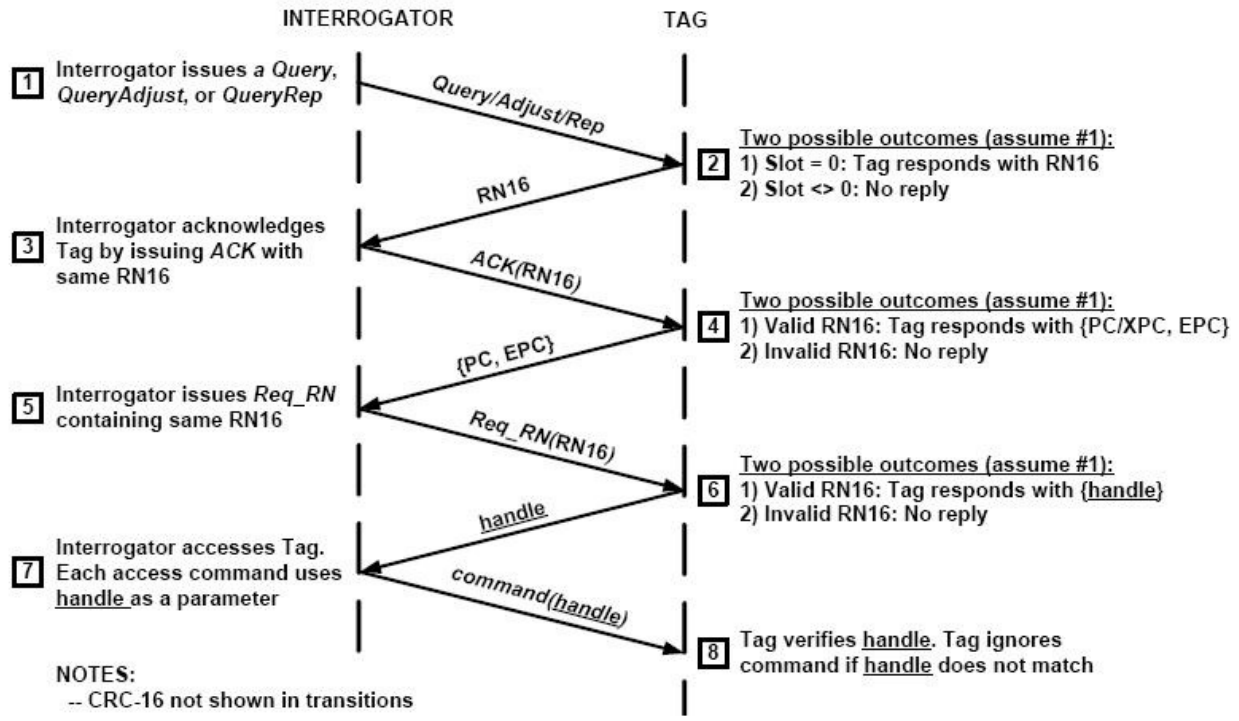


Figure 20: Tag Inventory and Access of a Single Tag [29]

4.1.3.4 Tag States

RFID tags have seven tag states in the Class 1 Gen 2 protocol: Ready, Arbitrate, Reply, Acknowledged, Open, Secured, and Killed. When the tag is not included in the current inventory, it is in the Ready state. The tag is in the Arbitrate state when it is participating in an inventory, but the slot counter of the tag has not reached zero yet. The tag generates the 16-bit random number and transmits it to the RFID reader, when it is in the Reply state. The tag goes to the Acknowledged state, if it receives an ACK from the reader. If not, the tag goes to the Arbitrate state. If the tag is in the Acknowledged state, then it can go to any state apart from the Killed state. An RFID tag that does not have a password of zero goes to the Open state from the Acknowledged state and gets a Req_RN from the RFID reader. A tag that is in the Open state can move to any state apart from Acknowledged state. An RFID tag with a zero password goes to the Secured state from the Acknowledged state after it gets a Req_RN from the reader. The tag

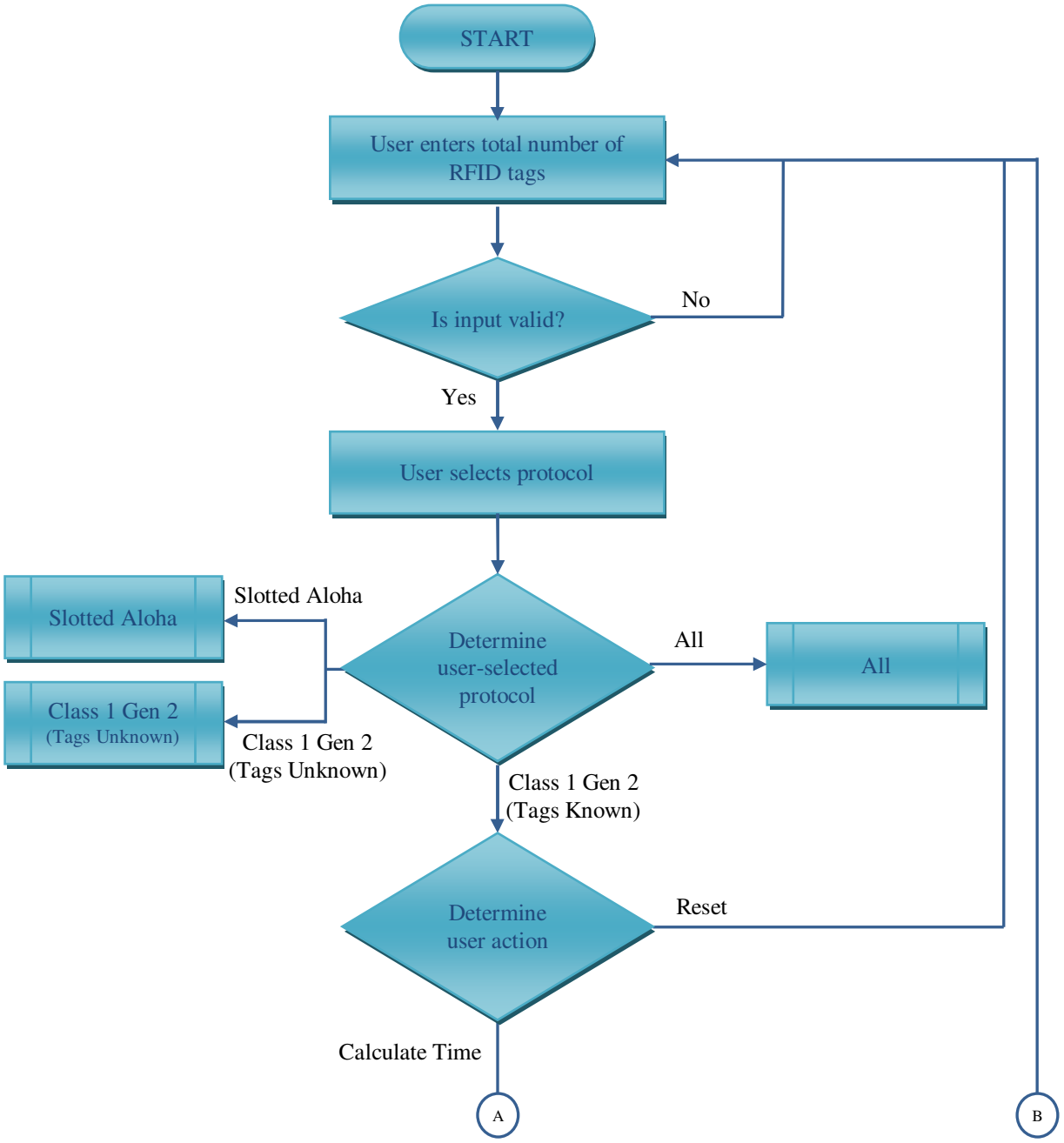
will also move to the Secured state from the Open state if it does not have a password of zero and when it gets an Access command. A tag that is in the Secured state can move to any state apart from the Open or Acknowledged states. The tag transmits a success reply to the RFID reader when it moves to the Killed state. Then it is disabled forever and never replies to any requests from the reader [1].

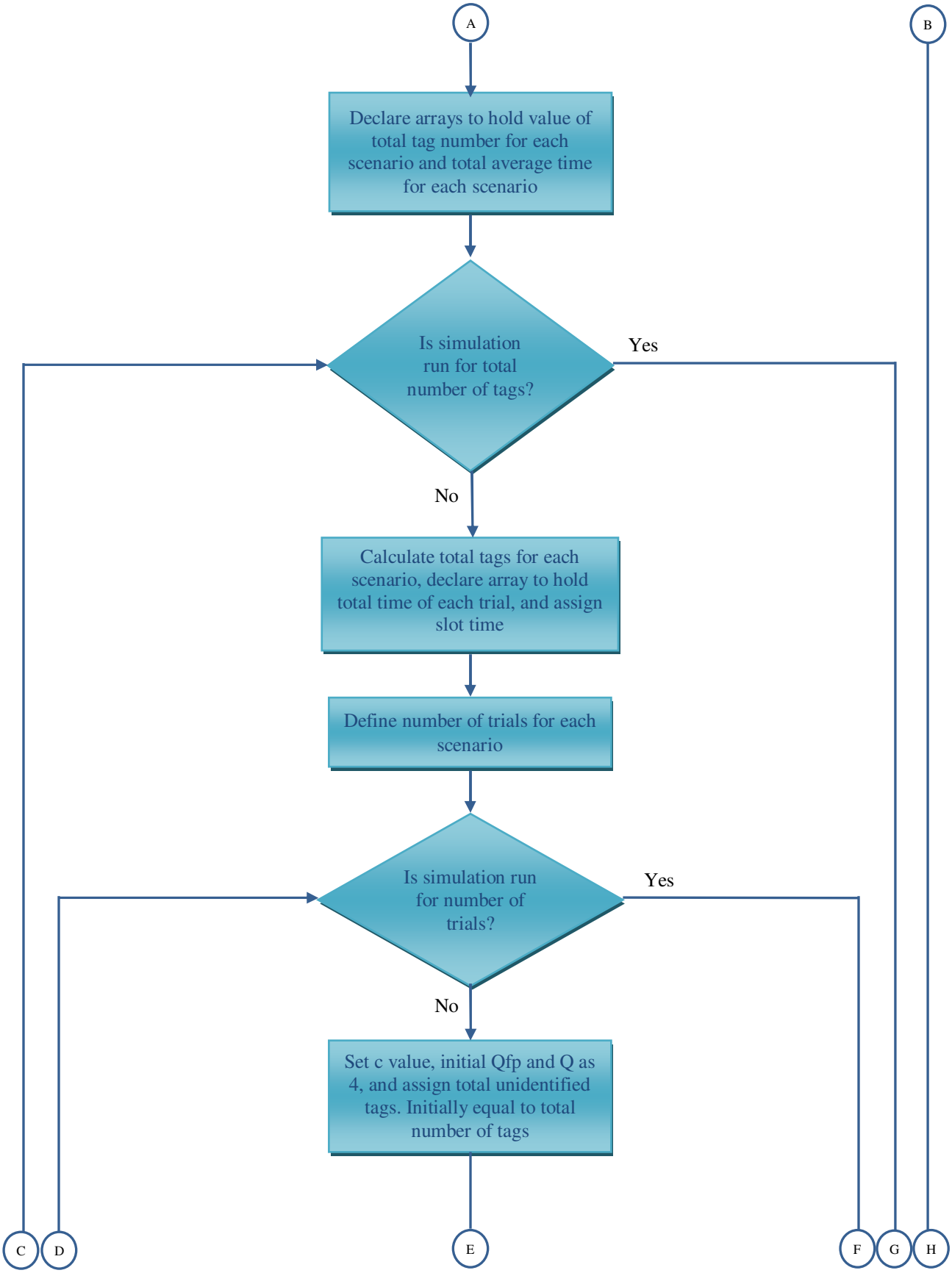
4.2 Simulation

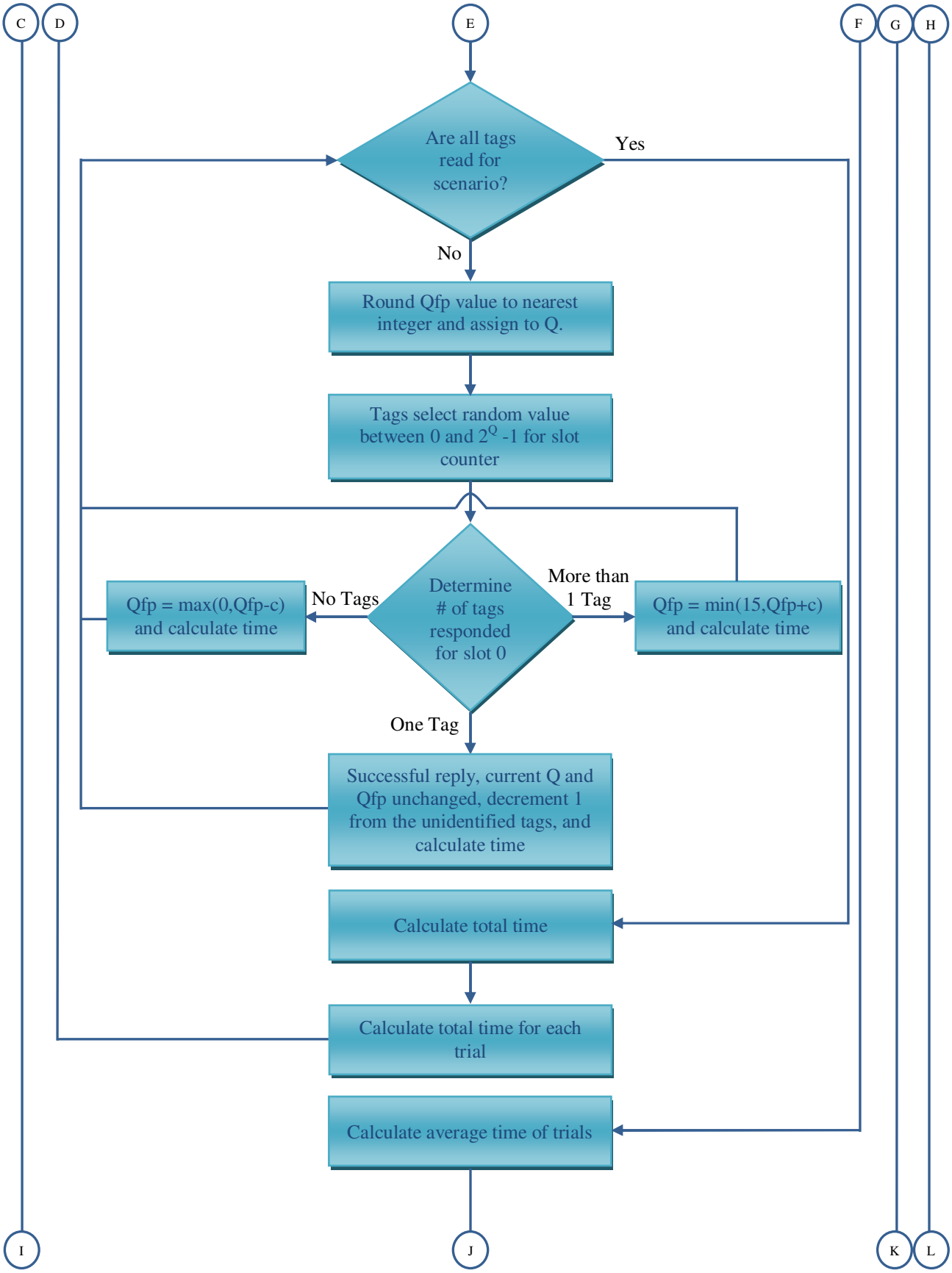
In this section, the performance of the EPCglobal Class-1 Generation-2 protocol for RFID was evaluated using the RFID Simulator. The RFID Simulator provides two options for the EPCglobal Class-1 Generation-2 protocol: the first option gives the time to identify all tags when the RFID reader knows beforehand the total number of tags in the range, and the second option gives the time to identify all tags when the RFID reader does not know beforehand the total number of tags in the range. The flow charts and the source code are demonstrated in the following sections. Then the Class 1 Gen 2 performance tests and results are explained in detail followed by the simulation results comparison between the Slotted Aloha and Class 1 Gen 2 protocols.

4.2.1 Flow Chart

The flow chart for the Class 1 Gen 2 protocol when the RFID reader knows the total number of tags in the range in advance is shown in Figure 21. The flow chart provides a fundamental idea of on how the RFID Simulator works for Class 1 Gen 2. This makes it easier to understand the code for those users who plan to modify or extend the functionality of the simulator.




A
B
Declare arrays to hold value of total tag number for each scenario and total average time for each scenario
Is simulation run for total number of tags?
Yes
No
Calculate total tags for each scenario, declare array to hold total time of each trial, and assign slot time
Define number of trials for each scenario
Is simulation run for number of trials?
Yes
No
Set c value, initial Qfp and Q as 4, and assign total unidentified tags. Initially equal to total number of tags
C
D
E
F
G
H


C
D
E
F
G
H
Are all tags read for scenario?
Yes
No
Round Qfp value to nearest integer and assign to Q.
Tags select random value between 0 and 2^Q -1 for slot counter
Determine # of tags responded for slot 0
No Tags
More than 1 Tag
Qfp = max(0,Qfp-c) and calculate time
Qfp = min(15,Qfp+c) and calculate time
One Tag
Successful reply, current Q and Qfp unchanged, decrement 1 from the unidentified tags, and calculate time
Calculate total time
Calculate total time for each trial
Calculate average time of trials
I
J
K
L

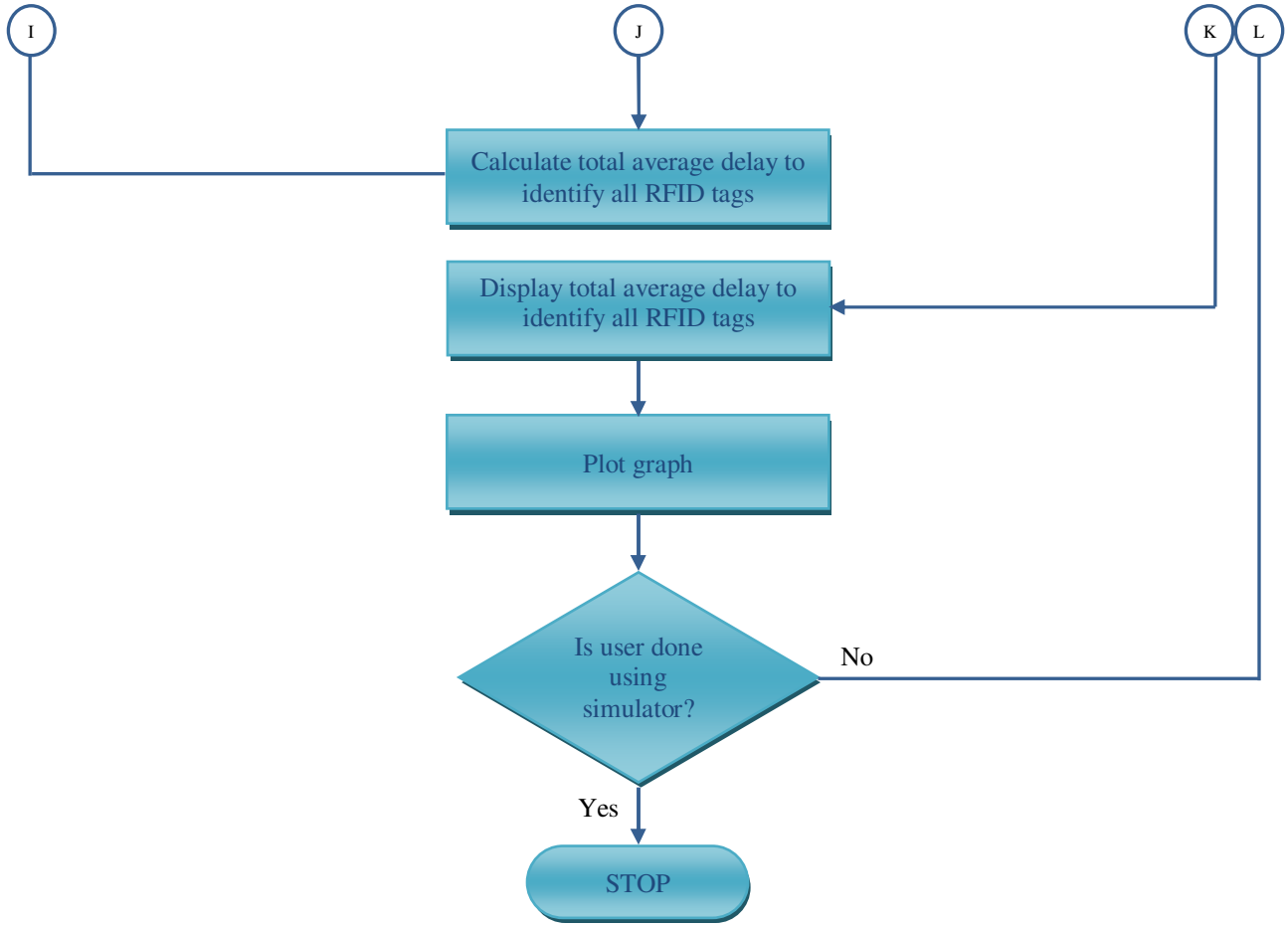
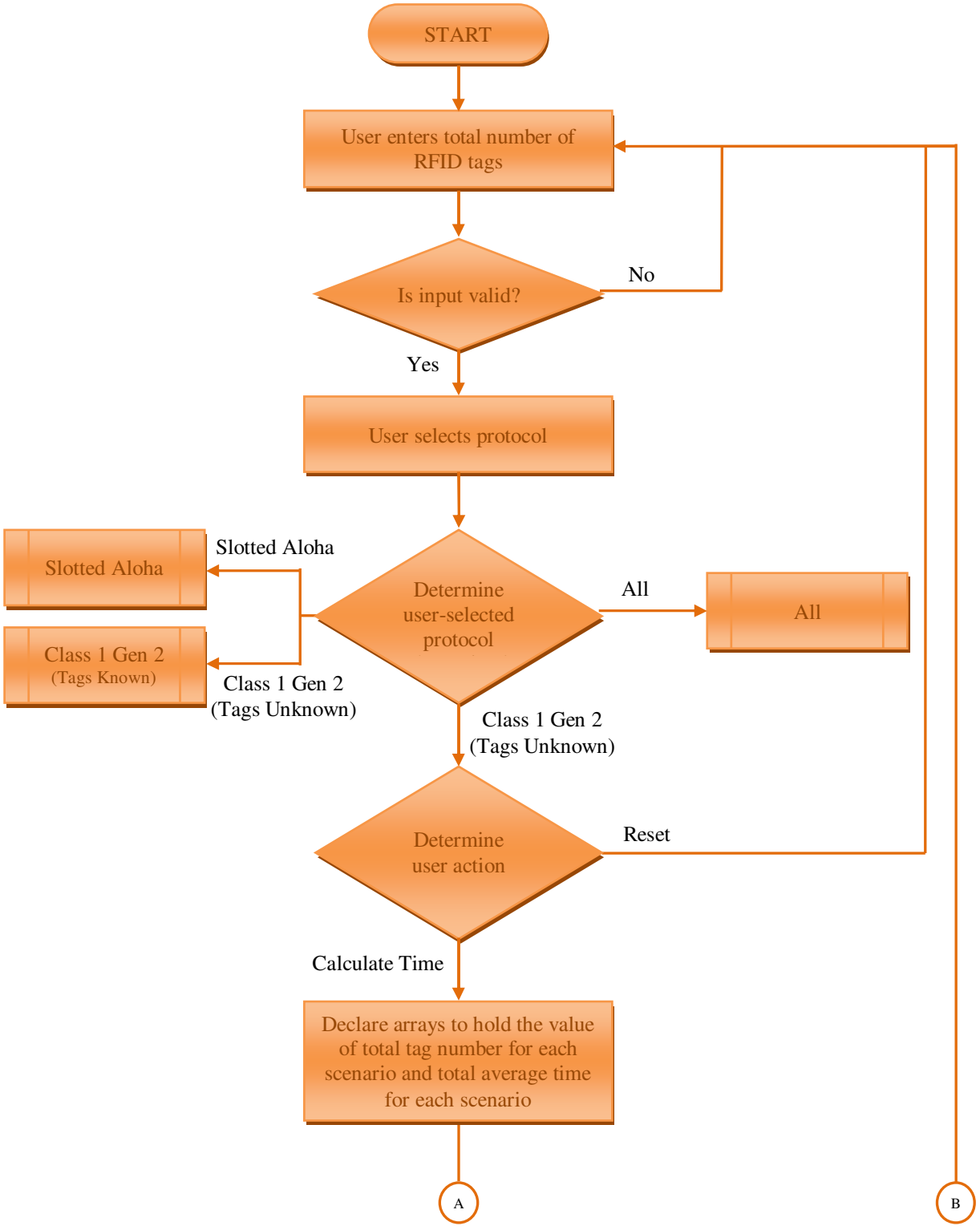
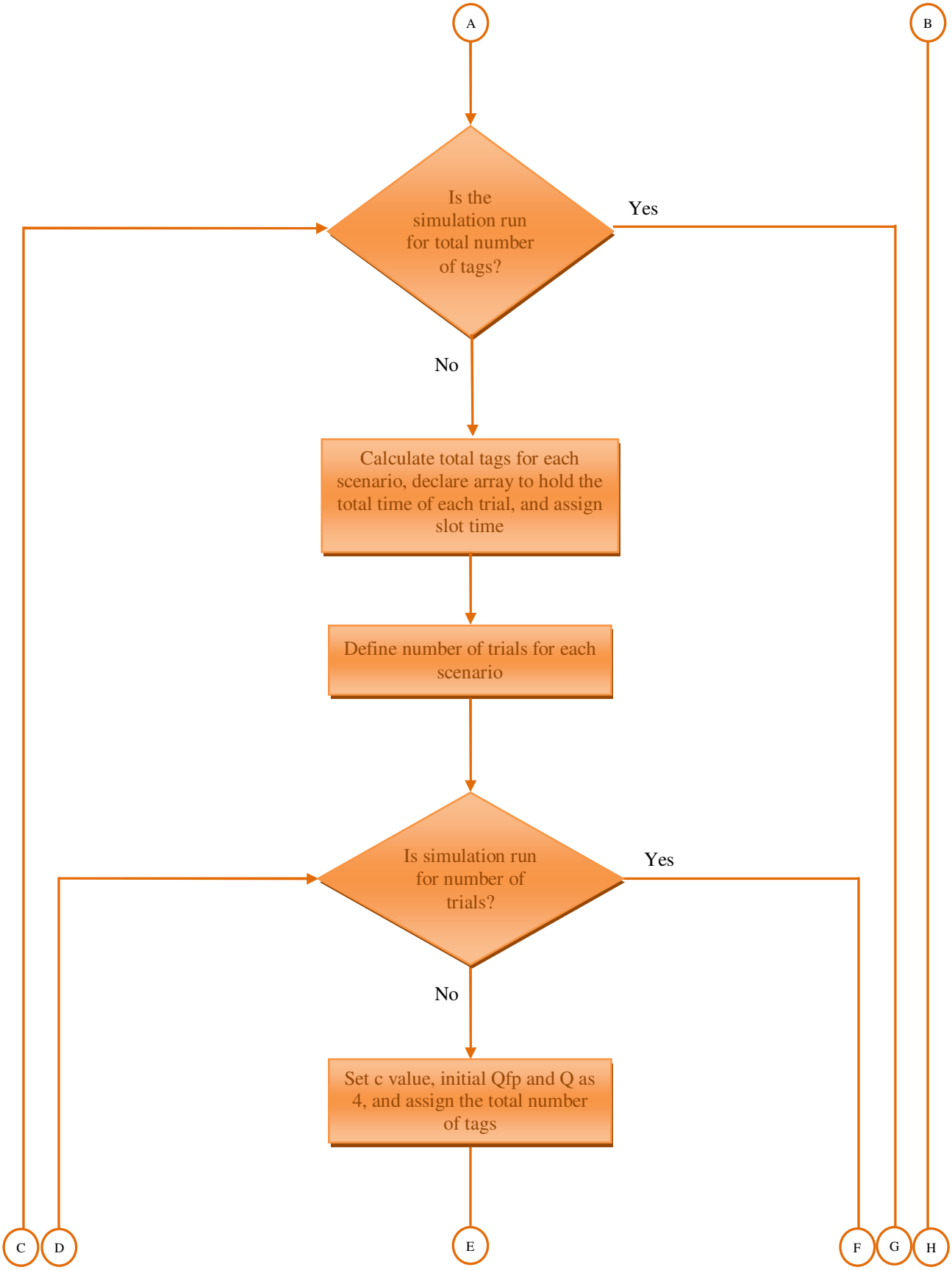
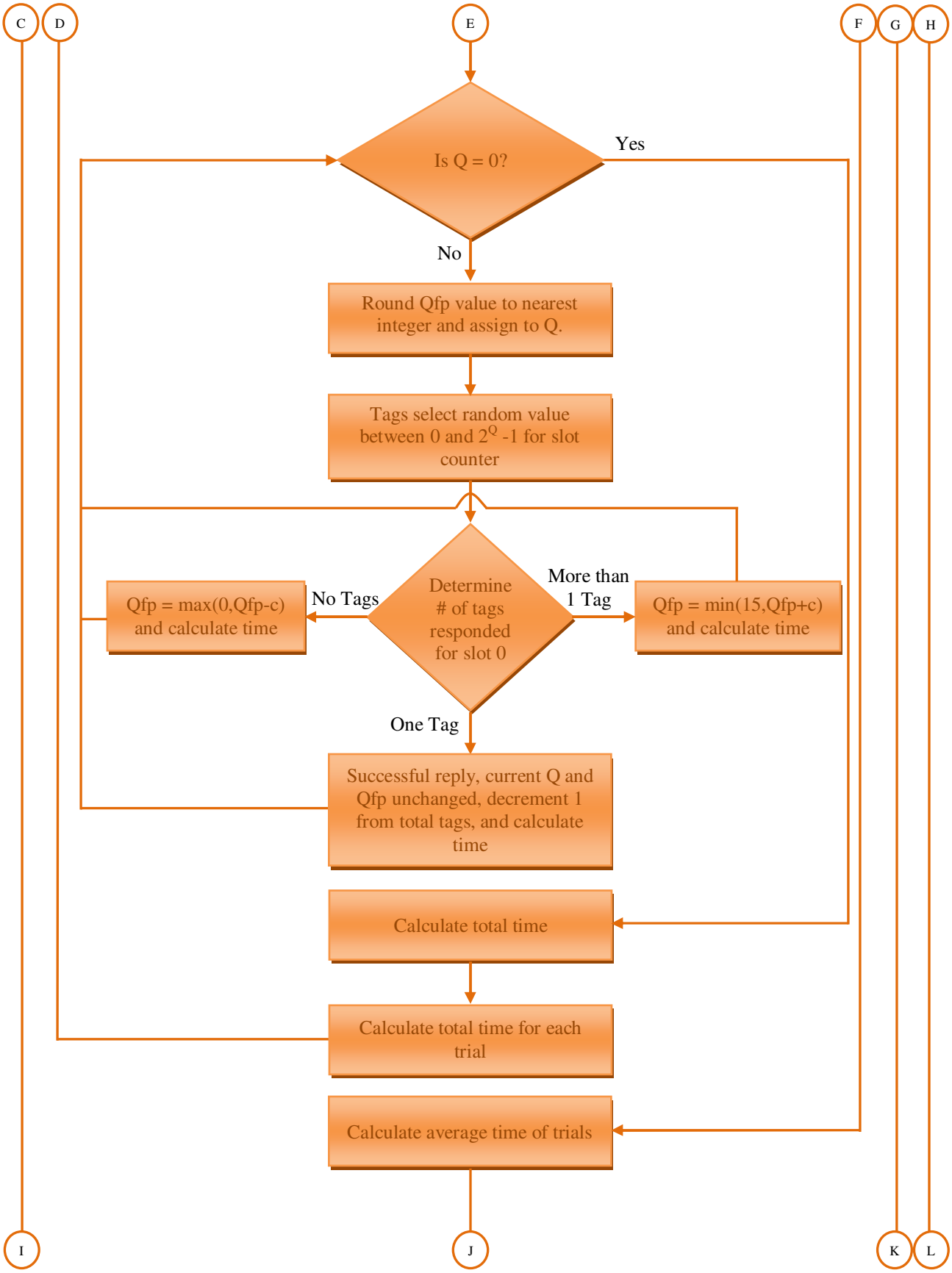


Figure 21: Class 1 Gen 2 Protocol Simulator Flow Chart (Tag Numbers Known)

The flow chart for the Class 1 Gen 2 protocol when the RFID reader does not know the total number of tags in the range in advance is shown in Figure 22.




A
B
Is the simulation run for total number of tags?
Yes
No
Calculate total tags for each scenario, declare array to hold the total time of each trial, and assign slot time
Define number of trials for each scenario
Is simulation run for number of trials?
Yes
No
Set c value, initial Qfp and Q as 4, and assign the total number of tags
C
D
E
F
G
H


C
D
E
F
G
H
Is Q = 0?
Yes
No
Round Qfp value to nearest integer and assign to Q.
Tags select random value between 0 and 2^Q -1 for slot counter
Determine # of tags responded for slot 0
No Tags
Qfp = max(0,Qfp-c) and calculate time
More than 1 Tag
Qfp = min(15,Qfp+c) and calculate time
One Tag
Successful reply, current Q and Qfp unchanged, decrement 1 from total tags, and calculate time
Calculate total time
Calculate total time for each trial
Calculate average time of trials
I
J
K
L

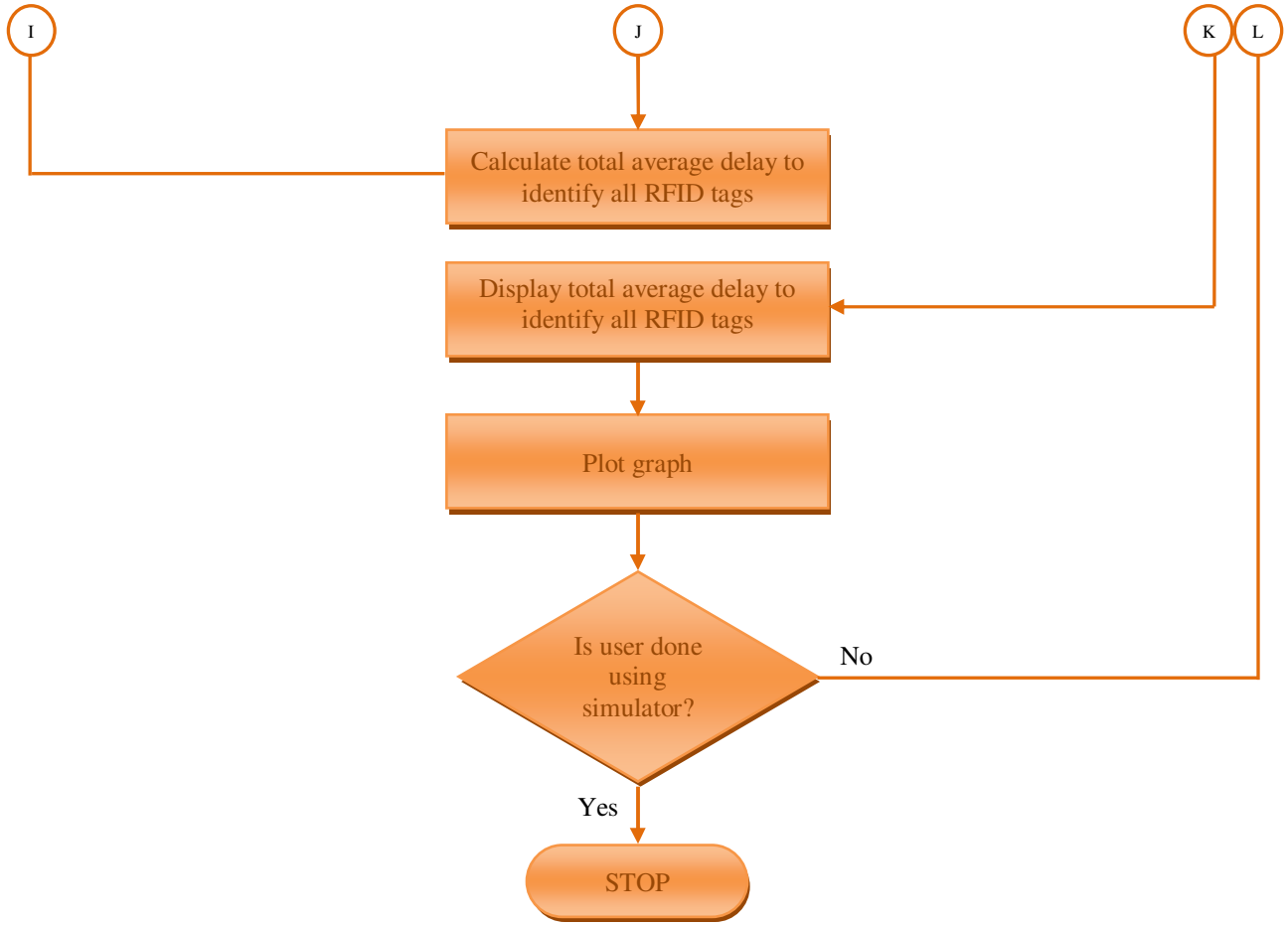


Figure 22: Class 1 Gen 2 Protocol Simulator Flow Chart (Tags Number Unknown)

4.2.2 Code

The entire source code for the RFID Simulator is included in Appendix A. All parameters, variables, and code functions are named in such a way as to allow the users to read, understand, and modify the code without any difficulty. Each and every line of the code is well documented.

4.2.3 Class 1 Gen 2 Performance Tests and Results

The performance of the EPCglobal Class-1 Generation-2 protocol for RFID was evaluated using the RFID Simulator. The simulator provides two options for the EPCglobal

Class-1 Generation-2 protocol. The delay time to identify RFID tags using Class 1 Gen 2 was obtained with both options using the RFID Simulator simulation results.

The first option for Class 1 Gen 2 gives the time to identify all tags when the RFID reader knows the total number of tags in the range beforehand. The user enters the total number of RFID tags that need to be identified, and the simulation will run until all the RFID tags are read successfully. To illustrate how the simulator works, a sample MATLAB Command Window output for Class 1 Gen 2 with ten RFID tags when the total tag number is known in advance by the RFID reader is shown in Figure 23.

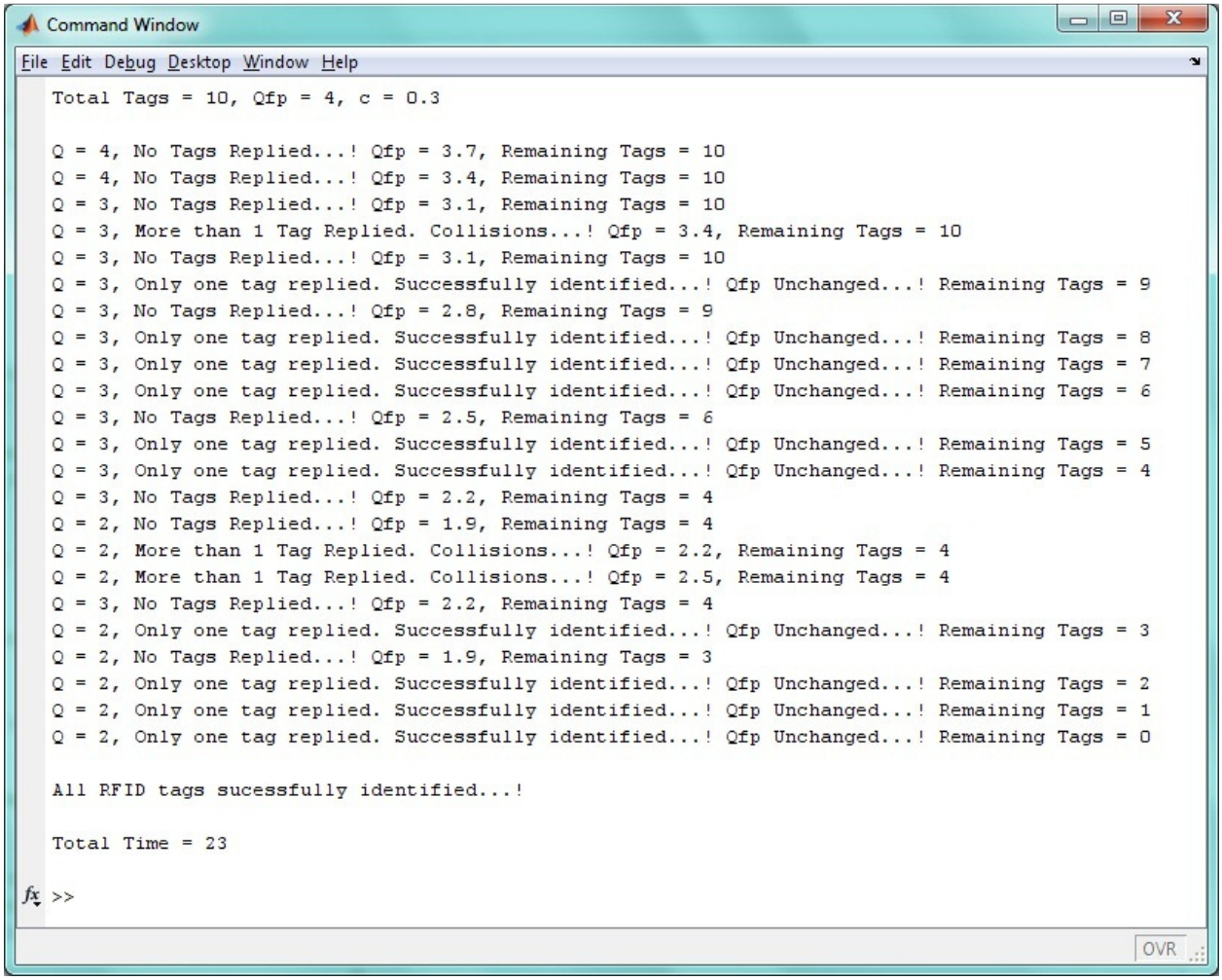


Figure 23: MATLAB Command Window Output Class 1 Gen 2 Protocol (Tag Numbers Known)

Figure 24 shows a sample RFID Simulator-generated graph identifying 100 RFID tags using Class 1 Gen 2 when the total tag number is known in advance by the RFID reader. The total number of RFID tags identified is shown along the x-axis and the total delay or time that it took to identify all RFID tags is shown along the y-axis. The slot time value used was 1 ms, and the value of C was 0.3.

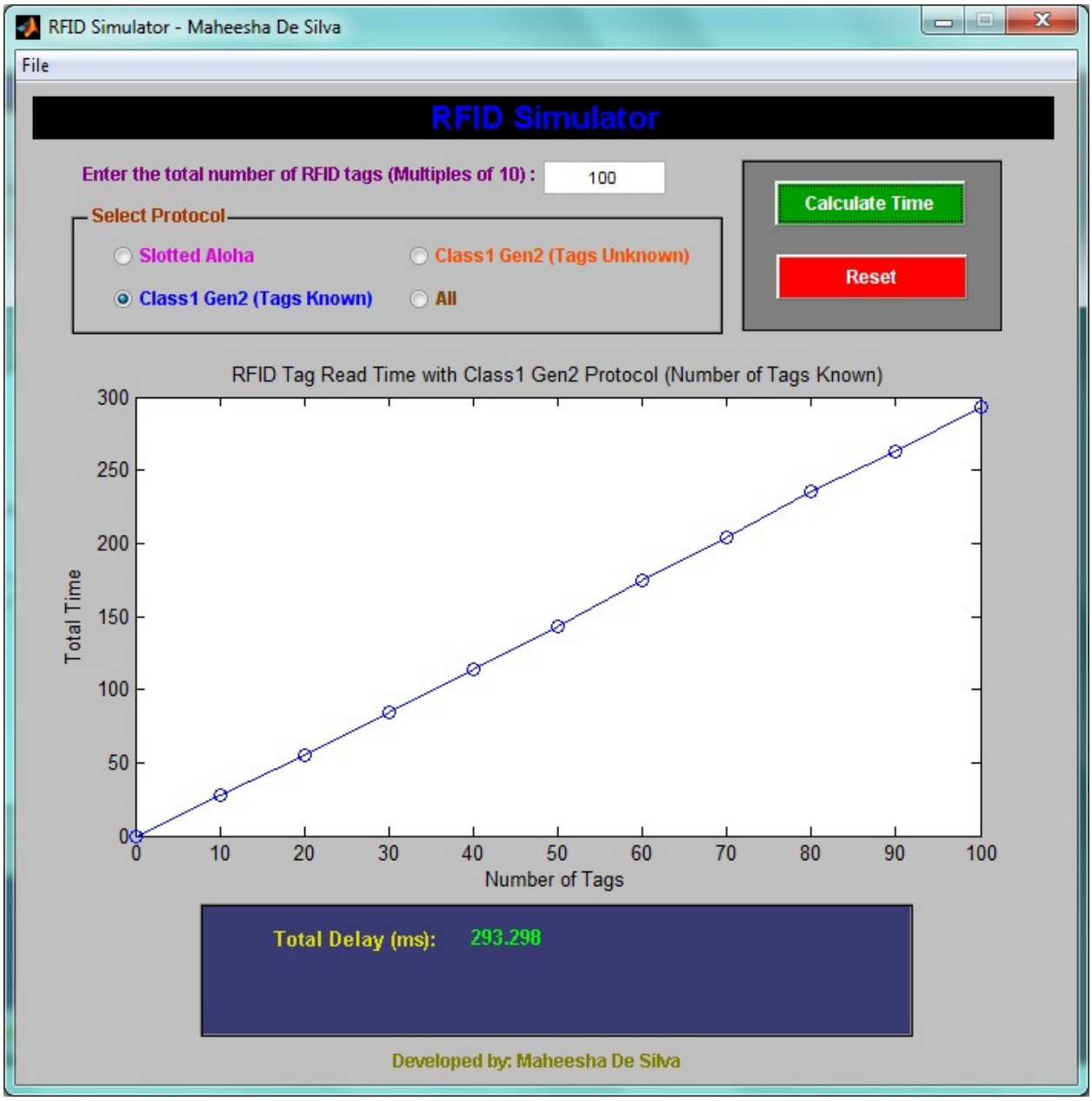


Figure 24: Time to Read 100 RFID Tags Using Class 1 Gen 2 Protocol (Tag Numbers Known)

The second option for Class 1 Gen 2 gives the time to identify all tags when the RFID reader does not know the total number of tags in the range beforehand. The user enters the total number of RFID tags that need to be identified, and the simulation will run until the Q value reaches zero. The Q value reaches zero when the tags no longer reply. The tags will no longer reply when there are no tags present in the field. This means that all the tags were read successfully. To illustrate how the simulator works, a sample MATLAB Command Window output for the Class 1 Gen 2 protocol with ten RFID tags when the total tag number is not known in advance by the RFID reader is shown in Figure 25.

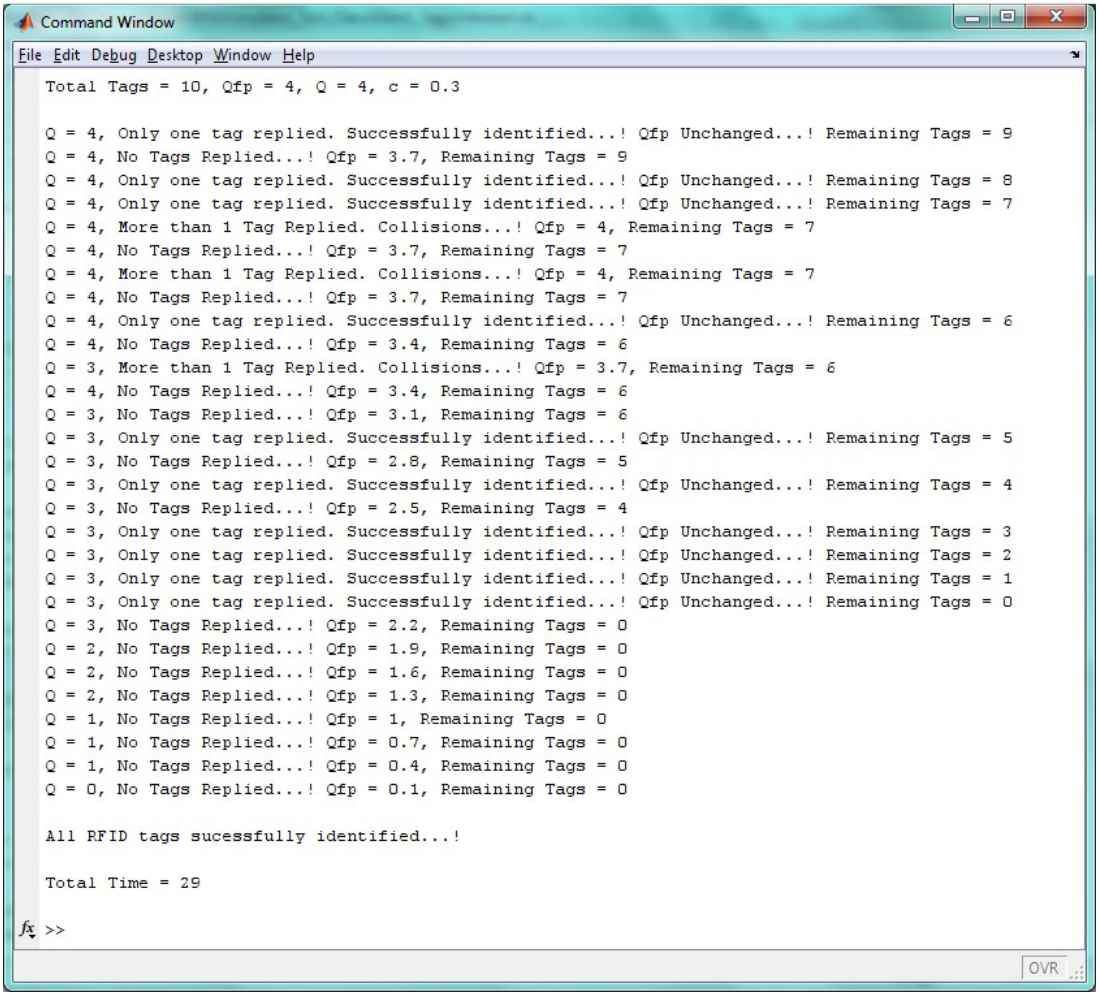


Figure 25: MATLAB Command Window Output Class 1 Gen 2 Protocol (Tag Numbers Unknown)

Figure 26 shows a sample RFID Simulator-generated graph identifying 100 RFID tags using the Class 1 Gen 2 protocol when the total tag number is not known in advance by the RFID reader. The total number of RFID tags identified is shown along the x-axis and the total delay or time that it took to identify all the RFID tags is shown along the y-axis. The slot time value used was 1 ms, and the value of C was 0.3.

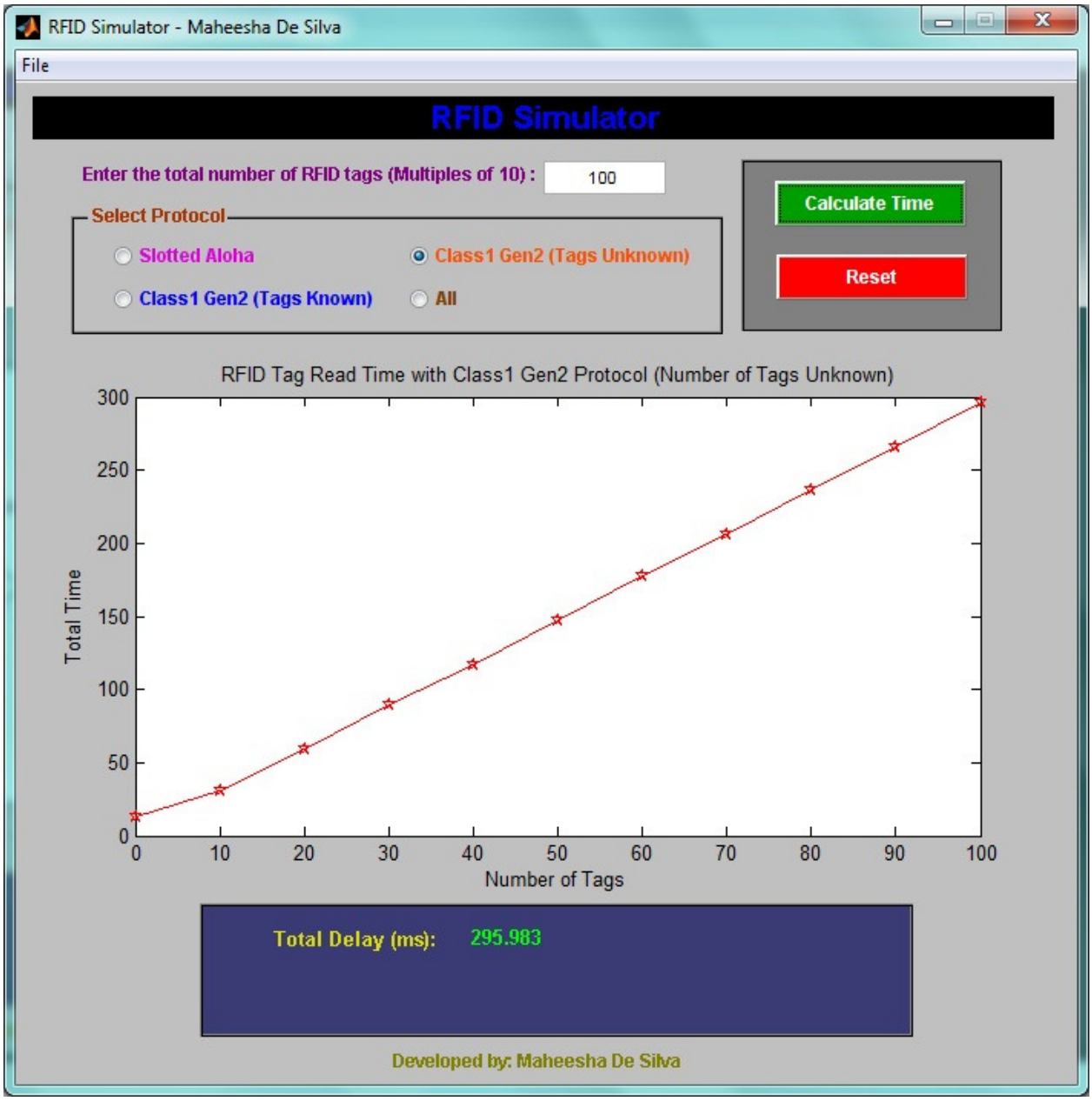


Figure 26: Time to Read 100 RFID Tags Using Class 1 Gen 2 Protocol (Tag Numbers Unknown)

The simulation was run for the following number of tags: 50, 100, 150, 200, 250, 300, 350, 400, 450, and 500. The RFID Simulator ran each simulation scenario for 1,000 times or trials, and the average value was used to obtain the most accurate value. Simulations were run with two different slot times. To compare the simulation results with the Slotted Aloha protocol, the slot time was taken as 1 ms. The following method was used to calculate the other slot times to compare the results with the experimental results in Chapter 5. By comparing Figure 18 to the RFID Simulator, the following reply times were found:

A successful single tag reply time:

$$T_{Query} + T_1 + T_{RN16} + T_2 + T_{ACK} + T_1 + T_{(PC/XPC + EPC + CRC)} + T_2$$

A collided reply time:

$$T_{Query} + T_1 + T_{(Collision\ Detected\ RN16)} + T_2$$

A no reply time:

$$T_{QueryRep} + T_1 + T_3$$

Notes [29]**:**

- **T_1**: Time from reader transmission to tag response.
- **T_2**: Time from tag response to reader transmission.
- **T_3**: Time a reader waits, after T_1, before it issues another command.
- **Query**: A reader starts an inventory round by sending a Query command, with a length of 22 bits, in one of the four sessions.

- **QueryRep**: The QueryRep command, with a length of 4 bits, replicates a preceding Query command without altering any parameters and without introducing any new tags into the round. The tag decrements its slot count each time it gets a QueryRep command.
- **ACK:** This acknowledgment of 18 bits is sent by the reader to the single tag.
- **RN16:** Tags backscatter a 16-bit random number when its slot counter reaches the value zero.
- **Collision-Detected RN16**: Collisions detected at the reader since more than one tag has replied with its 16-bit random number.
- **PC**: A tag backscatters a protocol control or PC, with a length of 16 bits, or the optional extended protocol control (XPC) word or words as a response to the ACK command from the reader. The PC fields consist of EPC length, a user-memory indicator (UMI), an XPC_W1 Indicator (XI), and a numbering system identifier (NSI
- **XPC**: A tag backscatters a PC or the optional XPC word or words, which are exactly 16 bits in length, as a response to the ACK command from the reader. If a tag implements the XPC word 1 (XPC_W1), it will also implement the XPC word 2 (XPC_W2). A tag cannot implement the XPC_W2 without implementing XPC_W1.
- **EPC**: The Electronic Product Code is contained in tag memory and is vendor specific. The EPC size for the ALN-9640 Squiggle Inlay is 96–480 bits [32].
- **CRC**: A cyclic redundancy check, with a length of 16 bits, is used by the tags to guarantee the validity of certain reader-to-tag commands. A reader also uses CRC to guarantee the validity of certain backscattered tags to reader replies.

The tag communication rate or the sub-carrier frequency of the SkyeTek SkyeModule M9 Compact Flash RFID reader is 40 kbps or 40 KHz [33]. It uses an FM0 baseband as the encoding format for the response-to-reader commands. The slot time for tag replies are [29]

$$Tpri = 1/BLF \tag{4.2}$$

Tpri is the backscatter-link pulse-repetition interval or the tag-to-reader link period. Tpri denotes the commanded period of an FM0 symbol. By substituting the value for the backscatter-link frequency (BLF) [29] in equation (4.2),

$$Tpri = \frac{1}{40 * 1024}$$

$$Tpri = 0.00002441406\ s$$

$$Tpri = 0.0244141\ ms$$

The minimum value for T_2 is 3.0 Tpri, and the maximum value is 20.0 Tpri. Since the tag is in the reply state, the maximum value for T_2 applies [29]. Therefore, the time from the tag response to the reader transmission is

$$T_2 = 20.0Tpri$$

$$T_2 = 20*0.02441406$$

$$\boldsymbol{T_2 = 0.4882812\ ms}$$

Since T_1 is hardware specific, it is assumed that $T_1 = T_2$. Therefore, the time from the reader transmission to a tag response is

$$\boldsymbol{T_1 = 0.4882812\ ms}$$

The time a reader waits after T_1 and before it issues another command [29] is

$$T_3 = Tpri$$

$$\boldsymbol{T_3 = 0.0244141\ ms}$$

The time for a Query command, which has a length of 22 bits, is

$$T_{Query} = \frac{22}{40 * 1024}$$

$$T_{Query} = 0.0005371094\ s$$

$$\boldsymbol{T_{Query} = 0.5371094\ ms}$$

The time for a QueryRep command, which has a length of 4 bits, is

$$T_{QueryRep} = \frac{4}{40 * 1024}$$

$$T_{QueryRep} = 0.0000976563\ s$$

$$\boldsymbol{T_{QueryRep} = 0.0976563\ ms}$$

The time for an ACK command, which has a length of 18 bits, is

$$T_{ACK} = \frac{18}{40 * 1024}$$

$$T_{ACK} = 0.0004394531\ s$$

$$\boldsymbol{T_{ACK} = 0.4394531\ ms}$$

The time for RN16, a 16-bit random number, is

$$T_{RN16} = \frac{16}{40 * 1024}$$

$$T_{RN16} = 0.000390625\ s$$

$$\boldsymbol{T_{RN16} = 0.390625\ ms}$$

The time for a collision-detected RN16, also a 16-bit random number, is

$$T_{(Collision\ Detected\ RN16)} = \frac{16}{40 * 1024}$$

$$T_{(Collision\ Detected\ RN16)} = 0.000390625\ s$$

$$\boldsymbol{T_{(Collision\ Detected\ RN16)} = 0.390625\ ms}$$

The time for two XPC words, exactly 16 bits in length, is

$$T_{XPC} = \frac{32}{40 * 1024}$$

$$T_{XPC} = 0.00078125\ s$$

$$\boldsymbol{T_{XPC} = 0.78125\ ms}$$

The ALN-9640 Squiggle Inlay has a 96-bit EPC, which can be extensible to 480 bits [32]. If a tag supports the PC functionally and does not support the XPC functionality, then the maximum length of the EPC that it will backscatter is 496 bits. The maximum length of the tags backscattered by the EPC is subtracted by two words if it supports the XPC functionality. Therefore, the length of the EPC is 464 bits. This is done to make room for the XPC_W1 and optional XPC_W2. In either situation, the tag's reply to the ACK command from the reader shall not go beyond 528 bits in length [29]. For this experiment, 96 bits for the RFID tag EPC length was used. Therefore, the reply of the tag for the reader's ACK command is

XPC + EPC + CRC

32 bits + 96 bits + 16 bits

144 bits (< 528 bits)

Hence, the time for the EPC is

$$T_{EPC} = \frac{144}{40 * 1024}$$

$$T_{EPC} = 0.003515625\ s$$

$$\boldsymbol{T_{EPC} = 3.515625\ ms}$$

The time for the CRC, 16 bits in length, is

$$T_{CRC} = \frac{16}{40 * 1024}$$

$$T_{CRC} = 0.000390625\ s$$

$$\boldsymbol{T_{CRC} = 0.390625\ ms}$$

Using the above values, the slot times for a single tag reply, collided tag reply, and no tag reply were calculated as follows:

$$\textit{Single Tag Reply Time} = T_{Query} + T_1 + T_{RN16} + T_2 + T_{ACK} + T_1 + T_{(PC/XPC + EPC + CRC)} + T_2$$

$$\textit{Single Tag Reply Time} = 0.5371094 + 0.4882812 + 0.390625 + 0.4882812 + 0.4394531 + 0.4882812 + (0.78125 + 3.515625 + 0.390625) + 0.4882812$$

$$\boldsymbol{\textit{Single Tag Reply Time} = 8.0078123\ ms}$$

$$\textit{Collided Reply Time} = T_{Query} + T_1 + T_{(Collision\ Detected\ RN16)} + T_2$$

$$\textit{Collided Reply Time} = 0.5371094 + 0.4882812 + 0.390625 + 0.4882812$$

$$\boldsymbol{\textit{Collided Reply Time} = 1.9042968\ ms}$$

$$\textit{No Reply Time} = T_{QueryRep} + T_1 + T_3$$

$$\textit{No Reply Time} = 0.0976563 + 0.4882812 + 0.0244141$$

$$\boldsymbol{\textit{No Reply Time} = 0.6103516\ ms}$$

Hence, the slot time to identify a single tag reply is 8.0078123 ms, a collided tag reply is 1.9042968 ms, and no tag reply is 0.6103516 ms.

As shown in Figure 27, the delay to identify zero RFID tags in the reader communication range with Class 1 Gen2 when the tag number is known in advance by the RFID reader and for any value of C and for any slot time value is 0 ms. This is because the reader knows the number of RFID tags in the area beforehand, and the simulator ends without any queries.

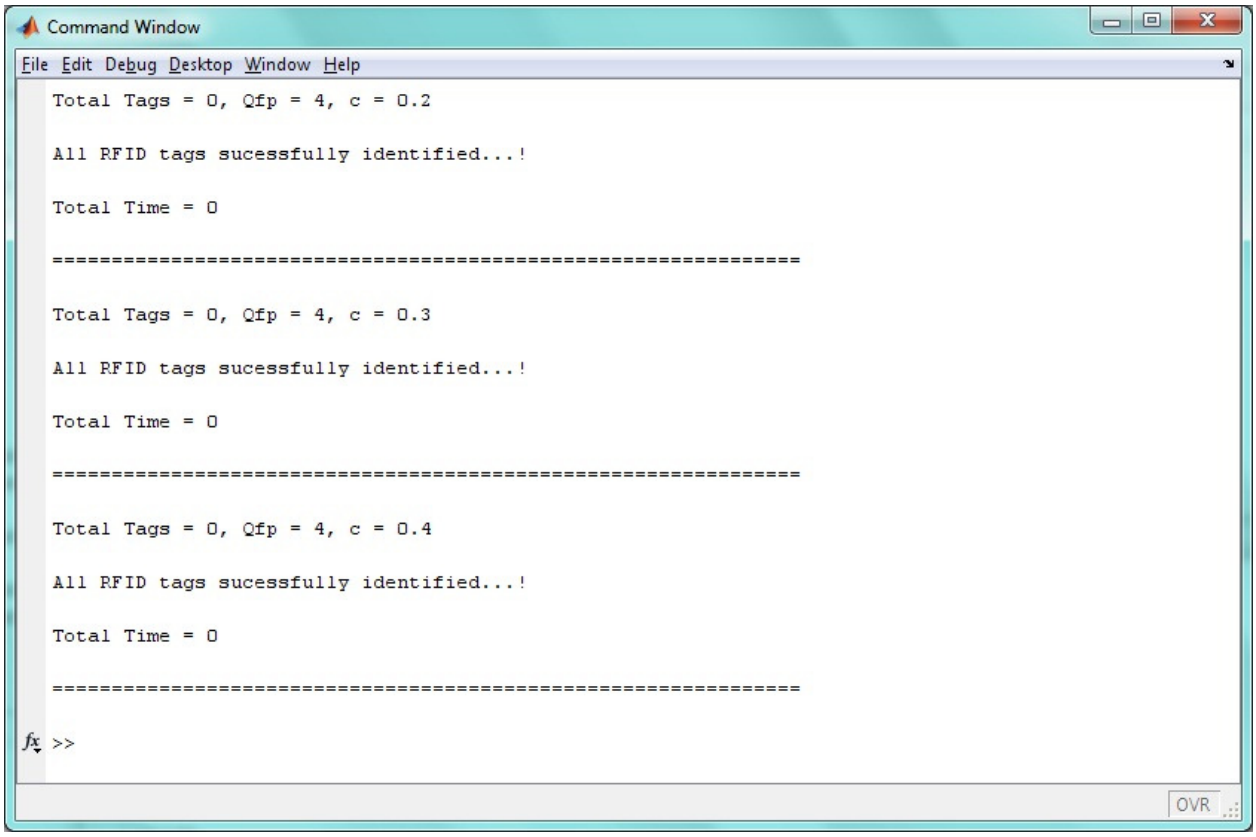


Figure 27: Time to Identify 0 Tags with Class 1 Gen 2 Protocol (Tag Numbers Known)

The delay to identify zero RFID tags in the reader communication range with Class 1 Gen 2, when the tag number is not known in advance by the RFID reader is not 0 ms, since the simulator ends when the value of Q reaches zero. This value depends on the value of C since C affects the Q value. Also, the total time or delay depends on the slot time value. As shown in Figure 28, when the value of C is equal to 0.2 and the slot time is 1 ms, the time to identify zero RFID tags is 19 ms.

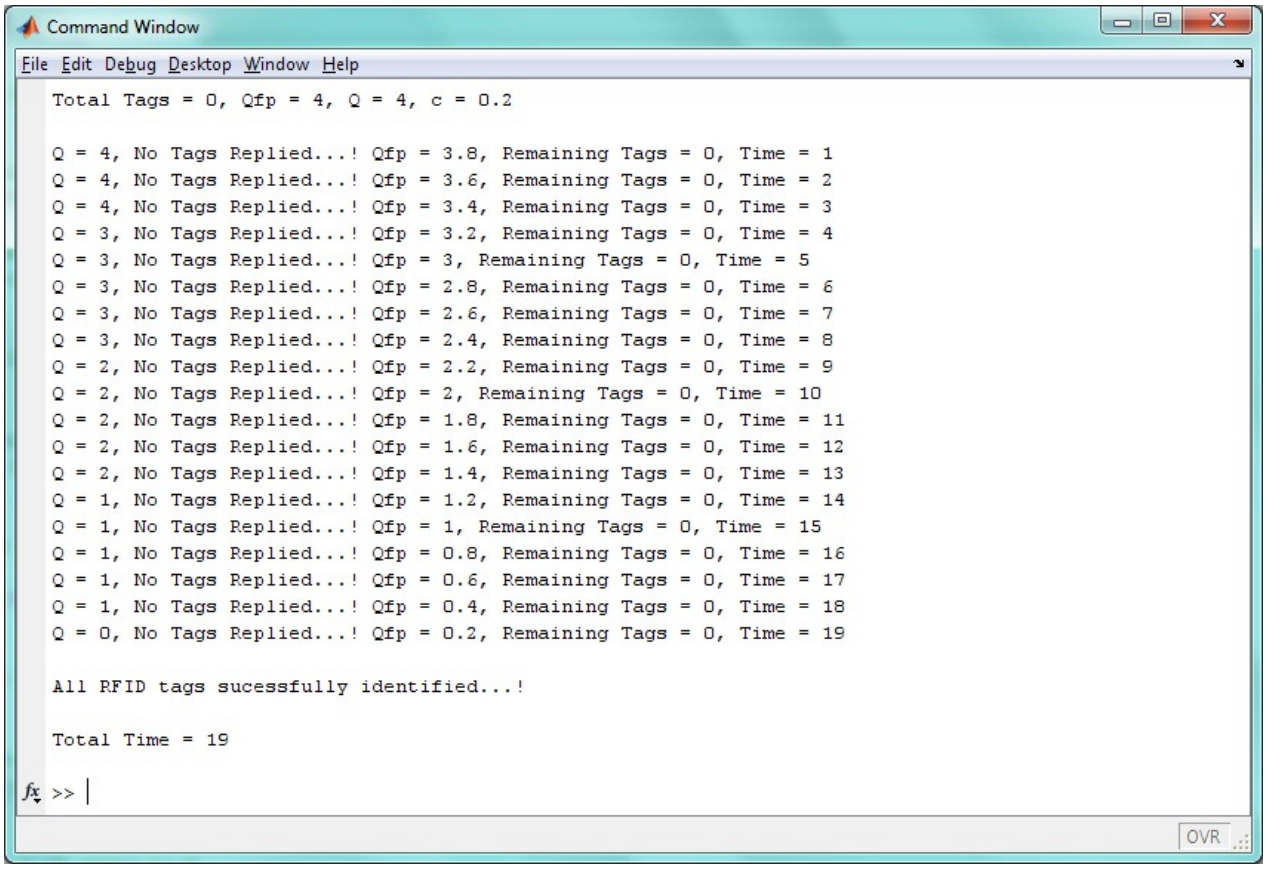


Figure 28: Time to Identify 0 Tags with Class 1 Gen 2 Protocol (Tag Numbers Unknown, C = 0.2, Slot Time = 1 ms)

As shown in Figure 29, when the value of C is equal to 0.2 and with calculated slot times, the time to identify zero RFID tags is 11.5967 ms.

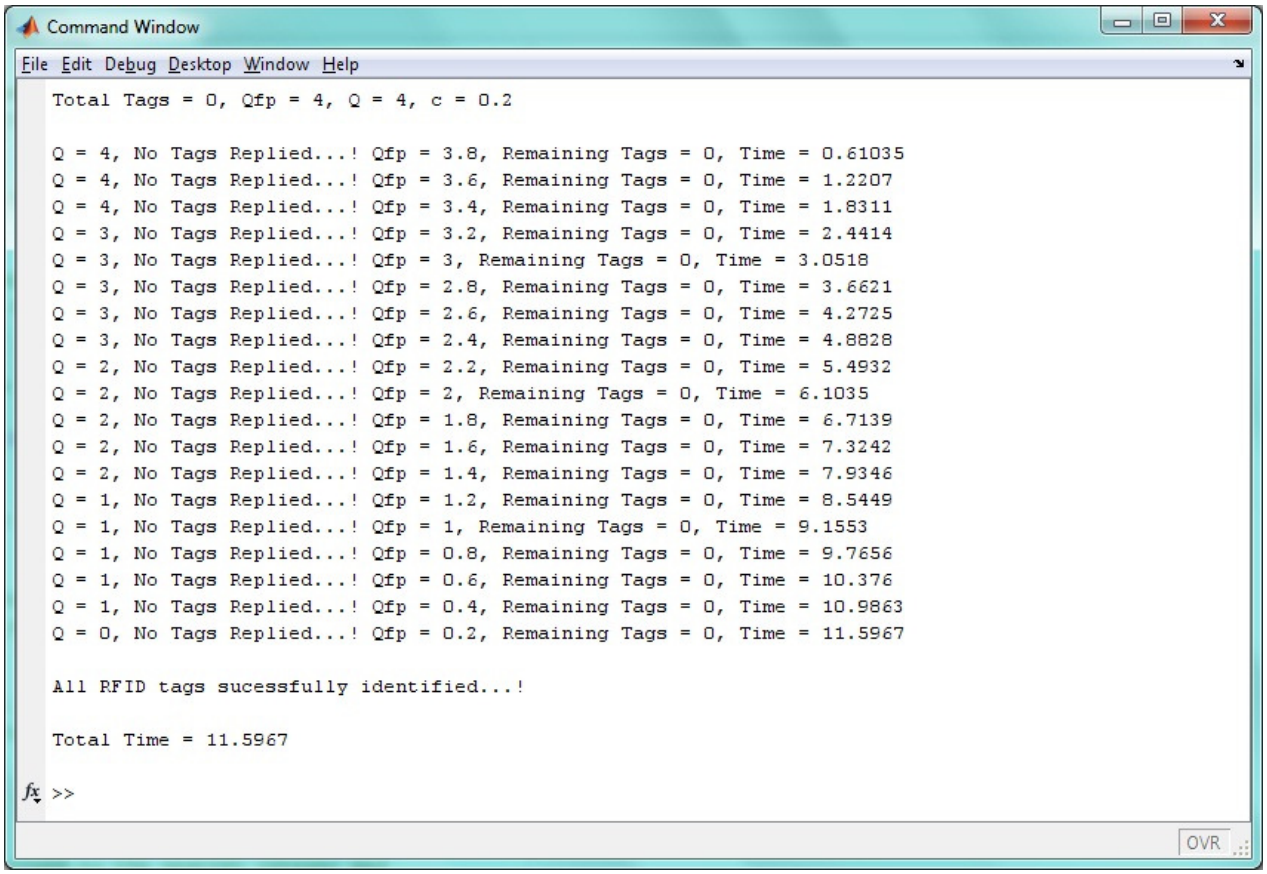


Figure 29: Time to Identify 0 Tags with Class 1 Gen 2 Protocol (Tag Numbers Unknown, C = 0.2, Calculated Slot Times)

The time to identify zero RFID tags in the reader communication range with Class 1 Gen 2, when the tag number is not known in advance by the RFID reader and when the value of C is equal to 0.3 and the slot time is 1 ms, is 13 ms, as shown in Figure 30.

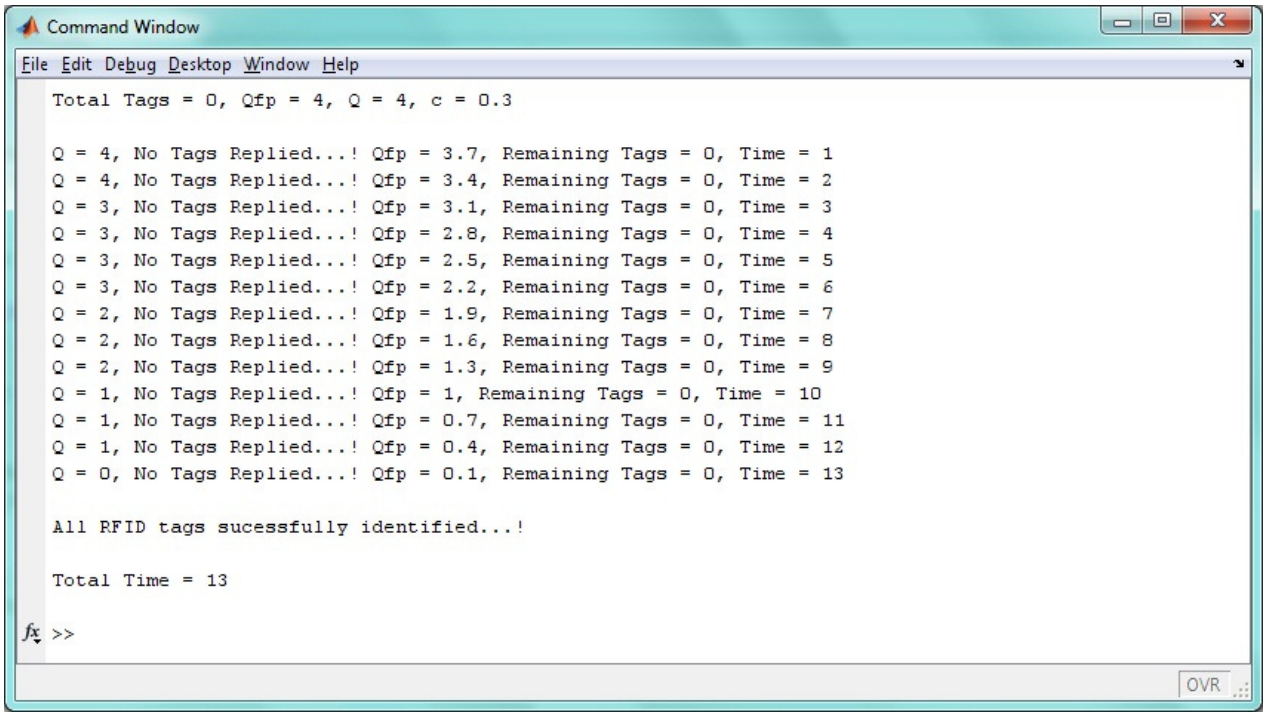


Figure 30: Time to Identify 0 Tags with Class 1 Gen 2 Protocol (Tag Numbers Unknown, C = 0.3, Slot Time = 1 ms)

It takes 7.9346 ms to identify zero RFID tags in the reader communication range with Class 1 Gen 2, when the tag number is not known in advance by the RFID reader and when the value of C is equal to 0.3 and using the calculated slot times, as shown in Figure 31.

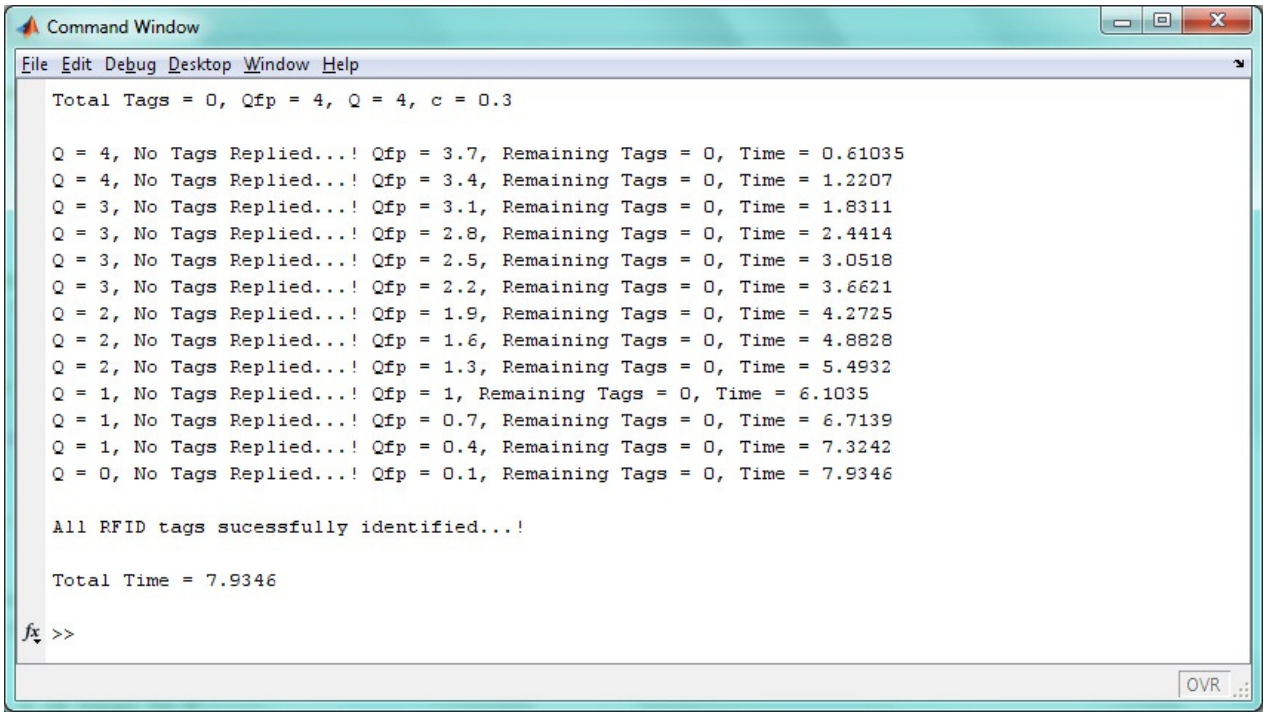


Figure 31: Time to Identify 0 Tags with Class 1 Gen 2 Protocol (Tag Numbers Unknown, C = 0.3, Calculated Slot Times)

The time to identify zero RFID tags in the reader communication range with Class 1 Gen 2, when the tag number is not known in advance by the RFID reader and when the value of C is equal to 0.4 and slot time is 1 ms, is 10 ms, as shown in Figure 32.

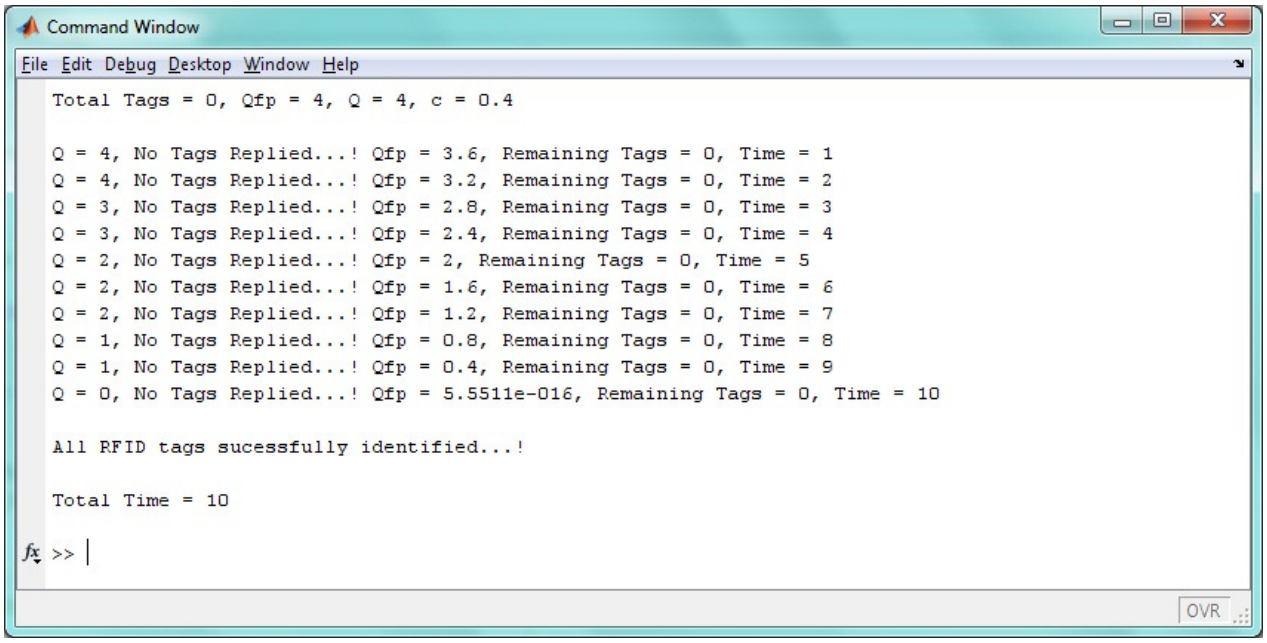


Figure 32: Time to Identify 0 Tags with Class 1 Gen 2 Protocol (Tag Numbers Unknown, C = 0.4, Slot Time = 1 ms)

It takes 6.1035 ms to identify zero RFID tags in the reader communication range with Class 1 Gen 2, when the tag number is not known in advance by the RFID reader and when the value of C is equal to 0.4 and using the calculated slot times as shown in Figure 33.

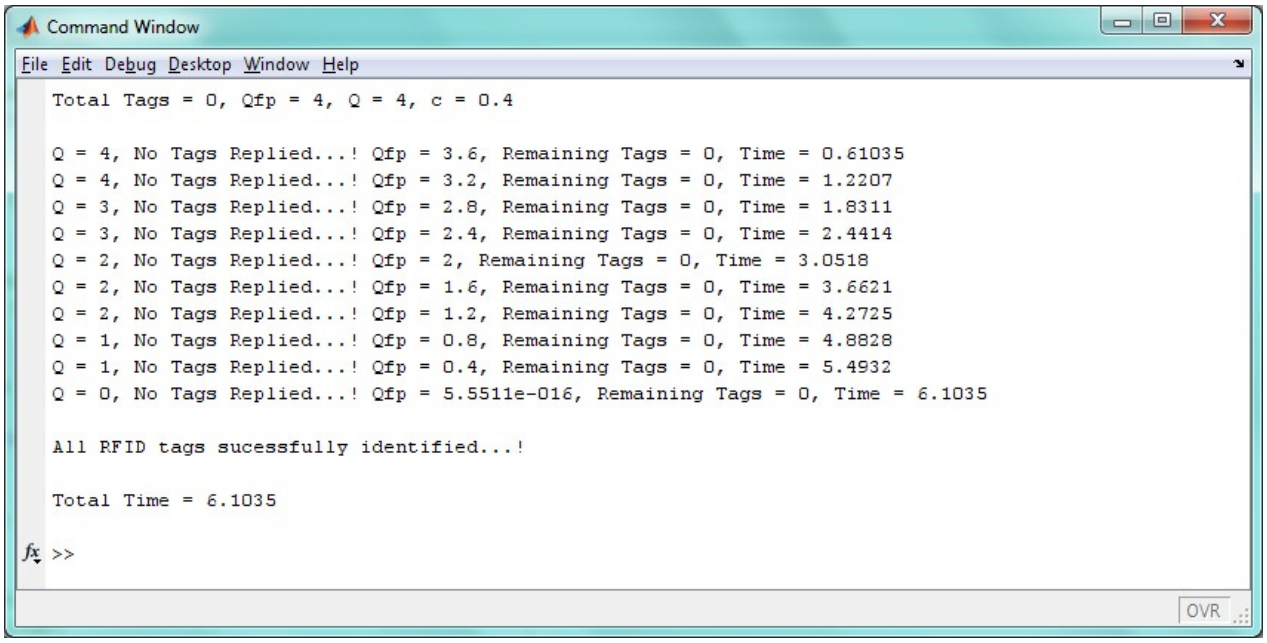


Figure 33: Time to Identify 0 Tags with Class 1 Gen 2 Protocol (Tags Number Unknown, C = 0.4, Calculated Slot Times)

The measured RFID tag read times with Class 1 Gen 2 when the slot time is 1 ms is shown in Table 8.

TABLE 8

SIMULATION RESULTS FOR CLASS 1 GEN 2 WITH 1 MS SLOT TIME

Total Tags	**Delay (ms)** Slot Time = 1 ms					
	Tags Number Known			**Tags Number Unknown**		
	C = 0.2	**C = 0.3**	**C = 0.4**	**C = 0.2**	**C = 0.3**	**C = 0.4**
0	0	0	0	19	13	10
50	144.431	143.594	145.959	151.772	147.210	147.265
100	292.627	294.082	294.556	299.979	296.972	297.113
150	439.615	441.039	446.542	447.226	445.020	447.896
200	583.697	590.599	594.906	592.574	592.368	598.225

Total Tags	Delay (ms) Slot Time = 1 ms					
	Tags Number Known			Tags Number Unknown		
	C = 0.2	C = 0.3	C = 0.4	C = 0.2	C = 0.3	C = 0.4
250	735.019	737.679	745.425	741.316	740.383	747.563
300	877.351	882.435	892.804	884.272	886.286	895.680
350	1024.730	1028.730	1037.390	1029.200	1033.520	1042.370
400	1169.490	1177.210	1190.110	1178.180	1182.110	1192.42
450	1317.600	1323.960	1338.230	1324.680	1327.420	1341.310
500	1458.370	1471.850	1489.350	1471.930	1476.340	1491.320

Figure 34 shows the graphical representation of the RFID tag read times with Class 1 Gen 2 when the slot time is 1 ms.

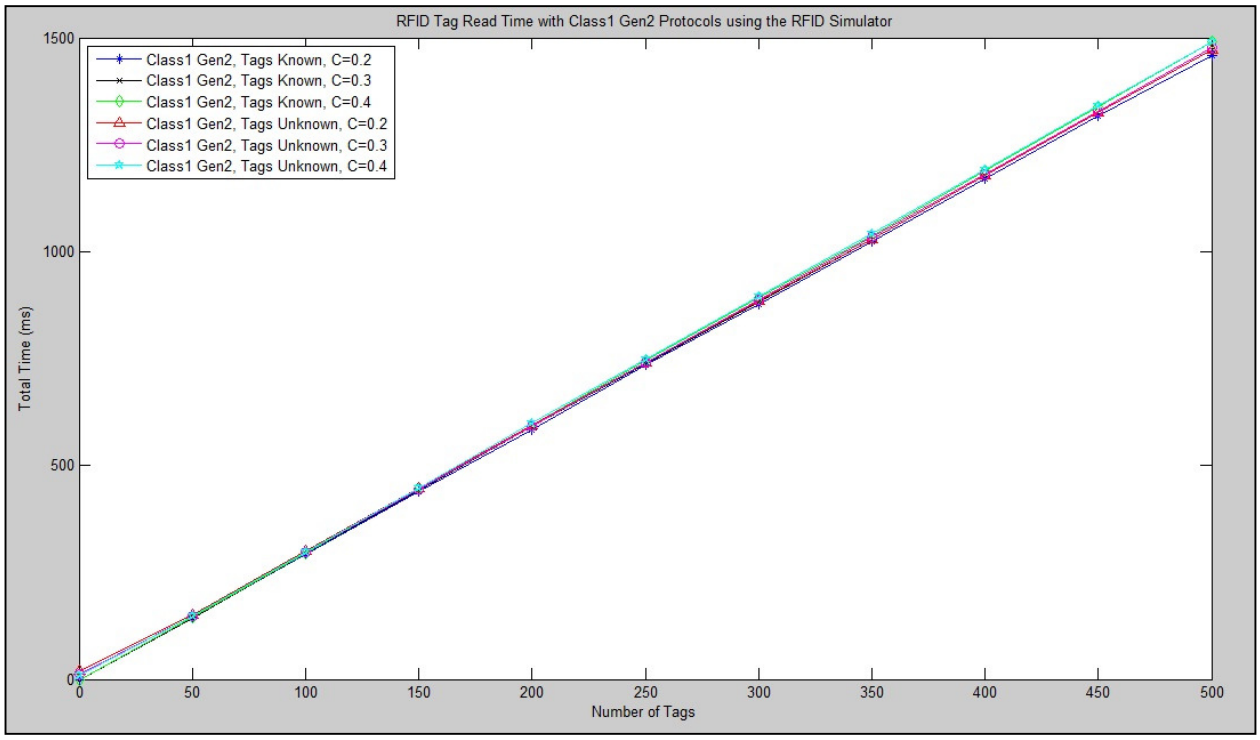


Figure 34: Time to Read 500 RFID Tags Using Class 1 Gen 2 Protocol with 1 ms Slot Time

The measured RFID tag read times with Class 1 Gen 2 when the slot time to identify a single tag reply is 8.0078123 ms, slot time to identify a collided tag reply is 1.9042968 ms, and slot time for the no tag reply is 0.6103516 ms is shown in Table 9.

TABLE 9

SIMULATION RESULTS FOR CLASS 1 GEN 2 WITH CALCULATED SLOT TIMES

Total Tags	**Delay (ms)** Slot Time Single Tag Reply = 8.0078123 ms Slot Time Collided Tag Reply = 1.9042968 ms Slot Time No Tag Reply = 0.6103516 ms					
	Tag Numbers Known			**Tag Numbers Unknown**		
	C = 0.2	**C = 0.3**	**C = 0.4**	**C = 0.2**	**C = 0.3**	**C = 0.4**
0	0	0	0	11.5967	7.9346	6.1035
10	96.019	96.230	96.931	100.839	98.251	97.922
20	197.469	198.289	199.864	201.719	201.245	200.872
30	302.499	303.596	305.473	307.672	305.790	306.808
40	407.228	408.258	410.70	411.091	409.204	412.115
50	512.259	511.586	515.047	515.991	513.917	517.118
60	617.391	617.666	621.633	622.236	619.069	623.501
70	721.902	723.176	725.950	727.449	725.457	726.864
80	825.305	827.535	831.473	830.654	831.112	832.083
90	930.071	932.754	935.518	935.958	936.206	936.921
100	1036.380	1036.330	1041.240	1040.410	1038.940	1041.950

Figure 35 shows the graphical representation of the RFID tag read times with Class 1 Gen 2 using the calculated slot times.

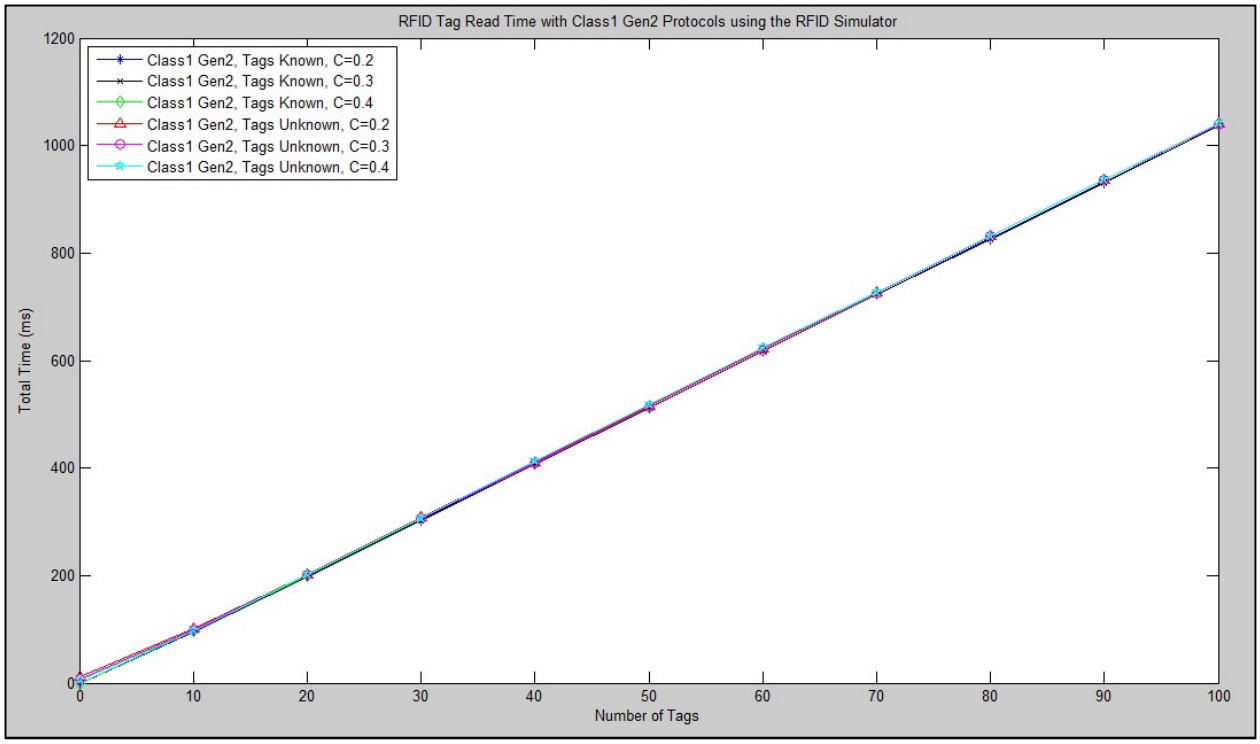


Figure 35: Time to Read 100 RFID Tags Using Class 1 Gen 2 Protocol with Calculated Slot Times

4.3 Simulation Results Comparison

In this section, the RFID Simulator results for the Slotted Aloha protocol are compared to the RFID Simulator results for the Class 1 Gen 2 protocol when the slot time is equal to 1 ms. Figure 36 shows a snapshot of the RFID Simulator for 50 RFID tags when all protocols (Slotted Aloha, Class 1 Gen 2 with tag numbers known by the reader, and Class 1 Gen 2 with tag numbers not known by the reader) are selected.

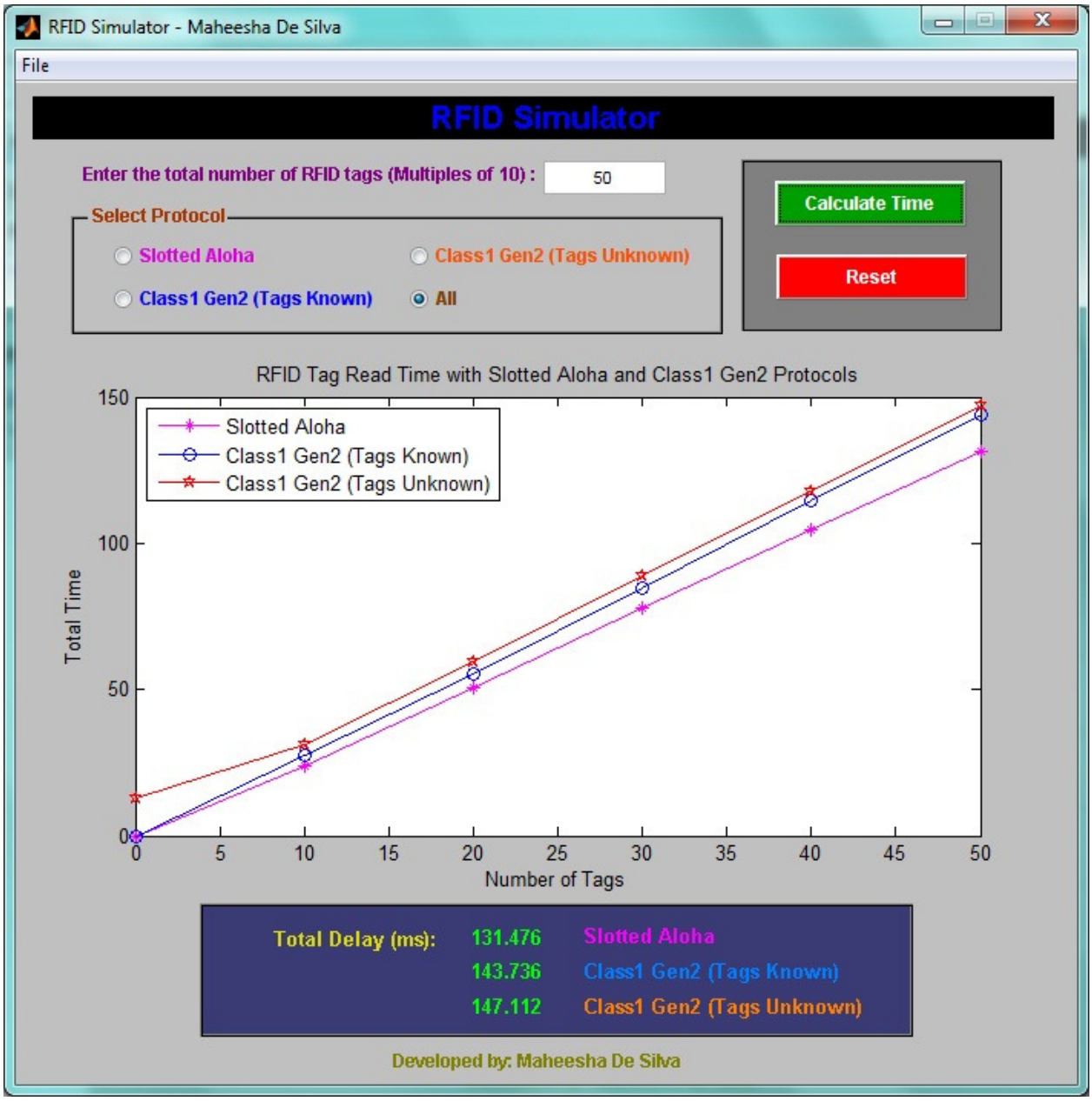


Figure 36: Time to Read 50 RFID Tags Using All Protocols

As shown in Figure 37, the delay to identify zero RFID tags in the reader communication range with the Slotted Aloha protocol with any value for the slot time is 0 ms since the reader knows the number of RFID tags in the area beforehand and the simulator ends without any requests.

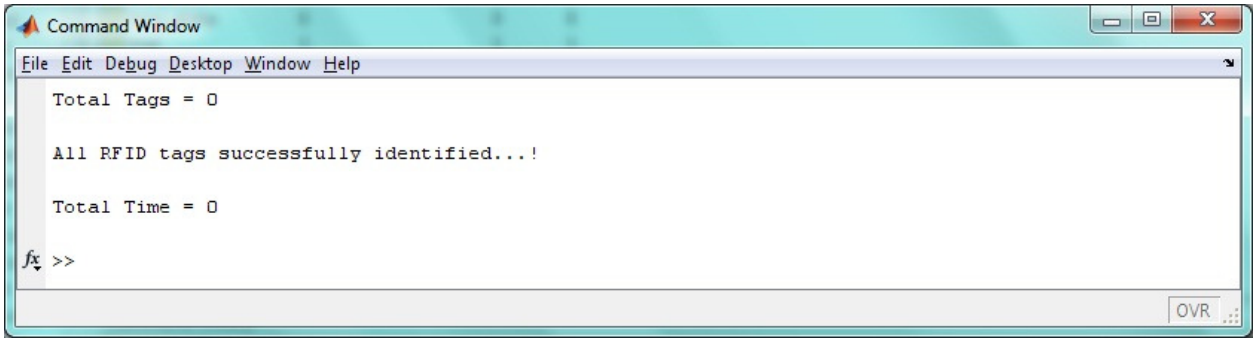


Figure 37: Time to Identify 0 Tags with Slotted Aloha Protocol

The measured RFID tag read times with the Slotted Aloha protocol, Class 1 Gen 2 protocol with tag numbers known by the reader, and Class 1 Gen 2 protocol with tag numbers not known by the reader are shown in Table 10.

TABLE 10

SIMULATION RESULTS FOR SLOTTED ALOHA AND CLASS 1 GEN 2 PROTOCOLS

Total Tags	**Delay (ms)** Slot Time = 1ms						
	Slotted Aloha	**C1G2 Tag Numbers Known**			**C1G2 Tag Numbers Unknown**		
		C = 0.2	**C = 0.3**	**C = 0.4**	**C = 0.2**	**C = 0.3**	**C = 0.4**
0	0	0	0	0	19	13	10
50	131.622	144.431	143.594	145.959	151.772	147.210	147.265
100	266.072	292.627	294.082	294.556	299.979	296.972	297.113
150	402.269	439.615	441.039	446.542	447.226	445.020	447.896
200	538.775	583.697	590.599	594.906	592.574	592.368	598.225
250	673.147	735.019	737.679	745.425	741.316	740.383	747.563
300	810.550	877.351	882.435	892.804	884.272	886.286	895.680
350	945.227	1024.730	1028.730	1037.390	1029.200	1033.520	1042.370
400	1080.380	1169.490	1177.210	1190.110	1178.180	1182.110	1192.42

Total Tags	Delay (ms) Slot Time = 1ms						
	Slotted Aloha	C1G2 Tag Numbers Known			C1G2 Tag Numbers Unknown		
		C = 0.2	C = 0.3	C = 0.4	C = 0.2	C = 0.3	C = 0.4
450	1218.740	1317.600	1323.960	1338.230	1324.680	1327.420	1341.310
500	1350.300	1458.370	1471.850	1489.350	1471.930	1476.340	1491.320

Figure 38 shows the graphical representation of the RFID tag read times for the Slotted Aloha protocol, Class 1 Gen 2 protocol with tag numbers known by the reader, and Class 1 Gen 2 protocol with tag numbers not known by the reader.

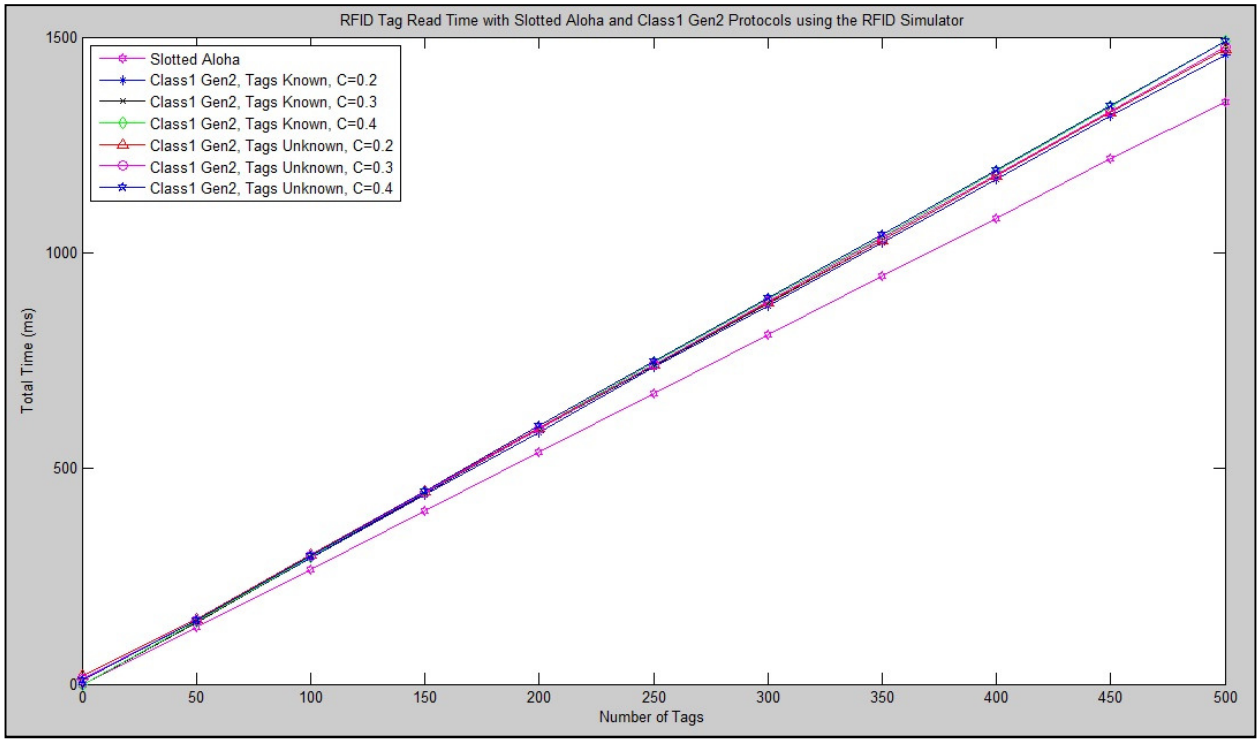


Figure 38: Time to Read 500 RFID Tags Using Slotted Aloha and Class 1 Gen 2 (Tag Numbers Known and Unknown by the Reader) Protocols

According to Figure 38, the Slotted Aloha protocol has the least delay in identifying RFID tags because with this protocol, the simulator gives the maximum throughput since the size

of the frame or the number of time slots was set equal to the total number of RFID tags in the area. Alternatively, the Q algorithm will obviously have a larger delay since the reader can count down only one slot at a time, slot 0, since only tags with a slot counter of zero are allowed to respond. Conversely, with the Slotted Aloha protocol, tags have the capability to respond to any slot.

In this chapter, the performance of the EPCglobal Class-1 Generation-2 anti-collision protocol was evaluated using the RFID Simulator and the results compared to those of the Slotted Aloha protocol. The next chapter shows an experimental study of the EPCglobal Class-1 Generation-2 anti-collision protocol and compares the results with the RFID Simulator values.

CHAPTER 5

PRACTICAL IMPLEMENTATION AND RESULTS COMPARISON

5.1 Testbed

A testbed was constructed to confirm the simulation results with real-world RFID systems. The read time for EPCglobal Class-1 Generation-2 (ISO 18000-6C) RFID tags was obtained using the testbed. The components used for the testbed and their features are explained in detail in the following sections.

5.1.1 Tags

The Alien Technology Squiggle Inlay RFID tags were used for the testbed. These tags are one of the most extensively used inlays for Class 1 Gen 2 and are specially designed for supply chain, asset tracking, and inventory management. These tags operate in frequencies between 860 and 960 MHz and support all compulsory and optional Class 1 Gen 2 commands. The Alien Squiggle Inlay tag is powered by the leading Higgs-3 IC. It has a 96-bit EPC memory bank that is capable of being extended to 480 bits [32]. The datasheet for the Alien Gen2 Squiggle RFID tags is included in Appendix B. Figure 39 shows an image of the Alien Gen2 Squiggle RFID tags.

Figure 39: Alien Gen2 Squiggle RFID Tags

5.1.2 External UHF Antenna

The antenna that was used for the testbed is an external RFID broadband UHF antenna for the M9 module. This antenna has the capability to work on every regulatory environment worldwide since its input frequency is 860–960 MHz. The RFID broadband UHF antenna offers a long range for the embedded UHF. It has a gain of 6.5 dBi and a linear polarization. The nominal input impedance is 50 Ohm. The external antenna cable connects directly to the micro-miniature coaxial (MMCX) connector on the M9 reader using a subminiature version A (SMA) cable along with an MMCX-SMA adapter [34]. The datasheet for the external RFID broadband UHF antenna is included in Appendix C. Figure 40 shows an image of the external RFID broadband UHF antenna.

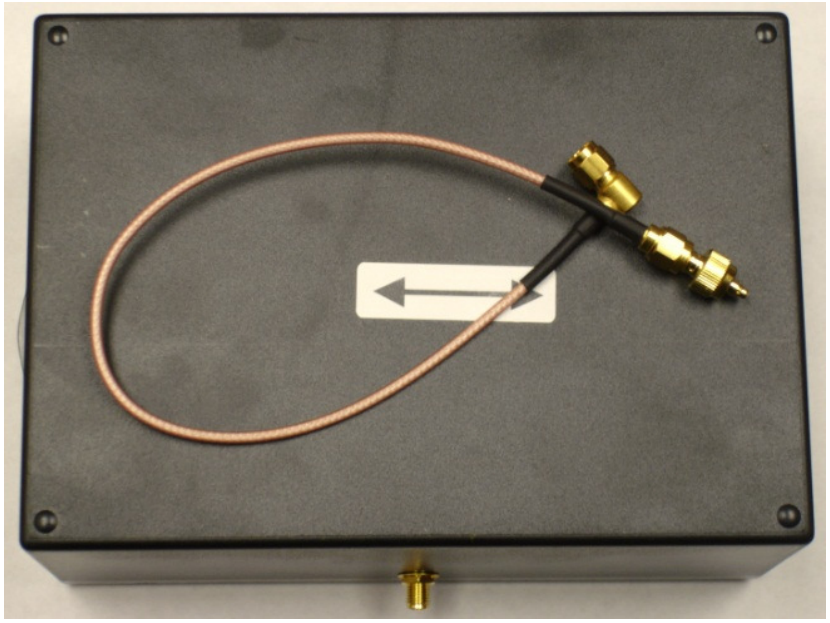


Figure 40: External RFID Broadband UHF Antenna

5.1.3 Reader

The RFID reader that was used for the testbed is the SkyeModule M9 Compact Flash form factor RFID reader from SkyeTek. This tiny, embedded UHF RFID reader has the capability to support EPC Class 1 Gen 1, ISO 18000-6B, ISO 18000-6C (EPC Class 1 Gen 2), and ISO 18000-6A5 tags. It operates in the frequency range of 862–955 MHz [35]. The M9 reader has a read range of 1 m to approximately 3.5 m with a 6 dBi linearly polarized antenna, and it complies with worldwide regulatory requirements. It also has software-configurable output power in the range of 10–27 dBm. The SkyeModule M9 reader is specifically designed for supply chain management, inventory management, access control, product authentication, anti-counterfeiting, handheld reading/encoding, printing, and patron management [33]. The SkyeModule M9 reader details are as follows:

- Firmware Version: 090100D6
- Firmware Personality: Standard
- Hardware Version: 01000004

- Product Code: 0009

The datasheet for the SkyeModule M9 compact flash RFID reader is included in Appendix D. Figure 41 shows an image of the SkyeModule M9 compact flash RFID reader.

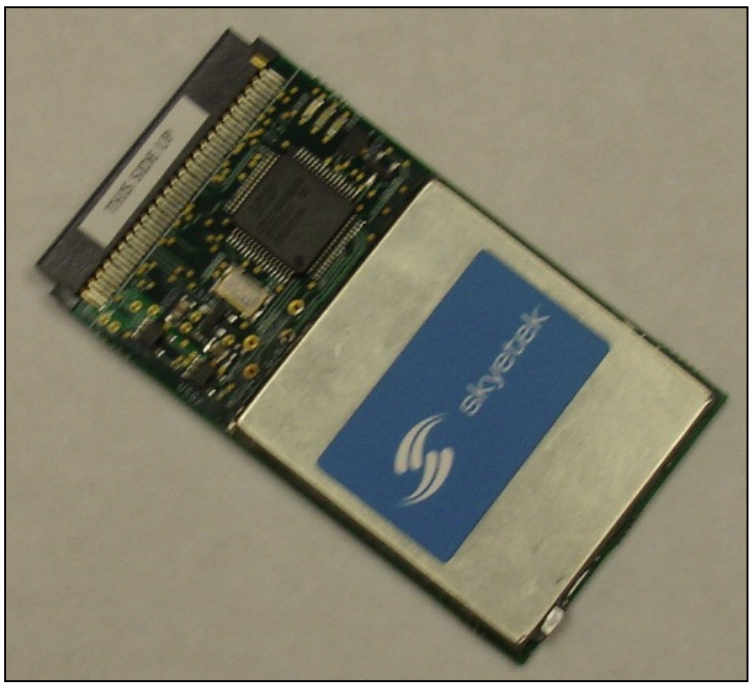


Figure 41: SkyeModule M9 Compact Flash RFID Reader

5.1.4 Common Blade Interface Board

The SkyeTek common blade interface board (CBiB) has the capability to support both the Compact Flash M9 reader and the Mounting Hole M9 reader form factors. The host interface board has a 50-pin, double-row connector on the top for the Compact Flash and a 24-pin, double-row connector on the bottom for the Mounting Hole. It also offers a variety of host interfaces, such as transistor-transistor logic (TTL), universal serial bus (USB), serial peripheral interface (SPI), and inter-integrated circuit (I2C). The TTL has a data rate of 9.6–115.2 kb/s, the USB 2.0 has a full speed of 12 Mb/s, the SPI has a data rate up to 10MHz, and the I2C has a data rate of 100/400 KHz. The common blade interface board also has an AC power connector to provide

power to the interface board and the RFID reader [36]. Figure 42 shows a picture of the SkyeTek common blade interface board.

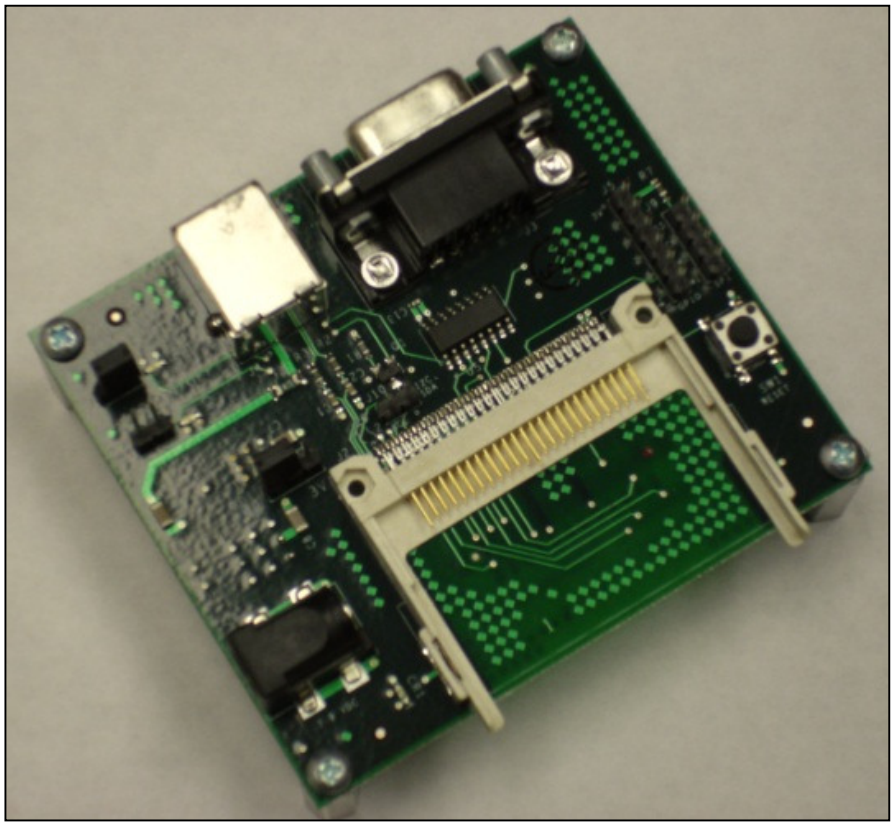


Figure 42: SkyeTek Common Blade Interface Board

5.1.5 Host Interface

A Universal Serial Bus 2.0 connection was used for the host interface board to host communication. The host detects the SkyeModule M9 reader as a USB device.

5.1.6 Host Computer

A Lenovo ThinkPad T61 X86-based personal computer (PC) with genuine Windows Vista Home Premium operating system was used as the host computer for the testbed. It has an Intel Core 2 Duo T7500 processor with 2 GB total memory.

5.1.7 Software

The SkyeTek SkyeWare 4 Version 4.2.0.1378 was used as the software for the testbed. SkyeWare 4 is a complete RFID reader application, which provides a powerful and simple interface to assess embedded RFID. Developers are able to instantly configure the reader and start testing the code by connecting a SkyeModule RFID reader to a host computer that has SkyeWare installed. The main feature of SkyeWare 4 is Anti-Collision. Anti-Collision shows the reader's anti-collision ability. It allows for testing the read rate and interference with having many tags in the reader field. Another feature is the Read Range. This feature allows the users to interpret and evaluate the read range of different tags through a graphical and user-friendly interface. SkyeWare 4 allows for configuring reader settings, radio parameters, and port settings to fit specific user conditions. Tag settings can be configured on a tag-by-tag basis in order to maximize performance. Other features include product authentication, access control, memory, and secure memory. All commands are sent to the reader using the SkyeTek Protocol v3 (STPv3) [37]. The datasheet for the SkyeTek SkyeWare 4 is included in Appendix E. Figure 43 shows a screenshot of the SkyeTek SkyeWare 4 homepage.

Figure 43: SkyeTek SkyeWare 4

5.2 Practical Evaluation

An experimental study was performed to calculate the RFID tag read times using the RFID testbed, as shown in Figure 44.

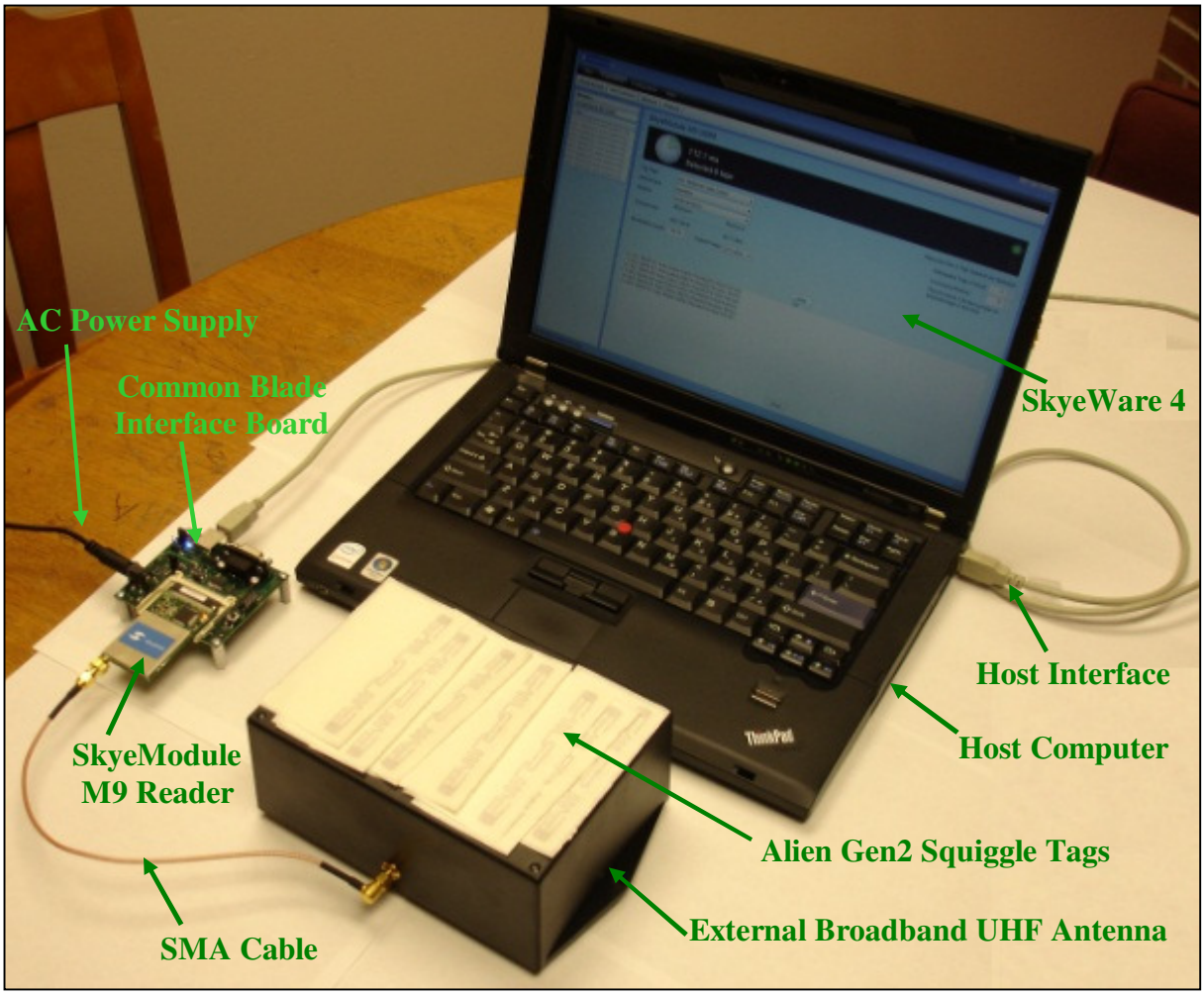


Figure 44: RFID Testbed

The SkyeTek M9 RFID reader was set to operate in session 1 during the inventory process. The SkyeTek SkyeModule M9 Compact Flash RFID reader has an 860–960 MHz carrier frequency and a tag communication rate (data rate) of 40 KHz or 40kbps, which is the subcarrier frequency since it uses FM0 baseband as the encoding format for the response-to-reader commands [33]. The inventory or the anti-collision mode of the reader was used to calculate the delay of the RFID tag identification. The reader chooses all RFID tags in the discovery area until a new RFID tag is found in the area or until the inventory process times out. The SkyeTek SkyeModule M9 Compact Flash RFID reader's Command Retry system parameter

controls how many times the operation is repeated. This parameter can be increased to improve reliability and affect the amount of time the operation takes, which is the timeout value of the inventory. The Command Retry value states the number of times a tag command is executed internally in the reader before responding with a failure response. The command is repeated internally for the number of times specified, unless there is a successful response. When a successful response occurs, the reader stops repeating the command and sends a success response back to the user. SkyeTek suggests setting the Command Retry system parameter to approximately double the number of expected tags in the area to optimize the inventory process [35]. But this increases the inventory time unnecessarily since the time that the reader takes before sending a failure message, when a failure occurs, gets increased. Therefore, to reduce unnecessary inventory time, the Command Retry time was decreased until the shortest time was found, at which the reader could identify all tags in the field. The Figure 45 shows the protocol messages output of the SkyeTek SkyeWare 4 when the Command Retry parameter (0x0011) is set to value 15.

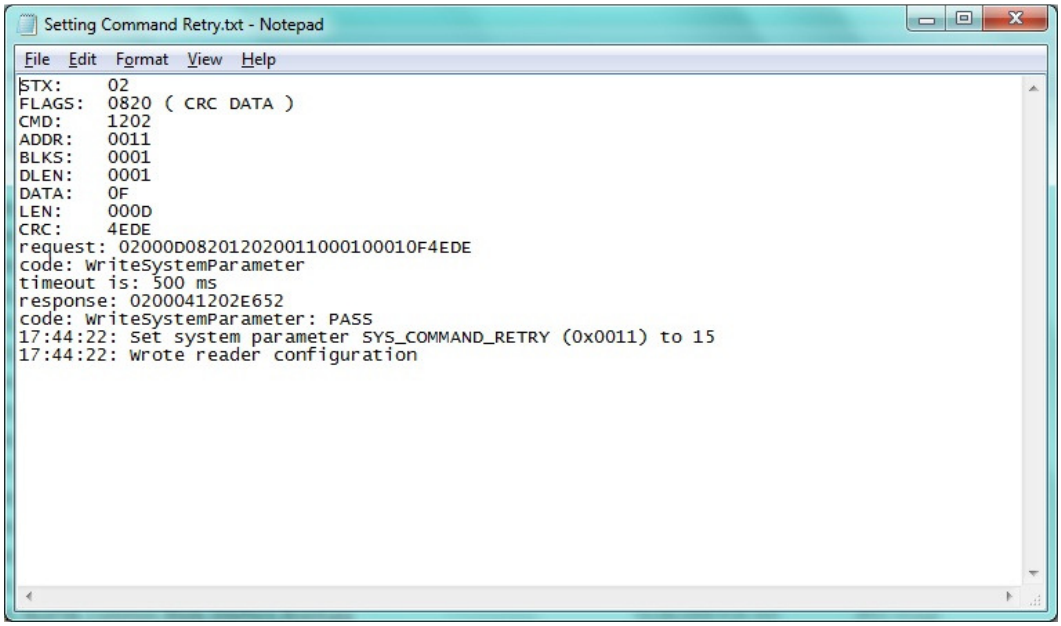


Figure 45: SkyeTek SkyeWare 4 Protocol Messages for Command Retry Value 15

The following SkyeWare 4 settings were used during the inventory process, according to the regional compliance of North America.

- Start Frequency: 902.3 MHz
- Current (Center) Frequency: 915.0 MHz
- Stop Frequency: 927.7 MHz
- Output Power: 27 dBm
- Hop Channel Spacing: 200 KHz
- Modulation Depth: 100 %
- Frequency Hop: Random
- Regulatory Mode: FCC

Random frequency hopping was used for the frequency hopping sequence since it improves reader performance by issuing the commands on different frequencies [35].

Figure 46 shows the SkyeWare4 protocol messages output when identifying 10 Alien Gen2 Squiggle RFID tags using the SkyeTek SkyeModule M9 Compact Flash RFID reader.

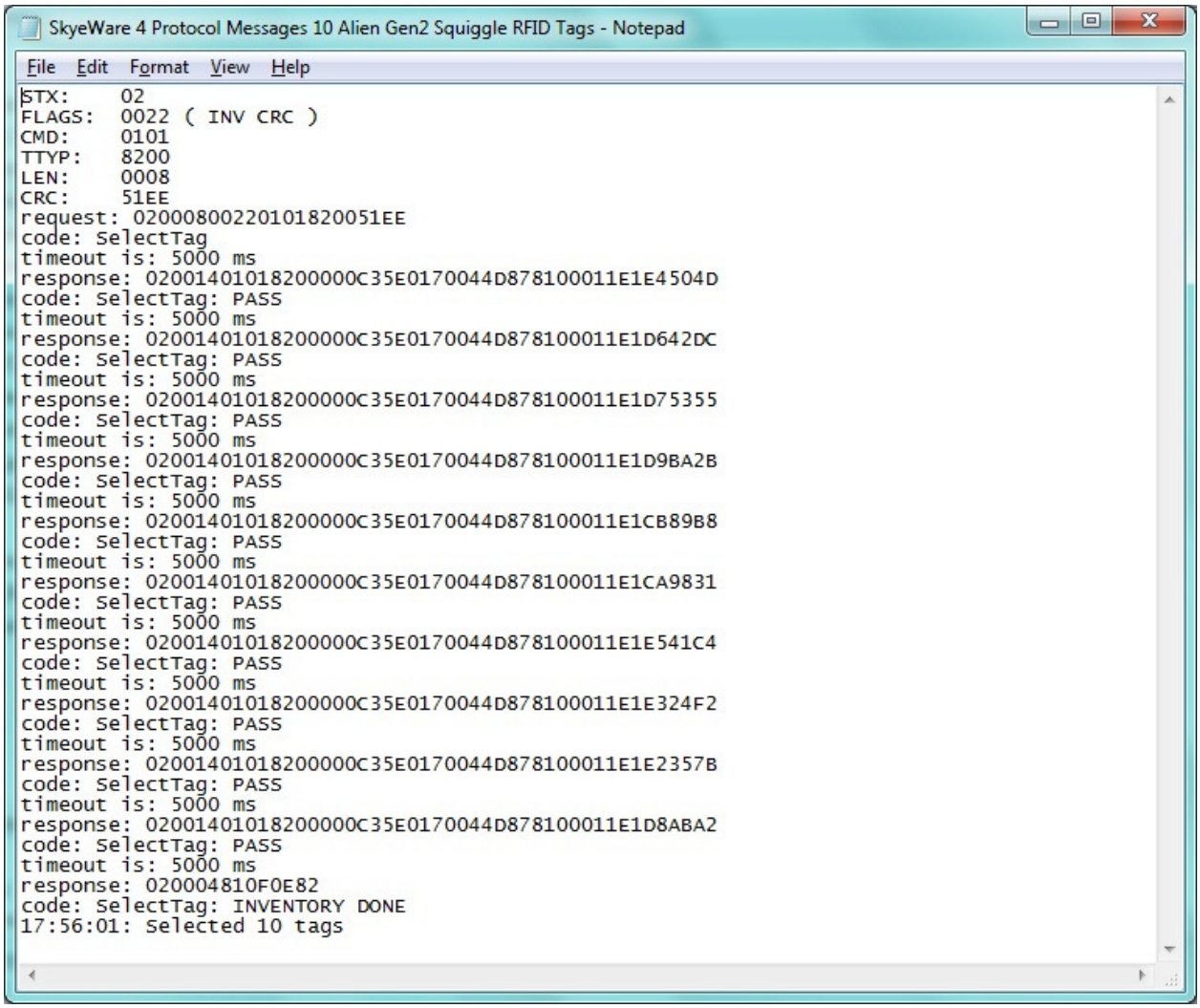


Figure 46: SkyeWare4 Protocol Messages Identifying 10 Alien Gen2 Squiggle RFID Tags

For the experimental study, a maximum of 40 RFID tags was identified by using the testbed. This was due to the limitations with the power level and the receive sensitivity of the SkyeTek M9 RFID reader. Another reason was the environment interference. At one point, there was too much interference, and at another point, there was no interference. The Command Retry parameter value and the way the tags were placed had to be manipulated in order to overcome these rigid situations. Figure 47 shows a snapshot of how the Alien Gen2 Squiggle RFID tags were placed in one of the inventories.

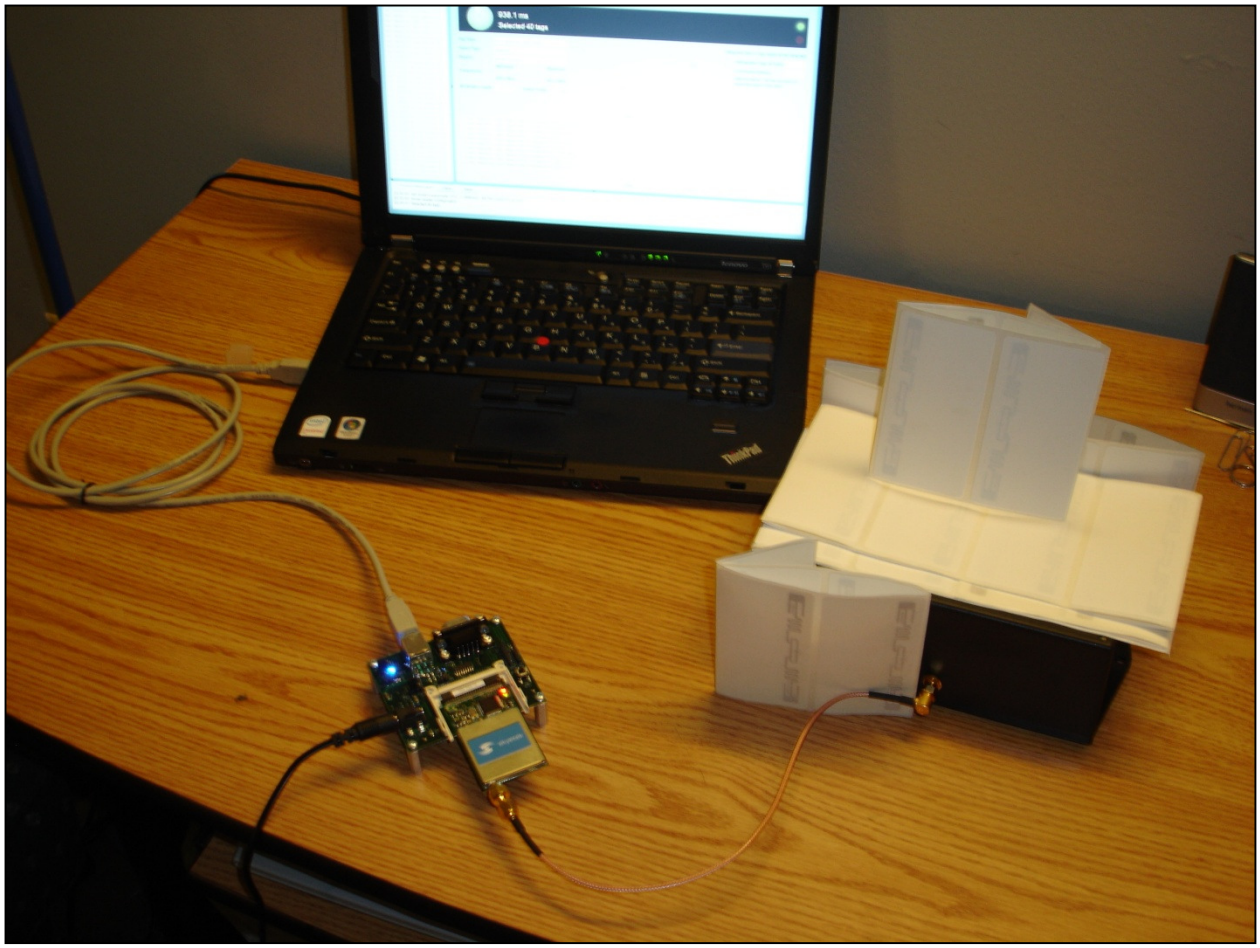


Figure 47: Alien Gen2 Squiggle RFID Tags Placement Snapshot

Figure 48 shows how the RFID testbed identifies 40 Alien Gen2 Squiggle RFID tags using the SkyeWare 4 Anti-Collision mode.

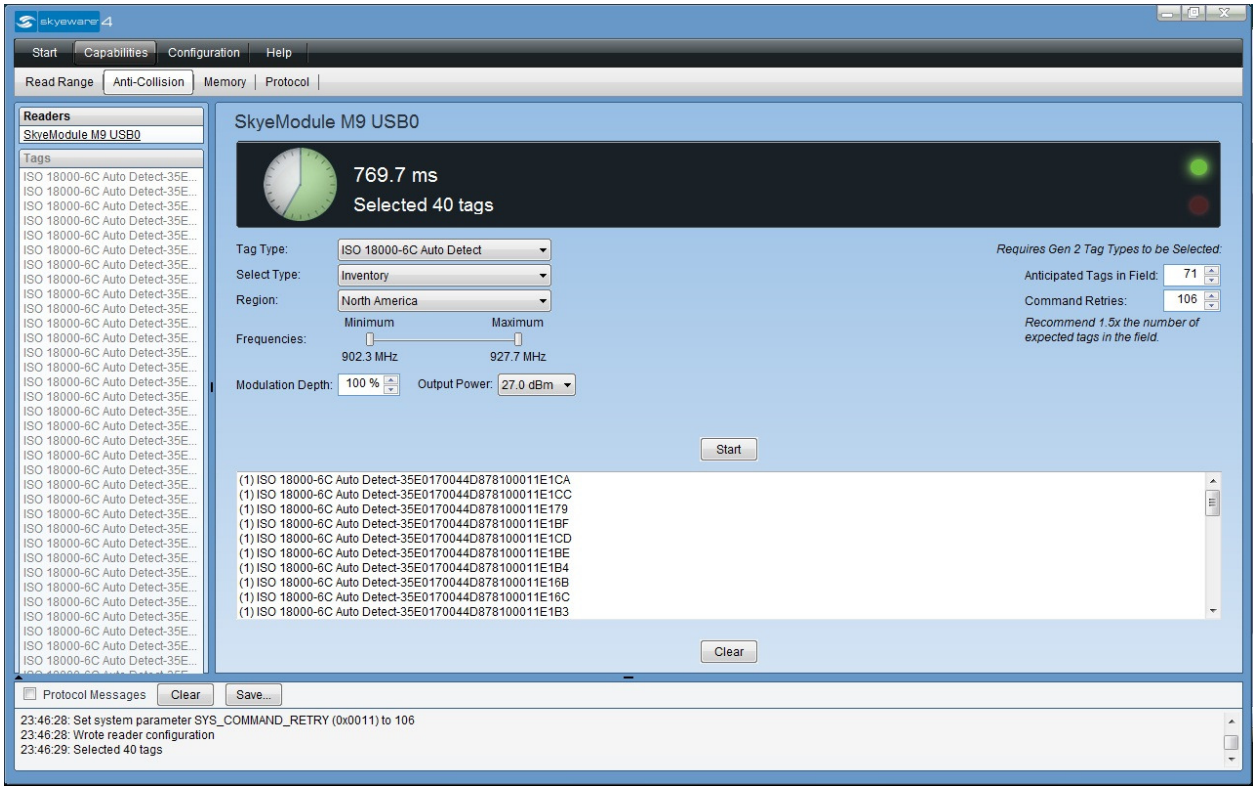


Figure 48: SkyeWare 4 Anti-Collision Mode Identifying 40 Alien Gen2 Squiggle RFID Tags

The measured RFID tag identification times for the EPC Class 1 Gen 2 protocol using the RFID testbed is shown in Table 11.

TABLE 11

RFID TESTBED TAG IDENTIFICATION TIME VALUES

Total Tags	**Delay (ms)**										
	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10	**Average**
0	64.7	64.6	66.6	64.4	65.6	64.3	64.4	65.6	68.5	64.4	65.31
10	202.5	205.6	195.6	213.2	203.4	215.8	202.9	188.5	219.0	206.2	205.27
20	327.7	329.2	324.3	340.8	328.2	318.7	339.0	362.7	328.4	329.5	332.85
30	716.1	599.5	649.5	625.3	601.2	653.6	666.6	600.9	614.3	561.1	628.81
40	767.1	847.8	769.7	909.7	787.2	783.7	782.6	767.5	847.7	869.8	813.28

5.3 Results Comparison

In this section, the RFID Simulator results of the EPC Class 1 Gen 2 protocol are compared with the RFID testbed results of the EPC Class 1 Gen 2 protocol. Table 12 shows the measured RFID Simulator tag read times with Class 1 Gen 2 when the slot time to identify a single tag reply is 8.0078123 ms, slot time to identify a collided tag reply is 1.9042968 ms, slot time for the no tag reply is 0.6103516 ms, and the measured RFID testbed tag read times with Class 1 Gen 2.

TABLE 12

SIMULATION AND EXPERIMENTAL RFID TAG READ TIMES WITH CLASS 1 GEN 2 PROTOCOL

Total Tags	**Delay (ms)**						
	Testbed	**RFID Simulator**					
		Tag Numbers Known			**Tag Numbers Unknown**		
		C = 0.2	**C = 0.3**	**C = 0.4**	**C = 0.2**	**C = 0.3**	**C = 0.4**
0	65.31	0	0	0	11.5967	7.9346	6.1035
10	205.27	96.019	96.230	96.931	100.839	98.251	97.922
20	332.85	197.469	198.289	199.864	201.719	201.245	200.872
30	628.81	302.499	303.596	305.473	307.672	305.790	306.808
40	813.28	407.228	408.258	410.70	411.091	409.204	412.115

Figure 49 shows the graphical representation of the RFID tag read times with Class 1 Gen 2 using the RFID Simulator and the RFID testbed.

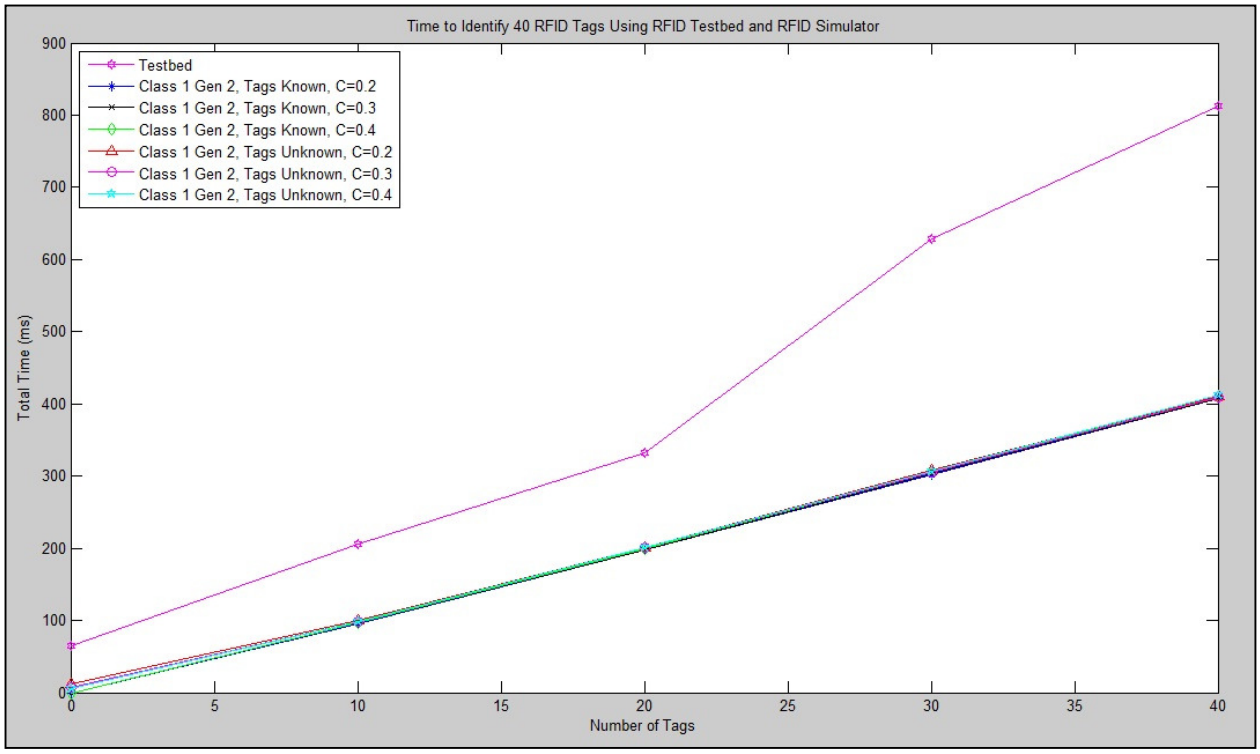


Figure 49: Time to Identify 40 RFID Tags Using RFID Testbed and RFID Simulator

According to Figure 49, the RFID Simulator takes less time to identify RFID tags than the RFID testbed for the EPCglobal Class-1 Generation-2 anti-collision protocol. There are many reasons for this. The main factor is environmental interference. The RFID Simulator does not consider interference when reading tags. Therefore, all non-collided tags will be read successfully. But with the RFID testbed, even tags that do not have any collisions sometimes will not be able to be identified due to environmental interference. Therefore, it takes more time to identify all RFID tags in the region.

Another factor is the antenna gain of the RFID testbed. A higher gain antenna provides a longer read range. However, this longer range is achieved through a smaller beam width, which in turn reduces the size of the read field, thus affecting read reliability. Consequently, more time is needed to identify all tags in the area. Antenna cable length is another cause for the greater delay in the RFID testbed. The antenna cable gain/loss is approximately -0.49 dB per meter, or

-0.15 dB per foot, for a standard RG58 coaxial cable [36]. The antenna polarization will also change the RFID testbed tag read time. A linear antenna will add range but requires that the tag is in a specific orientation to the antenna before the reader can detect the tag. A circular polarized antenna in general has less range, but it allows orienting the tag to the antenna in multiple ways. The radio frequency power of the SkyeTek SkyeModule M9 RFID reader affects the tag read as well. The radio frequency power was set to the maximum, which is 27 dBm, to permit the reader to identify all tags in the region quickly without any failures. Tag orientation (angle of tag to reader field), tag type, and manufacturer all induce performance variations. Even though SkyeTek uses EPCglobal Class-1 Generation-2 Protocol to communicate between the reader and tag, it uses SkyeTek specific SkyeTek Protocol V3 messages to communicate between the common blade interface board and PC, which in turn adds an additional time to RFID testbed tag identification. As mentioned in section 5.2, the SkyeTek specific Command Retry parameter changes the timeout value of the RFID reader, which consequently adds more delay to the RFID testbed tag read time.

Tags will not reply as soon as they receive a command from the reader with the RFID testbed. Tags take some time to change their state, encode data, and backscatter the reply. The rise time, settling time, and fall time of RFID tags also adds more delay to the RFID testbed. Prior to inventory, the RFID reader goes through a Select round in the RFID testbed, which does not occur in the RFID Simulator. The RFID Simulator does not take into account invalid ACKs and invalid RN16s sent by the RFID tags. But with the RFID testbed these invalid ACKs and invalid RN16s will occur and add more delay to tag identification time.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

This research presented the complete development of the RFID Simulator from scratch to evaluate the performance of Slotted Aloha and EPCglobal Class-1 Generation-2 protocols for RFID systems. This allows RFID system designers and researchers to simulate different environments before deployment to correctly identify various factors like the number of RFID readers needed, where to place these readers, etc. The RFID Simulator was designed to replicate a real-life RFID environment, can be used to imitate hardware, and has the capability to calculate the delay to any number of RFID tags, which is not possible with real-life RFID systems. As a result, the performance of RFID systems can be improved significantly, which in turn can generate enormous financial profits for companies. The integrity of the RFID Simulator was verified by comparing its results with mathematical analysis and experimental results. The RFID Simulator is a complete all-in-one package that was designed with the ability to be extended to a commercial RFID simulator, which will help immensely in the future development of RFID.

6.2 Future Work

Collisions during RFID communication are considered to be one of the greatest challenges in RFID development since they decrease system performance significantly. Enhanced RFID anti-collision protocols must be developed to overcome this issue. As a result, these protocols are a dynamic area of research. A complete all-in-one RFID simulator, free of charge, will be an enormous assist to the development of RFID anti-collision protocols and RFID technology. In this section, several future improvements of the RFID Simulator are presented.

One of the major improvements for the RFID Simulator would be taking into account the influence of environmental interference. Even non-collided tags could not be identified due to interference. For example, metals, wood, water, etc. will interfere with RFID signals, reducing its power and making it unrecognizable to RFID readers or tags. Another area of future work might be to add multiple RFID reader capability to the RFID Simulator to allow researchers to identify the improvement in tag identification time when there are multiple RFID readers in the area. Another improvement for the RFID Simulator would be the consideration of invalid ACKs and invalid RN16s sent by the RFID tags. Other future work might be to include reflecting the delays for antenna gain, antenna polarization, radio frequency power of the RFID reader, tag orientation, etc. in the RFID Simulator.

REFERENCES

REFERENCES

[1] Glover, B., and H. Bhatt. *RFID Essentials*. 1st ed. CA: O'Reilly, 2006.

[2] Brown, D. E. *RFID Implementation*. NY: McGraw-Hill, 2007.

[3] Hunt, V. D., A. Puglia, and M. Puglia. *RFID: A Guide to Radio Frequency Identification*. Hoboken, NJ: Wiley-Interscience, 2007.

[4] Banks, J., D. Hanny, M. A. Pachano, and L. G. Thompson. *RFID Applied*. Hoboken, NJ: John Wiley & Sons, 2007.

[5] "Regulatory status for using RFID in the UHF spectrum." *Frequency Regulations UHF*. 18 Mar 2009. EPCglobal, Web. 10 Jul 2009. <http://www.epcglobalinc.org/tech/freq_reg/RFID_at_UHF_Regulations_20090318.pdf>.

[6] Katz, D., G. Ouellette, R. Gentile, and G. Olivadoti. "Fast, Versatile Blackfin Processors Handle Advanced RFID Reader Applications." *AnalogDialogue* 40.09 (2006): 1. Web. 10 Jul 2009. <http://www.analog.com/library/analogDialogue/archives/40-09/rfid.html>.

[7] "RFID Active Tag." *RFID Components*. Web. 17 Jul 2009. <http://www.netrand.com/p_products_details04.php?proID=61>.

[8] "RFID Passive Tag." *RFID Components*. Web. 17 Jul 2009. <http://www.netrand.com/p_products_details.php?proID=62>.

[9] "MC9090-G RFID Handheld Mobile Computer." *MC9090-G RFID Handheld Mobile Computer*. Web. 17 Jul 2009. <http://www.motorola.com/Business/US-EN/Business+Product+and+Services/RFID/RFID+Readers/MC9090-G_RFID_US-EN>.

[10] Burkett, B. "RFID Software Basics." *Dynasys RFiD*. 19 Jul 2004. Dynasys Technologies, Web. 19 Jul 2009. <http://rfidusa.com/superstore/product_info.php?products_id=199>.

[11] Finkenzeller, K. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. 2nd ed. Hoboken, NJ: Wiley, 2003.

[12] White, J. B. "'Honey? Where's My Keyless Fob?' ." *THE WALL STREET JOURNAL* (2005): P7. Web. 20 Jul 2009. <http://online.wsj.com/article/SB112811826476357203.html?mod=todays_us_pursuits>.

[13] Graafstra, A. "Hands On: How radio-frequency identification and I got personal." *IEEE Spectrum* (2007): Web. 20 Jul 2009. <http://spectrum.ieee.org/computing/hardware/hands-on/1>.

REFERENCES (continued)

[14] "About ISO." *ISO*. 2009. International Organization for Standardization, Web. 25 Jul 2009. <http://www.iso.org/iso/about.htm>.

[15] de Azambuja, M. C., C. A. M. Marcon, and F. P. Hessel. "Survey of Standardized ISO 18000-6 RFID Anti-Collision Protocols." *IEEE: The Second International Conference on Sensor Technologies and Applications*. (2008): 468-73.

[16] "ISO RFID Standards: A Complete List." *Industrywizards.com*. IndustryWizards.com, Web. 25 Jul 2009. <http://rfidwizards.com/index.php?option=com_content&view=article&id=242:iso-rfid-standards-a-complete-list&catid=227:standards>.

[17] "History of the EPC and EPCglobal." *discoverrfid*. EPCglobal, Web. 2 Aug 2009. <http://www.discoverrfid.org/service/about-us/history-of-epcglobal.html>.

[18] Wyld, D. C. "RFID 101: The Next Big Thing for Management." *IEEE*. 35.2 (2007): 3.

[19] "The Labs." *AUTO-ID LABS*. Auto-ID Labs, Web. 10 Aug 2009. <http://www.autoidlabs.org/the-labs/page.html>.

[20] "Overview: What is EPCglobal?." *GS1 EPCglobal*. GS1 Member Organisations, Web. 10 Aug 2009. <http://www.epcglobalinc.org/about>.

[21] "EPCglobal Board of Governors." *GS1 EPCglobal*. GS1 Member Organisations, Web. 10 Aug 2009. <http://www.epcglobalinc.org/about/governance/>.

[22] "Standards." *GS1 EPCglobal US*. 2009. GS1 US, Web. 11 Aug 2009. <http://www.epcglobalus.org/Standards/EPCglobalStandards/tabid/185/Default.aspx>.

[23] Tanenbaum, A. S. *Computer Networks*. 4th ed. New Delhi: Prentice Hall India, 2005. 251-54.

[24] Bertsekas, D., and R. Gallager. *Data Networks*. 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1992. 277.

[25] "Vulnerable Period." *Slotted ALOHA: Principle*. Web. 15 Aug 2009. <http://www.lkn.ei.tum.de/lehre/mmprog/mac/protocols/collision/aloha2.htm>.

[26] Rahman, K. H., F. Ahmed, S. A. Sagor, and G. Mostafa. "An Efficient Anti-Collision Technique for Radio Frequency Identification Systems." *Computer and information technology*. (2007): 1-6.

REFERENCES (continued)

[27] "Throughput versus Offered Traffic for ALOHA Systems." *Chapter 4: Medium Access Control Sublayer*. Web. 21 Aug 2009. <http://homepages.ius.edu/rwisman/B438/HTML/Chapter4.htm>.

[28] De Silva, M., and K. Deegala. "A Math Model to Predict RFID Tag Read Time Using Frame Slotted Aloha Protocol." (2009): 1-4.

[29] "Specification for RFID Air Interface." *Class 1 Generation 2 UHF Air Interface Protocol Standard "Gen 2"*. 11 May 2008. GS1 EPCglobal, Web. 18 Jun 2009. <http://www.epcglobalinc.org/standards/uhfc1g2/uhfc1g2_1_2_0-standard-20080511.pdf>.

[30] Daneshmand, M., C. Wang, and K. Sohraby. "A New Slot-Count Selection Algorithm for RFID Protocol." *Communications and Networking in China*. (2007): 926-30.

[31] Wang, J., Y. Zhao, and D. Wang. "A Novel Fast Anti-Collision Algorithm for RFID Systems." *Wireless Communications, Networking and Mobile Computing*. (2007): 2044-47.

[32] "Product Overview: ALN-9640 Squiggle Inlay." *RFID TAGS*. 20 Aug 2009. Alien Technology, Web. 8 Sep 2009. <http://www.alientechnology.com/docs/products/DS_ALN_9640.pdf>.

[33] "SkyeModule M9." *SkyeTek*. 06 May 2008. SkyeTek, Inc., Web. 14 Aug 2009. <http://www.skyetek.com/Portals/0/Documents/Products/SkyeModule_M9_DataSheet.pdf >.

[34] "RFID Broadband UHF Antenna." Westminster, CO: SkyeTek, Inc., 2007.

[35] "SkyeModule M9 Reference Guide." Westminster, CO: SkyeTek, Inc., 2008.

[36] "SkyeTek Development Kit User Guide." Westminster, CO: SkyeTek, Inc., 2008.

[37] "SkyeWare 4." *SkyeTek*. SkyeTek, Inc., Web. 12 Sep 2009. <http://www.skyetek.com/ProductsServices/EmbeddedRFIDReaders/DeveloperKits/SkyeWare4/tabid/418/Default.aspx>.

APPENDICES

APPENDIX A

RFID SIMULATOR CODE

```
% ============================================================%
% Developed By : M H Maheesha H De Silva (g222m497)          %
% Assignment   : Master's Thesis                             %
% Title        : An Experimental Study of EPCglobal Class-1  %
%                Generation-2 Anti-Collision Protocol for    %
%                RFID Systems                                %
% File Names   : RFIDSimulator.fig and RFIDSimulator.m       %
% Department   : Electrical Engineering and Computer Science %
% University   : Wichita State University                    %
% ============================================================%

% NOTE: Due to the software limitations, the slot count starts from 1
% (not 0)

function varargout = RFIDSimulator(varargin)
% RFIDSimulator M-file for RFIDSimulator.fig
%      RFIDSimulator, by itself, creates a new RFIDSimulator or raises the
%      existing singleton*.
%
%      H = RFIDSimulator returns the handle to a new RFIDSimulator or the
%      handle to the existing singleton*.
%
%      RFIDSimulator('CALLBACK',hObject,eventData,handles,...) calls the
%      local function named CALLBACK in RFIDSimulator.M with the given
%      input arguments.
%
%      RFIDSimulator('Property','Value',...) creates a new RFIDSimulator or
%      raises the existing singleton*.  Starting from the left, property
%      value pairs are applied to the RFIDSimulator before
%      RFIDSimulator_OpeningFunction gets called.  Anunrecognized property
%      name or invalid value makes property applicationstop.  All inputs
%      are passed to RFIDSimulator_OpeningFcn via varargin.
%
%      *See RFIDSimulator Options on GUIDE's Tools menu.  Choose
%      "RFIDSimulator allows only one instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help RFIDSimulator

% Last Modified by GUIDE v2.5 26-Nov-2009 10:47:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @RFIDSimulator_OpeningFcn, ...
                   'gui_OutputFcn',  @RFIDSimulator_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
```

```
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --------------------------------------------------------------------- %

% Executes just before RFIDSimulator is made visible.
function RFIDSimulator_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to RFIDSimulator (see VARARGIN)

% Clear command window
clc

% Choose default command line output for RFIDSimulator
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes RFIDSimulator wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --------------------------------------------------------------------- %

function File_Menu_Callback(hObject, eventdata, handles)
% hObject    handle to FileMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------- %

function Print_Menu_Callback(hObject, eventdata, handles)
% hObject    handle to Print_Menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
printdlg(handles.figure1)

% --------------------------------------------------------------------- %
```

```
function Close_Menu_Callback(hObject, eventdata, handles)
% hObject    handle to Close_Menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
                     ['Close ' get(handles.figure1,'Name') '...'],...
                     'Yes','No','Yes');
if strcmp(selection,'No')
    return;
end

delete(handles.figure1)

% --------------------------------------------------------------------- %

% Outputs from this function are returned to the command line.
function varargout = RFIDSimulator_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --------------------------------------------------------------------- %

function Input_Callback(hObject, eventdata, handles)
% hObject    handle to Input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Input as text
%        str2double(get(hObject,'String')) returns contents of Input as
%        a double
input = str2double(get(hObject,'String'));

% Checking to see whether the input is a number. If not, assign
% input1_editText to zero and display the error message
if isnan(input)
    set(hObject, 'String', 0);
    errordlg('Input must be a number ...!','Error');
    return;
end

% Checking to see whether the input is a multiple of 10. If not, assign
% input1_editText to zero and display the error message
multiple_of_ten = mod(input,10);

if (multiple_of_ten ~=0)
    set(hObject, 'String', 0);
```

```
    errordlg('Input must be a multiple of 10 ...!','Error');

    return;
end

% Checking to see whether the input is empty. If so, assign
% input1_editText to zero
if (isempty(input))
     set(hObject,'String','0')
end

% Save the new input value
handles.data_input = round(input);
guidata(hObject, handles);

% --------------------------------------------------------------------

% Executes during object creation, after setting all properties.
function Input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --------------------------------------------------------------------

function SelectProtocol_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to unitgroup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------

% Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% If the user selects the Slotted Aloha Protocol
if (get(handles.SlottedAloha,'Value'))
```

```
% Slotted Aloha Protocol
% ----------------------

    % Clear command window
    clc

    % Declaring an array to hold the value of total tag number for each
    % scenario (0, 10, 20, 30, .... , 490, 500, 510, ....)
    total_aloha_tags = [];

    % Declaring an array to hold the total average time for each scenario
    total_aloha_average_time = [];

    % The simulation is run starting with 0 tags until total number of
    % tags specified by the user
    for scenario_aloha = 0:10:handles.data_input

        % Calling the progressbar function
        progressbar(scenario_aloha/handles.data_input)

        % Array to hold the total tags for each scenario
        total_aloha_tags = [total_aloha_tags scenario_aloha];

        % Array to hold the total time of each trial
        time_all_aloha_trials = [];

        % Each slot is assigned a time in miliseconds
        slot_time = 1;

        % No of trials for each scenario
        number_of_aloha_trials = 1000;

        % Simulation is run for "number_of_aloha_trials" trials
        for trial = 1:number_of_aloha_trials

            % Initial total time to read all RFID tags is 0
            time_aloha = 0;

            % Assigning the initial total number of tags before any
            % collisions
            number_of_tags = scenario_aloha;

            % The loop is run until all the tags are read
            while number_of_tags ~= 0

                % Randomly assigning slots for each RFID tag
                slot_number = ceil(number_of_tags*rand(1,number_of_tags));

                % Initially 0 collisions
                collisions = 0;
```

```
                % For all tags
                for i = 1:number_of_tags

                    % Initially flag is set to 0 (unset)
                    flag = 0;

                    for j = 1:number_of_tags

                        % To compare each tag slot number with other tag
                        % slot numbers
                        k = i+j;

                        if k <= number_of_tags

                            % If there is a collision and the collision
                            % was not already considered
                            % (slot number is not 0)
                            if slot_number(i) == slot_number(k) &
slot_number(i) > 0

                                % flag is set (not equal to 0)
                                flag = flag + 1;

                                % Adding 1 to existing collisions
                                collisions = collisions + 1;

                                % Making the collision already considered
                                % by making the slot number equal to 0
                                slot_number(k) = 0;
                            end
                        end
                    end

                    % If the flag is set (not equal to 0), then there is a
                    % collision
                    if flag ~= 0

                        % Adding another 1 (the base tag that used to
                        % compare) to existing collisions
                        collisions = collisions + 1;
                    end

                    % Tag is read (no collision)
                    slot_number(i) = 0;
                end

                % Calculating the time
                time_aloha = time_aloha + slot_time*number_of_tags;

                % Reading the collided tags again
                number_of_tags = collisions;
```

```
            end

            % Array to hold the total time of each trial
            time_all_aloha_trials = [time_all_aloha_trials time_aloha];
        end

        % Calculating the average time
        average_time = mean(time_all_aloha_trials);

        % Array to hold the total average times for each scenario
        total_aloha_average_time = [total_aloha_average_time average_time];
    end

    % Calculating the last array index to find the total time
    index_aloha = (handles.data_input/10)+1;

    % Displaying the total time to identify all the RFID tags
    handles.total_aloha_average_time = total_aloha_average_time(index_aloha);
    set(handles.TotalDelay, 'String', handles.total_aloha_average_time);

    % Plotting the graph
    plot(total_aloha_tags,total_aloha_average_time,'m*-')

    % Clearing previous values
    set(handles.aloha_text, 'String', ' ');
    set(handles.c1g2_tagsknown_text, 'String', ' ');
    set(handles.c1g2_tagsunknown_text, 'String', ' ');
    set(handles.TotalDelay2, 'String', ' ');
    set(handles.TotalDelay3, 'String', ' ');

    % Labeling the plot
    xlabel('Number of Tags')
    ylabel('Total Time')
    title('RFID Tag Read Time with Frame Slotted Aloha Protocol')


% If the user selects the Class1 Gen2 Protocol with Tags Known
elseif (get(handles.Class1Gen2TagsKnown,'Value'))

% Class1 Gen2 Protocol with Tags Number Known by the Reader
% ---------------------------------------------------------

    % Clear command window
    clc

    % Declaring an array to hold the value of total tag number for each
    % scenario (0, 10, 20, 30, .... , 490, 500, 510, ....)
    total_c1g2_tags = [];
```

```
% Declaring an array to hold the total average time for each scenario
total_c1g2_average_time = [];

% The simulation is run starting with 0 tags until total number of
% tags specified by the user
for scenario_c1g2 = 0:10:handles.data_input

    % Calling the progressbar function
    progressbar(scenario_c1g2/handles.data_input)

    % Array to hold the total tags for each scenario
    total_c1g2_tags = [total_c1g2_tags scenario_c1g2];

    % Array to hold the total time of each trial
    time_all_c1g2_trials = [];

    % Each slot is assigned a time in miliseconds
    slot_time_single_tag_reply = 8.0078123;
    slot_time_collided_reply = 1.9042968;
    slot_time_no_reply = 0.6103516;

    % No of trials for each scenario
    number_of_c1g2_trials = 1000;

    % Simulation is run for "number_of_c1g2_trials" trials
    for trial = 1:number_of_c1g2_trials

        % The values for c are 0.1 < c < 0.5
        c = 0.3;

        % The floating-point depiction of Q
        % Initial value of Qfp is 4
        Qfp = 4;

        % Initial value of Q is 4
        Q = 4;

        % Initial total time to read all RFID tags is 0
        query_time = 0;

        % Assigning the value for the unidentified tags
        % Initially, it is equal to total number of tags
        total_unidentified_tags = scenario_c1g2;

        % The loop is run until all the tags are read
        while (total_unidentified_tags ~= 0)

            % The Qfp value is round to the nearest integer and
            % assigned to Q
            Q = round(Qfp);
```

```
% Used to identify the number of tag responses
count = 0;

% Tags selecting the random value between 0 and 2^Q -1
% for the slot counter
slot_number = ceil(2^Q*rand(1,total_unidentified_tags));

% Checking the number of tags that selected the value 1
% randomly for slot counter
for i = 1:total_unidentified_tags
    if (slot_number(i) == 1)
        count = count + 1;
    end
end

% Only one tag has selected value 1 for slot counter
if (count == 1)

    % The reader will receive a successful reply
    % Current value of Q and Qfp will remain unchanged
    total_unidentified_tags = total_unidentified_tags - 1;

    % Calculating the time (delay)
    query_time = query_time + slot_time_single_tag_reply;

% No tags had selected value 1 for the slot counter
% Therefore, no replies to the reader from the tags
elseif (count == 0)

    % The value of Q is too large or there is a small
    % amount of remaining tags
    % Qfp is decremented by the value of c
    % Qfp will be assigned 0, if it gets a negative value
    Qfp = max(0,Qfp-c);

    % Calculating the time (delay)
    query_time = query_time + slot_time_no_reply;

% More than one tag has selected 1 for slot counter
% The replies will get collided at the reader
else

    % The value of Q is considered as too small or there
    % is lot of remaining tags
    % Qfp is incremented by the value of c
    % But still the maximum value of Qfp is 15
    Qfp = min(15,Qfp+c);

    % Calculating the time (delay)
    query_time = query_time + slot_time_collided_reply;
end
```

```
            end

            % Array to hold the total time of each trial
            time_all_c1g2_trials = [time_all_c1g2_trials query_time];
        end

        % Calculating the average time
        average_time = mean(time_all_c1g2_trials);

        % Array to hold the total average times for each scenario
        total_c1g2_average_time = [total_c1g2_average_time average_time];
    end

    % Calculating the last array index to find the total time
    index_c1g2 = (handles.data_input/10)+1;

    % Displaying the total time to identify all the RFID tags
    handles.total_c1g2_delay = total_c1g2_average_time(index_c1g2);
    set(handles.TotalDelay, 'String', handles.total_c1g2_delay);

    % Plotting the graph
    plot(total_c1g2_tags,total_c1g2_average_time,'bO-')

    % Clearing previous values
    set(handles.aloha_text, 'String', ' ');
    set(handles.c1g2_tagsknown_text, 'String', ' ');
    set(handles.c1g2_tagsunknown_text, 'String', ' ');
    set(handles.TotalDelay2, 'String', ' ');
    set(handles.TotalDelay3, 'String', ' ');

    % Labeling the plot
    xlabel('Number of Tags')
    ylabel('Total Time')
    title('RFID Tag Read Time with Class1 Gen2 Protocol (Number of Tags
Known)')


% If the user selects the Class1 Gen2 Protocol with Tags Unknown
elseif (get(handles.Class1Gen2TagsUnknown,'Value'))

% Class1 Gen2 Protocol with Tags Number Unknown by the Reader
% ------------------------------------------------------------

    % Clear command window
    clc

    % Declaring an array to hold the value of total tag number for each
    % scenario (0, 10, 20, 30, .... , 490, 500, 510, ....)
    total_c1g2_tags = [];
```

```
% Declaring an array to hold the total average time for each scenario
total_c1g2_average_time = [];

% The simulation is run starting with 0 tags until total number of
% tags specified by the user
for scenario_c1g2 = 0:10:handles.data_input

    % Calling the progressbar function
    progressbar(scenario_c1g2/handles.data_input)

    % Array to hold the total tags for each scenario
    total_c1g2_tags = [total_c1g2_tags scenario_c1g2];

    % Array to hold the total time of each trial
    time_all_c1g2_trials = [];

    % Each slot is assigned a time in miliseconds
    slot_time_single_tag_reply = 8.0078123;
    slot_time_collided_reply = 1.9042968;
    slot_time_no_reply = 0.6103516;

    % No of trials for each scenario
    number_of_c1g2_trials = 1000;

    % Simulation is run for "number_of_c1g2_trials" trials
    for trial = 1:number_of_c1g2_trials

        % The values for c are 0.1 < c < 0.5
        c = 0.3;

        % The floating-point depiction of Q
        % Initial value of Qfp is 4
        Qfp = 4;

        % Initial value of Q is 4
        Q = 4;

        % Initial total time to read all RFID tags is 0
        query_time = 0;

        % Assigning the value for the total tags
        total_tags = scenario_c1g2;

        % The loop is run until Q is equal to 0
        % That is, no unidentified tags present (all tags are read)
        while (Q ~= 0)

            % The Qfp value is round to the nearest integer and
            % assigned to Q
            Q = round(Qfp);
```

```
% Used to identify the number of tag responses
count = 0;

% Tags selecting the random value between 0 and 2^Q -1
% for the slot counter
slot_number = ceil(2^Q*rand(1,total_tags));

% Checking the number of tags that selected the value 1
% randomly for slot counter
for i = 1:total_tags
    if (slot_number(i) == 1)
        count = count + 1;
    end
end

% Only one tag has selected value 1 for slot counter
if (count == 1)

    % The reader will receive a successful reply
    % Current value of Q and Qfp will remain unchanged
    total_tags = total_tags - 1;

    % Calculating the time (delay)
    query_time = query_time + slot_time_single_tag_reply;

% No tags had selected value 1 for the slot counter
% Therefore, no replies to the reader from the tags
elseif (count == 0)

    % The value of Q is too large or there is a small
    % amount of remaining tags
    % Qfp is decremented by the value of c
    % Qfp will be assigned 0, if it gets a negative value
    Qfp = max(0,Qfp-c);

    % Calculating the time (delay)
    query_time = query_time + slot_time_no_reply;

% More than one tag has selected 1 for slot counter
% The replies will get collided at the reader
else

    % The value of Q is considered as too small or there
    % is lot of remaining tags
    % Qfp is incremented by the value of c
    % But still the maximum value of Qfp is 15
    Qfp = min(15,Qfp+c);

    % Calculating the time (delay)
    query_time = query_time + slot_time_collided_reply;
end
```

```
            end

            % Array to hold the total time of each trial
            time_all_c1g2_trials = [time_all_c1g2_trials query_time];
        end

        % Calculating the average time
        average_time = mean(time_all_c1g2_trials);

        % Array to hold the total average times for each scenario
        total_c1g2_average_time = [total_c1g2_average_time average_time];
    end

    % Calculating the last array index to find the total time
    index_c1g2 = (handles.data_input/10)+1;

    % Displaying the total time to identify all the RFID tags
    handles.total_c1g2_delay = total_c1g2_average_time(index_c1g2);
    set(handles.TotalDelay, 'String', handles.total_c1g2_delay);

    % Plotting the graph
    plot(total_c1g2_tags,total_c1g2_average_time,'rp-')

    % Clearing previous values
    set(handles.aloha_text, 'String', ' ');
    set(handles.c1g2_tagsknown_text, 'String', ' ');
    set(handles.c1g2_tagsunknown_text, 'String', ' ');
    set(handles.TotalDelay2, 'String', ' ');
    set(handles.TotalDelay3, 'String', ' ');

    % Labeling the plot
    xlabel('Number of Tags')
    ylabel('Total Time')
    title('RFID Tag Read Time with Class1 Gen2 Protocol (Number of Tags
Unknown)')


% If the user selects all protocols
elseif (get(handles.All,'Value'))
    % Clear command window
    clc

% Slotted Aloha Protocol
% ----------------------

    % Declaring an array to hold the value of total tag number for each
    % scenario (0, 10, 20, 30, .... , 490, 500, 510, ....)
    total_aloha_tags = [];

    % Declaring an array to hold the total average time for each scenario
    total_aloha_average_time = [];
```

```
% The simulation is run starting with 0 tags until total number of
% tags specified by the user
for scenario_aloha = 0:10:handles.data_input

    % Array to hold the total tags for each scenario
    total_aloha_tags = [total_aloha_tags scenario_aloha];

    % Array to hold the total time of each trial
    time_all_aloha_trials = [];

    % Each slot is assigned a time in miliseconds
    slot_time = 1;

    % No of trials for each scenario
    number_of_aloha_trials = 1000;

    % Simulation is run for "number_of_aloha_trials" trials
    for trial = 1:number_of_aloha_trials

        % Initial total time to read all RFID tags is 0
        time_aloha = 0;

        % Assigning the initial total number of tags before any
        % collisions
        number_of_tags = scenario_aloha;

        % The loop is run until all the tags are read
        while number_of_tags ~= 0

            % Randomly assigning slots for each RFID tag
            slot_number = ceil(number_of_tags*rand(1,number_of_tags));

            % Initially 0 collisions
            collisions = 0;

            % For all tags
            for i = 1:number_of_tags

                % Initially flag is set to 0 (unset)
                flag = 0;

                for j = 1:number_of_tags

                    % To compare each tag slot number with other tag
                    % slot numbers
                    k = i+j;

                    if k <= number_of_tags
```

```
                            % If there is a collision and the collision
                            % was not already considered
                            % (slot number is not 0)
                            if slot_number(i) == slot_number(k) &
slot_number(i) > 0

                                % flag is set (not equal to 0)
                                flag = flag + 1;

                                % Adding 1 to existing collisions
                                collisions = collisions + 1;

                                % Making the collision already considered
                                % by making the slot number equal to 0
                                slot_number(k) = 0;
                            end
                        end
                    end

                    % If the flag is set (not equal to 0), then there is a
                    % collision
                    if flag ~= 0

                        % Adding another 1 (the base tag that used to
                        % compare) to existing collisions
                        collisions = collisions + 1;
                    end

                    % Tag is read (no collision)
                    slot_number(i) = 0;
                end

                % Calculating the time
                time_aloha = time_aloha + slot_time*number_of_tags;

                % Reading the collided tags again
                number_of_tags = collisions;
            end

            % Array to hold the total time of each trial
            time_all_aloha_trials = [time_all_aloha_trials time_aloha];
        end

        % Calculating the average time
        average_time = mean(time_all_aloha_trials);

        % Array to hold the total average times for each scenario
        total_aloha_average_time = [total_aloha_average_time average_time];
    end
```

```
    % Calculating the last array index to find the total time
    index_aloha = (handles.data_input/10)+1;

    % Displaying the total time to identify all the RFID tags
    handles.total_aloha_average_time = total_aloha_average_time(index_aloha);
    set(handles.TotalDelay, 'String', handles.total_aloha_average_time);


% Class1 Gen2 Protocol with Tags Number Known by the Reader
% ---------------------------------------------------------

    % Declaring an array to hold the value of total tag number for each
    % scenario (0, 10, 20, 30, .... , 490, 500, 510, ....)
    total_c1g2_tags = [];

    % Declaring an array to hold the total average time for each scenario
    total_c1g2_average_time = [];

    % The simulation is run starting with 0 tags until total number of
    % tags specified by the user
    for scenario_c1g2 = 0:10:handles.data_input

        % Calling the progressbar function
        progressbar(scenario_c1g2/handles.data_input)

        % Array to hold the total tags for each scenario
        total_c1g2_tags = [total_c1g2_tags scenario_c1g2];

        % Array to hold the total time of each trial
        time_all_c1g2_trials = [];

        % Each slot is assigned a time in miliseconds
        slot_time_single_tag_reply = 8.0078123;
        slot_time_collided_reply = 1.9042968;
        slot_time_no_reply = 0.6103516;

        % No of trials for each scenario
        number_of_c1g2_trials = 1000;

        % Simulation is run for "number_of_c1g2_trials" trials
        for trial = 1:number_of_c1g2_trials

            % The values for c are 0.1 < c < 0.5
            c = 0.3;

            % The floating-point depiction of Q
            % Initial value of Qfp is 4
            Qfp = 4;
```

```
% Initial value of Q is 4
Q = 4;

% Initial total time to read all RFID tags is 0
query_time = 0;

% Assigning the value for the unidentified tags
% Initially, it is equal to total number of tags
total_unidentified_tags = scenario_c1g2;

% The loop is run until all the tags are read
while (total_unidentified_tags ~= 0)

    % The Qfp value is round to the nearest integer and
    % assigned to Q
    Q = round(Qfp);

    % Used to identify the number of tag responses
    count = 0;

    % Tags selecting the random value between 0 and 2^Q -1
    % for the slot counter
    slot_number = ceil(2^Q*rand(1,total_unidentified_tags));

    % Checking the number of tags that selected the value 1
    % randomly for slot counter
    for i = 1:total_unidentified_tags
        if (slot_number(i) == 1)
            count = count + 1;
        end
    end

    % Only one tag has selected value 1 for slot counter
    if (count == 1)

        % The reader will receive a successful reply
        % Current value of Q and Qfp will remain unchanged
        total_unidentified_tags = total_unidentified_tags - 1;

        % Calculating the time (delay)
        query_time = query_time + slot_time_single_tag_reply;

    % No tags had selected value 1 for the slot counter
    % Therefore, no replies to the reader from the tags
    elseif (count == 0)

        % The value of Q is too large or there is a small
        % amount of remaining tags
        % Qfp is decremented by the value of c
        % Qfp will be assigned 0, if it gets a negative value
        Qfp = max(0,Qfp-c);
```

```
                    % Calculating the time (delay)
                    query_time = query_time + slot_time_no_reply;

                % More than one tag has selected 1 for slot counter
                % The replies will get collided at the reader
                else

                    % The value of Q is considered as too small or there
                    % is lot of remaining tags
                    % Qfp is incremented by the value of c
                    % But still the maximum value of Qfp is 15
                    Qfp = min(15,Qfp+c);

                    % Calculating the time (delay)
                    query_time = query_time + slot_time_collided_reply;
                end
            end

            % Array to hold the total time of each trial
            time_all_c1g2_trials = [time_all_c1g2_trials query_time];
        end

        % Calculating the average time
        average_time = mean(time_all_c1g2_trials);

        % Array to hold the total average times for each scenario
        total_c1g2_average_time = [total_c1g2_average_time average_time];
    end

    % Calculating the last array index to find the total time
    index_c1g2 = (handles.data_input/10)+1;

    % Displaying the total time to identify all the RFID tags
    handles.total_c1g2_delay = total_c1g2_average_time(index_c1g2);
    set(handles.TotalDelay2, 'String', handles.total_c1g2_delay);


% Class1 Gen2 Protocol with Tags Number Unknown by the Reader
% -----------------------------------------------------------

    % Declaring an array to hold the value of total tag number for each
    % scenario (0, 10, 20, 30, .... , 490, 500, 510, ....)
    total_c1g2_tags_unknown = [];

    % Declaring an array to hold the total average time for each scenario
    total_c1g2_average_time_unknown = [];

    % The simulation is run starting with 0 tags until total number of
    % tags specified by the user
    for scenario_c1g2 = 0:10:handles.data_input
```

```
% Array to hold the total tags for each scenario
total_c1g2_tags_unknown = [total_c1g2_tags_unknown scenario_c1g2];

% Array to hold the total time of each trial
time_all_c1g2_trials = [];

% Each slot is assigned a time in miliseconds
slot_time_single_tag_reply = 8.0078123;
slot_time_collided_reply = 1.9042968;
slot_time_no_reply = 0.6103516;

% No of trials for each scenario
number_of_c1g2_trials = 1000;

% Simulation is run for "number_of_c1g2_trials" trials
for trial = 1:number_of_c1g2_trials

    % The values for c are 0.1 < c < 0.5
    c = 0.3;

    % The floating-point depiction of Q
    % Initial value of Qfp is 4
    Qfp = 4;

    % Initial value of Q is 4
    Q = 4;

    % Initial total time to read all RFID tags is 0
    query_time = 0;

    % Assigning the value for the total tags
    total_tags = scenario_c1g2;

    % The loop is run until Q is equal to 0
    % That is, no unidentified tags present (all tags are read)
    while (Q ~= 0)

        % The Qfp value is round to the nearest integer and
        % assigned to Q
        Q = round(Qfp);

        % Used to identify the number of tag responses
        count = 0;

        % Tags selecting the random value between 0 and 2^Q -1
        % for the slot counter
        slot_number = ceil(2^Q*rand(1,total_tags));
```

```
        % Checking the number of tags that selected the value 1
        % randomly for slot counter
        for i = 1:total_tags
            if (slot_number(i) == 1)
                count = count + 1;
            end
        end

        % Only one tag has selected value 1 for slot counter
        if (count == 1)

            % The reader will receive a successful reply
            % Current value of Q and Qfp will remain unchanged
            total_tags = total_tags - 1;

            % Calculating the time (delay)
            query_time = query_time + slot_time_single_tag_reply;

        % No tags had selected value 1 for the slot counter
        % Therefore, no replies to the reader from the tags
        elseif (count == 0)

            % The value of Q is too large or there is a small
            % amount of remaining tags
            % Qfp is decremented by the value of c
            % Qfp will be assigned 0, if it gets a negative value
            Qfp = max(0,Qfp-c);

            % Calculating the time (delay)
            query_time = query_time + slot_time_no_reply;

        % More than one tag has selected 1 for slot counter
        % The replies will get collided at the reader
        else

            % The value of Q is considered as too small or there
            % is lot of remaining tags
            % Qfp is incremented by the value of c
            % But still the maximum value of Qfp is 15
            Qfp = min(15,Qfp+c);

            % Calculating the time (delay)
            query_time = query_time + slot_time_collided_reply;
        end
    end

    % Array to hold the total time of each trial
    time_all_c1g2_trials = [time_all_c1g2_trials query_time];
end
```

```
        % Calculating the average time
        average_time = mean(time_all_c1g2_trials);

        % Array to hold the total average times for each scenario
        total_c1g2_average_time_unknown = [total_c1g2_average_time_unknown
average_time];
    end

    % Calculating the last array index to find the total time
    index_c1g2_tagsunknown = (handles.data_input/10)+1;

    % Displaying the total time to identify all the RFID tags
    handles.total_c1g2_unknown_delay =
total_c1g2_average_time_unknown(index_c1g2_tagsunknown);
    set(handles.TotalDelay3, 'String', handles.total_c1g2_unknown_delay);

    % Labels to distinguish delays
    set(handles.aloha_text, 'String', 'Slotted Aloha');
    set(handles.c1g2_tagsknown_text, 'String', 'Class1 Gen2 (Tags Known)');
    set(handles.c1g2_tagsunknown_text, 'String', 'Class1 Gen2 (Tags
Unknown)');

    % Plotting the graphs
    plot(total_aloha_tags,total_aloha_average_time,'m*-
',total_c1g2_tags,total_c1g2_average_time,'bO-
',total_c1g2_tags_unknown,total_c1g2_average_time_unknown,'rp-')

    % Labeling the plot
    xlabel('Number of Tags')
    ylabel('Total Time')
    title('RFID Tag Read Time with Slotted Aloha and Class1 Gen2 Protocols')
    legend('Slotted Aloha','Class1 Gen2 (Tags Known)','Class1 Gen2 (Tags
Unknown)',2);
end

% --------------------------------------------------------------------- %

% Executes on button press in SlottedAloha.
function SlottedAloha_Callback(hObject, eventdata, handles)
% hObject    handle to SlottedAloha (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of SlottedAloha

% --------------------------------------------------------------------- %

% Executes on button press in Class1Gen2TagsKnown.
function Class1Gen2TagsKnown_Callback(hObject, eventdata, handles)
% hObject    handle to Class1Gen2TagsKnown (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Class1Gen2TagsKnown

% ------------------------------------------------------------------- %

% Executes on button press in All.
function All_Callback(hObject, eventdata, handles)
% hObject    handle to All (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of All

% ------------------------------------------------------------------- %

function TotalDelay_Callback(hObject, eventdata, handles)
% hObject    handle to TotalDelay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of TotalDelay as text
%        str2double(get(hObject,'String')) returns contents of TotalDelay
%        as a double

% ------------------------------------------------------------------- %

% Executes during object creation, after setting all properties.
function TotalDelay_CreateFcn(hObject, eventdata, handles)
% hObject    handle to TotalDelay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% ------------------------------------------------------------------- %

% Executes on button press in Reset.
function Reset_Callback(hObject, eventdata, handles)
% hObject    handle to Reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
initialize_gui(gcbf, handles, true);

% ------------------------------------------------------------------- %
```

```
function initialize_gui(fig_handle, handles, isreset)
handles.data_input = 0;

% Setting values to 0
set(handles.Input, 'String', handles.data_input);
set(handles.TotalDelay,  'String', handles.data_input);

% Clearing values
set(handles.aloha_text, 'String', ' ');
set(handles.c1g2_tagsknown_text, 'String', ' ');
set(handles.c1g2_tagsunknown_text, 'String', ' ');
set(handles.TotalDelay2, 'String', ' ');
set(handles.TotalDelay3, 'String', ' ');

% Selecting Slotted Aloha option as the default
set(handles.SelectProtocol, 'SelectedObject', handles.SlottedAloha);

% Clearing the plot
axes(handles.axes1);
cla;
xlabel(' ')
ylabel(' ')
title(' ')

% Update handles structure
guidata(handles.figure1, handles);

% ------------------------------------------------------------------- %

% Progress Bar function
function progressbar(fractiondone, position)
% Description:
%   progressbar(fractiondone,position) provides an indication of the
% progress of some task using graphics and text. Calling progressbar
% repeatedly will update the figure and automatically estimate the amount
% of time remaining.

persistent progfig progpatch starttime lastupdate

% Set defaults for variables not passed in
if nargin < 1
    fractiondone = 0;
end

if nargin < 2
    position = 0;
end
```

```
try

    % Access progfig to see if it exists ('try' will fail if it doesn't)
    dummy = get(progfig,'UserData');

    % If progress bar needs to be reset, close figure and set handle to
    % empty
    if fractiondone == 0

        % Close progress bar
        delete(progfig)

        % Set to empty so a new progress bar is created
        progfig = [];
    end

catch

    % Set to empty so a new progress bar is created
    progfig = [];
end

% If task completed, close figure and clear vars, then exit
percentdone = floor(100*fractiondone);

% Task completed
if percentdone == 100

    % Close progress bar
    delete(progfig)

    % Clear persistent vars
    clear progfig progpatch starttime lastupdate
    return
end

% Create new progress bar if needed
if isempty(progfig)

    % Calculate position of progress bar in normalized units
    scrsz = [0 0 1 1];
    width = scrsz(3)/4;
    height = scrsz(4)/50;

    if (length(position) == 1)

        % Padding from left or right edge of screen
        hpad = scrsz(3)/64;
```

```
    % Padding from top or bottom edge of screen
    vpad = scrsz(4)/24;

    % Default
    left   = scrsz(3)/2 - width/2;

     % Default
    bottom = scrsz(4)/2 - height/2;

    switch position

         % Center
        case 0
            % Do nothing (default)

        % Top-right
        case 1
            left   = scrsz(3) - width  - hpad;
            bottom = scrsz(4) - height - vpad;

        % Top-left
        case 2
            left   = hpad;
            bottom = scrsz(4) - height - vpad;

        % Bottom-left
        case 3
            left   = hpad;
            bottom = vpad;

        % Bottom-right
        case 4
            left   = scrsz(3) - width  - hpad;
            bottom = vpad;

        % Random
        case 5
            left   = rand * (scrsz(3)-width);
            bottom = rand * (scrsz(4)-height);

        otherwise
            warning('position must be (0-5). Reset to 0.')
    end

    position = [left bottom];

elseif length(position) == 2
```

```
        % Error checking on position
        if (position(1) < 0) | (scrsz(3)-width < position(1))
            position(1) = max(min(position(1),scrsz(3)-width),0);
            warning('Horizontal position adjusted to fit on screen.')
        end

        if (position(2) < 0) | (scrsz(4)-height < position(2))
            position(2) = max(min(position(2),scrsz(4)-height),0);
            warning('Vertical position adjusted to fit on screen.')
        end

    else
        error('position is not formatted correctly')
    end

    % Initialize progress bar
    progfig = figure(...
        'Units',            'normalized',...
        'Position',         [position width height],...
        'NumberTitle',      'off',...
        'Resize',           'off',...
        'MenuBar',          'none',...
        'BackingStore',     'off' );
    progaxes = axes(...
        'Position',         [0.02 0.15 0.96 0.70],...
        'XLim',             [0 1],...
        'YLim',             [0 1],...
        'Box',              'on',...
        'ytick',            [],...
        'xtick',            [] );
    progpatch = patch(...
        'XData',            [0 0 0 0],...
        'YData',            [0 0 1 1],...
        'EraseMode',        'none' );
    set(progfig,  'ButtonDownFcn',{@changecolor,progpatch});
    set(progaxes, 'ButtonDownFcn',{@changecolor,progpatch});
    set(progpatch,'ButtonDownFcn',{@changecolor,progpatch});
    changecolor(0,0,progpatch)

    % Set time of last update to ensure a redraw
    lastupdate = clock - 1;

    % Task starting time reference
    if isempty(starttime) | (fractiondone == 0)
        starttime = clock;
    end

end
```

```
% Enforce a minimum time interval between updates
if etime(clock,lastupdate) < 0.01
    return
end

% Update progress patch
set(progpatch,'XData',[0 fractiondone fractiondone 0])

% Update progress figure title bar
if (fractiondone == 0)
    titlebarstr = ' 0%';
else
    runtime = etime(clock,starttime);
    timeleft = runtime/fractiondone - runtime;
    timeleftstr = sec2timestr(timeleft);
    titlebarstr = sprintf('%2d%%    %s remaining',percentdone,timeleftstr);
end
set(progfig,'Name',titlebarstr)

% Force redraw to show changes
drawnow

% Record time of this update
lastupdate = clock;

% ------------------------------------------------------------------------ %

% Change the color of the progress bar patch
function changecolor(h,e,progpatch)

% Must be <= 3.0 - This keeps the color from being too light
colorlim = 2.8;
thiscolor = rand(1,3);

while sum(thiscolor) > colorlim
    thiscolor = rand(1,3);
end
set(progpatch,'FaceColor',thiscolor);

% ------------------------------------------------------------------------ %

% Convert a time measurement from seconds into a human readable string.
function timestr = sec2timestr(sec)

% Convert seconds to other units

% Days
d = floor(sec/86400);
sec = sec - d*86400;
```

```
% Hours
h = floor(sec/3600);
sec = sec - h*3600;

% Minutes
m = floor(sec/60);
sec = sec - m*60;

% Seconds
s = floor(sec);

% Create time string
if d > 0
    if d > 9
        timestr = sprintf('%d day',d);
    else
        timestr = sprintf('%d day, %d hr',d,h);
    end

elseif h > 0
    if h > 9
        timestr = sprintf('%d hr',h);
    else
        timestr = sprintf('%d hr, %d min',h,m);
    end

elseif m > 0
    if m > 9
        timestr = sprintf('%d min',m);
    else
        timestr = sprintf('%d min, %d sec',m,s);
    end

else
    timestr = sprintf('%d sec',s);
end

%======================================================================%
```

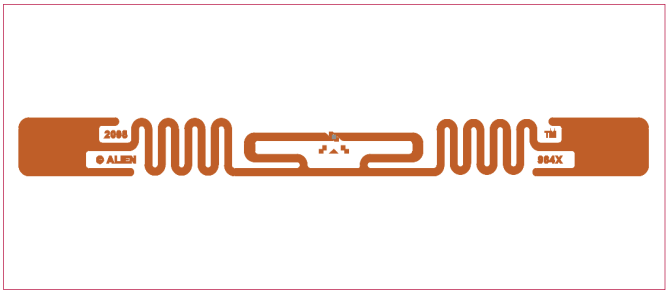
APPENDIX B

ALIEN GEN2 SQUIGGLE RFID TAG DATASHEET

Product Overview

ALN-9640 Squiggle® Inlay

The Alien Technology® ALN-9640 Squiggle® is a high performance general purpose RFID inlay for use in a wide variety of applications



Powered by Alien®'s break-through Higgs™3 UHF RFID IC and innovative Squiggle antenna design, the ALN-9640 delivers industry leading EPC Gen 2 performance and reliability at competitive prices.

ALN-9640 inlays are World Tag compliant, enabling consistent operation across the diverse frequencies of the Americas, Europe, Middle East, Asia, and Africa.

With its Higgs-3 core, the Squiggle delivers unprecedented performance and a rich feature set including a 32-bit TID, a 64-bit Unique TID for authentication and serialization applications, an extensible EPC memory bank, 512-bits of user memory for distributed data applications, and password protected read and write support capabilities to prevent unauthorized viewing and modification of the tag's data.

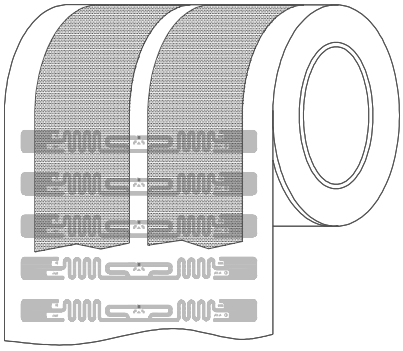
Typical applications for the Squiggle include, but are not limited to, corrugate cases, pallet placards, apparel hang tags, baggage tags, shipping labels, asset management, and file folder labels.

FEATURES

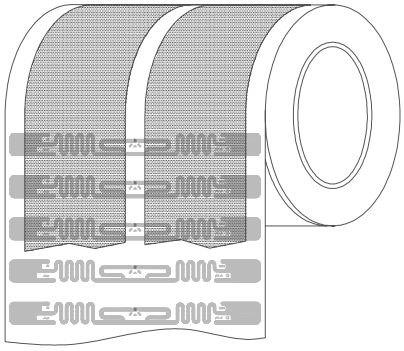
- Exceptional performance
- EPC Gen 2 (v1.2.0) compliant
- ISO/IEC 18000-6C compliant
- Worldwide RFID UHF operation (840-960MHz)
- Higgs™-3 IC with 800-bits of Nonvolatile Memory
  - 32-bit TID
  - 64-bit Unique TID
  - 96-bit EPC Memory, extensible to 480-bits
  - 512-bit User Memory
  - 32-bit Access password
  - 32-bit Kill password
- Pre-programmed with a unique, unalterable 64-bit serial number (ideal for authentication)
- User Memory can be Block Perma-Locked
- User Memory can be Read Password protected in 64-bit blocks, prohibiting unintended Reads without an access password
- Supports all Mandatory and Optional Gen 2 commands including item level commands
- Custom commands for high speed programming
- Available in high-yield, high-capacity dry/wet inlay rolls for high volume converting processes

ALN-9640 Squiggle® Inlay

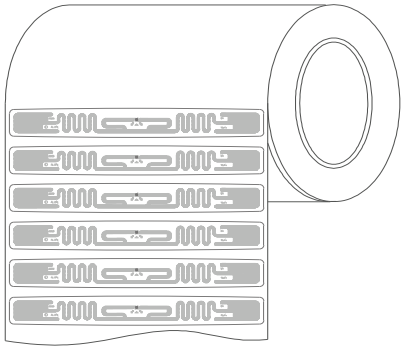
ALN-9640 Inlay Orientation



ALN-9640-FR
(Dry Unslit Roll)



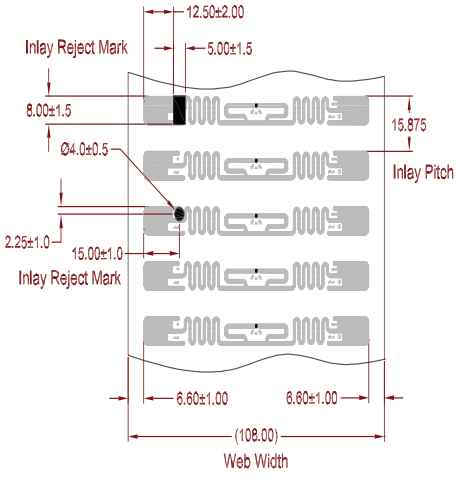
ALN-9640-FSR
(Dry Slit Roll)



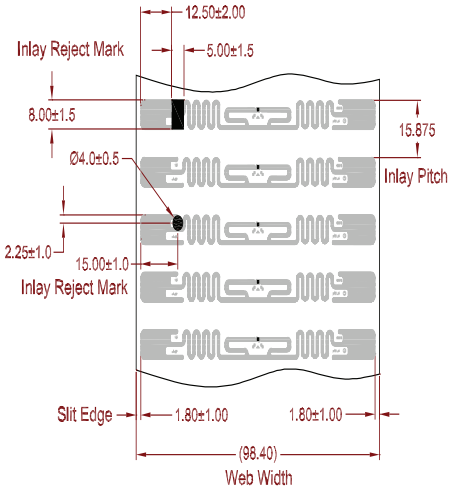
ALN-9640-FWRW
(White Wet Inlay)

Standard Alien Inlay rolls unwind with metal antenna side facing outward with respect to the core.

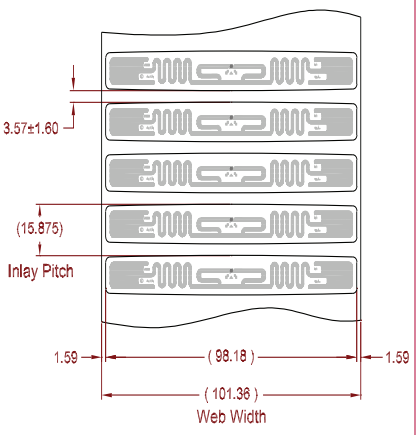
ALN-9640 Inlay Specifications



ALN-9640-FR
(Dry Unslit Roll)

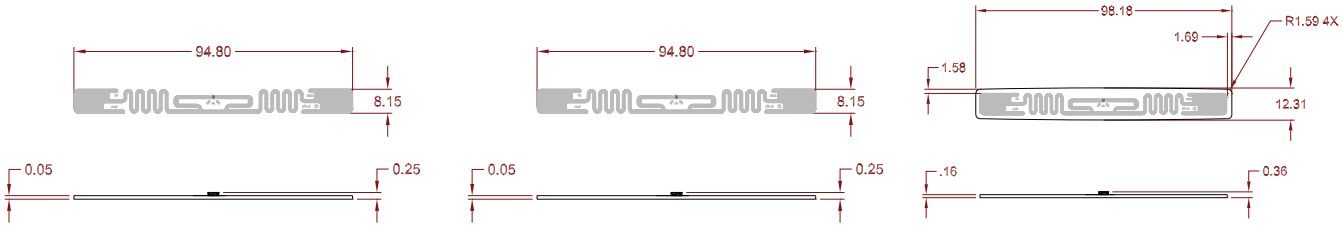


ALN-9640-FSR
(Dry Slit Roll)



ALN-9640-FWRW
(White Wet Inlay)

ALN-9640 Inlay General Dimensions



ALN-9640-FR
(Dry Unslit Inlay)

ALN-9640-FSR
(Dry Slit Inlay)

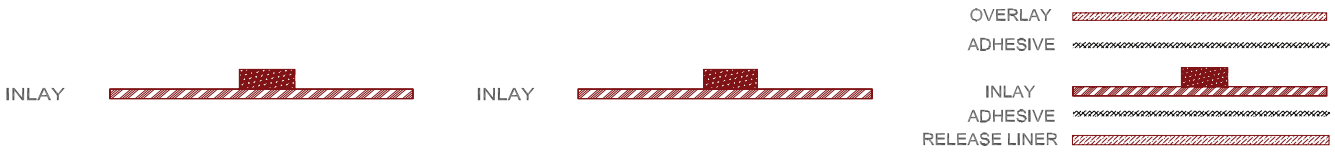
ALN-9640-FWRW
(White Wet Inlay)

ALN-9640 Inlay Stackup

DRY INLAY THICKNESS, ±10%	
OVER ANTENNA	0.05 mm
OVER CHIP	0.25 mm

DRY INLAY THICKNESS, ±10%	
OVER ANTENNA	0.05 mm
OVER CHIP	0.25 mm

WHITE WET INLAY THICKNESS, ±10%	
OVER ANTENNA	0.16 mm
OVER CHIP	0.36 mm

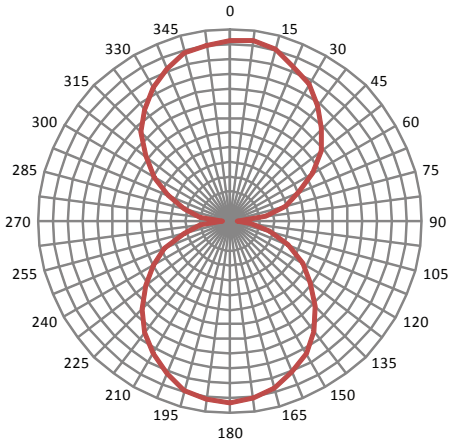


ALN-9640-FR
(Dry Unslit Inlay)

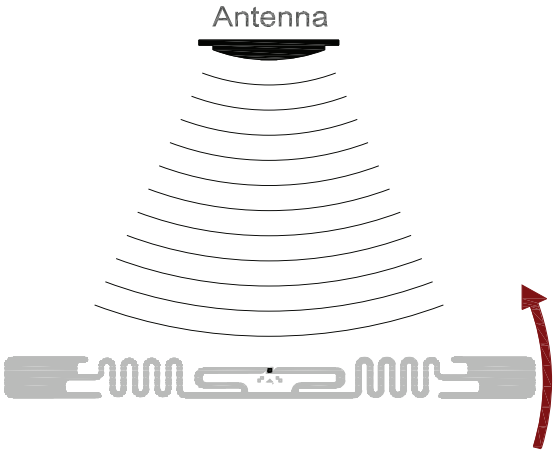
ALN-9640-FSR
(Dry Slit Inlay)

ALN-9640-FWRW
(White Wet Inlay)

ALN-9640 Inlay Angular Sensitivity



Angular Sensitivity
(Relative Read Range vs. Orientation)



Angular Sensitivity
Inlay is rotated in the x, y, plane about the z axis
(tag shown at 0° with respect to face of antenna)

ALN-9640 Squiggle® Inlay

ALN-9640 Specifications

Dry Inlay	
Antenna Width	3.732" [94.8mm]
Antenna Length	0.319" [8.1mm]
Web Width (-FR)	4.252" [108.0mm]
Web Width (-FSR)	3.874" [98.4mm]
Web Pitch	0.625" [15.875mm]
Core Width	3.874" [98.4mm]
Core ID	6" [152.4mm]
Core Material	Fiberboard
Interleaf Material	Paper
Interleaf Width	1.5" [38.1mm]
Inlays per Roll	20,000 Nominal
Maximum Roll OD	< 16" [406.4mm]
Roll Labeling Data	Roll #, Quantity

Wet Inlay	
Inlay Width	3.866" [98.2mm]
Inlay Length	0.484" [12.3mm]
Web Width (-FWR)	3.992" [101.4mm]
Web Pitch	0.625" [15.875mm]
Core Width	4.752" [120.7mm]
Core ID	6" [152.4mm]
Core Material	Fiberboard
Inlays per Roll	20,000 Nominal
Maximum Roll OD	< 16" [406.4mm]
Roll Labeling Data	Roll #, Quantity
Overlay (-FWRW)	2.5 Mil White Polyolefin
Overlay Adhesive (-FWRW)	Acrylic
Inlay Adhesive	Acrylic
Adhesive Application Temperature	> +36.5°F [+2°C]
Adhesive Service Temperature	-4°F to +199.4°F [-20°C to +93°C]
Release Liner	40# SCK

Environmental	
Shelf Life	2 Years at +77°F [+25°C] @ 40%RH
Recommended Storage	+77°F [+25°C] @ 40% RH
Storage Limits	-13°F to 122°F [-25°C to +50°C] 20% to 90% RH Non-condensing
Operating Limits	-40°F to +158°F [-40°C to +70°C] 20% to 90% RH Non-condensing
Bend Diameter	> 1.97" [50mm]
Pressure	< 5N/mm^2
Drop Resistance	Per ASTM D5276
Write Cycles	10,000
RoHs	2002/95/EC Compliant
ESD – HBM / CDM	> 5.0kV / > 1.5kV

RFID	
ISO/IEC 18000-6C	
EPCglobal Class 1 Gen 2	
Integrated Circuit	Alien Higgs-3
EPCglobal Certificate	950110126000001084
Operating Frequency	840–960 MHz
EPC Size	96 - 480 Bits
User Memory	512 Bits
TID	32 Bits
Unique TID	64 Bits
Access Password	32 Bits
Kill Password	32 Bits

HANDLING PRECAUTIONS Observe standard handling practices to minimize ESD.

DISCLAIMER Application recommendations are guidelines only - actual results may vary and should be confirmed. This is a general purpose product not designed or intended for any specific application.

ALIEN

Alien Technology
18220 Butterfield Blvd.
Morgan Hill, CA 95037
866-RFID NOW
www.alientechnology.com

20 August 2009

APPENDIX C

EXTERNAL RFID BROADBAND UHF ANTENNA DATASHEET

SkyeTek, Inc.
11030 Circle Point Rd., ste. 300
Westminster, CO 80020
U.S.A.

RFID Broadband UHF Antenna

Specifications:

Product Code:

SP-AN-04-UF-BB6LP

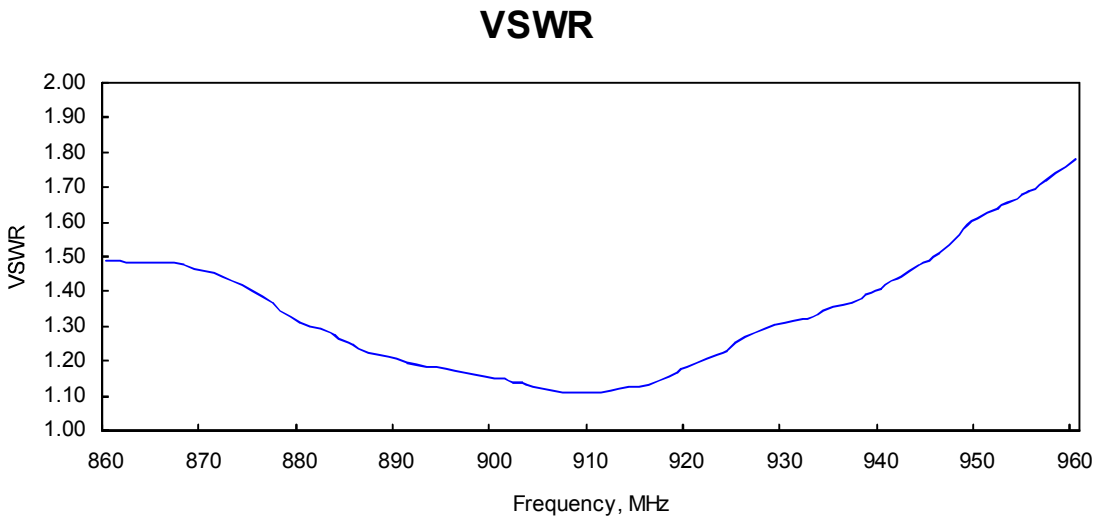
Electrical:

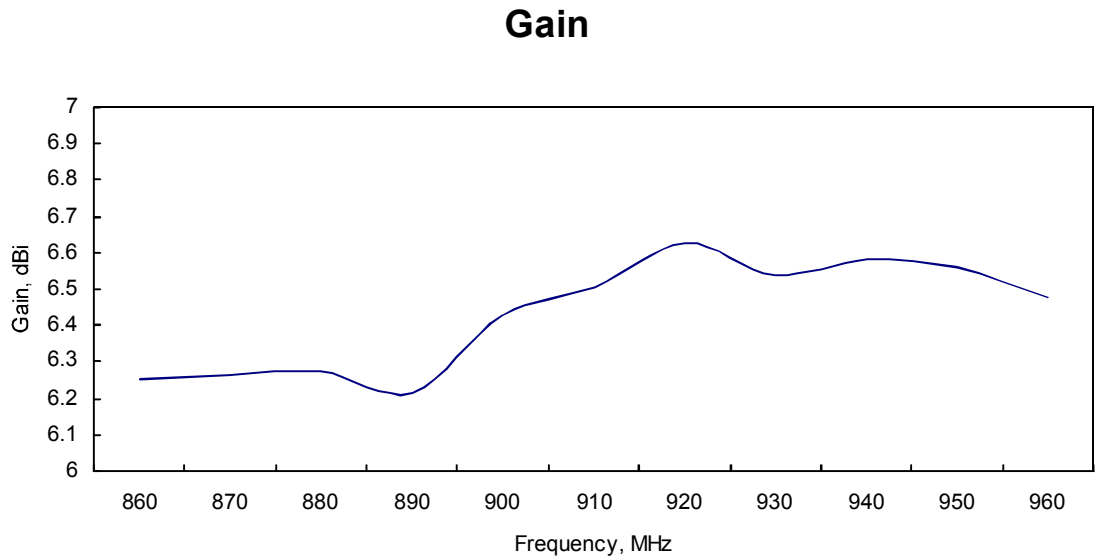
Gain	6.5 dBi
Input Frequency	860 – 960MHz
VSWR	< 2:1
Polarization	Linear
Input Impedance	50 Ohm (nominal)

Mechanical:

Dimensions	127mm x 201mm x 73mm
Weight	0.7 lbs

VSWR and Gain:





More Information:

For additional information, please visit the SkyeTek website or contact SkyeTek:

http://www.skyetek.com

Sales: sales@skyetek.com
Technical Support: techsupport@skyetek.com

APPENDIX D

SKYEMODULE M9 RFID READER DATASHEET

physical made digital

skyetek

SkyeModule M9

900MHz UHF RFID Reader/Writer

BENEFITS:

» Meets strict regulatory requirements for worldwide operation

» Superior embeddability for fast integration and time-to-market

» Variety of tag vendor choices to comply with Wal-Mart, DoD, ATA

» Low power consumption

» Cost-effective and highly scalable

» Common Blade technology: common hardware and software interface with the SkyeModule M2 HF reader for maximum design and solution flexibility

FEATURES:

» 862-955 MHz

» Tiny footprint - 49% smaller than a business card

» Extensive tag compatibility and optimization with Tagnostic® and TagIQ™

» Minimal power consumption for maximum read range

» Configurable output power

» Simple firmware upgrades

» Variety of host interfaces: TTL, USB, SPI & I^2C

» Simple and intuitive API

» Secure memory support

Product Overview

The SkyeModule™ M9 is the world's smallest, globally compliant UHF module. Its one-of-a-kind combination of high performance, security, and cost/space/power efficiency makes it the industry's price per performance leader, delivering the following benefits:

Ease of integration by using the SkyeAPI, a single library that abstracts, simplifies, and automates tag and protocol-specific functions from the host system.

Investment protection through SkyeTek's Advanced Universal Reader Architecture (AURA) permitting upgrading of modules in the field to grow with the evolution and cost savings in tag and reader technologies.

Tagnostic® support for more EPC Class 1 Gen 1/2 and ISO 18000-6B tags than any other comparable reader allowing customers to fully optimize their application.

TagIQ™ that recognizes the unique characteristics of each tag so that read/write performance is maximized for each individual tag type.

Global SKU that provides regulatory pre-scan certification for major markets including FCC, ETSI (302 208), Korea, Taiwan, Australia/New Zealand, Singapore & Hong Kong.

Unparalleled size that is half the size of a typical business card.

Performance optimization achieved through best-in-class power control (10 – 27dBm), noise reduction technology, and power management – essential for embedded applications.

Industry's first standards-based UHF security. Industry-leading privacy protection and anti-counterfeiting/anti-tampering available on any C1G2 tag with user memory.

Enhanced reliability through anti-collision and dense reader mode capability.

Unprecedented price-performance and TCO, best exemplified by ReaderWare and ReaderDNA licensing which allows customers to manufacture modules at cost.

Applications

The SkyeModule M9 has been created specifically for several applications that share common requirements for tag support, protocol, performance, and security. The M9 is an ideal solution for:

- Product Authentication and Anti-counterfeiting
- Handheld Reading/Encoding
- Inventory Management
- Printing and Encoding
- Patron Management
- Access Control
- Asset Management

» www.skyetek.com

SkyeModule M9

Software and Security

Software
SkyeAPI C/.NET API
SkyeTek Protocol v3
SkyeWare 4 developer interface
Demonstration applications

SkyeOS™ Embedded
TagIQ™
Fast Inventory with anti-collision
Field upgradeable firmware bootloader

SkyeSecurity[1]
Clone and tamper protection
Encryption (AES, TDEA)
Key Derivation Function (KDF)
Pseudo-Random Number Generator (PRNG)
Secure key store

Tag Support[2]

Protocol	Verified Manufacturers
EPC C1G1	Alien, STMicro
ISO18000-6B	Fujitsu, NXP, UPM Raflatac/Rafsec
EPC C1G2 / ISO18000-6C	Alien, Atmel, Avery Dennison, Hitachi, Impinj, Omron, TI, UPM Raflatac/Rafsec
IP-X, EM4122, EM4444	EM Microelectronics

Specifications

Frequency
862-955 MHz

Physical	CF[3]	MH[3]
Length:	66 mm	70 mm
Width:	36 mm	53 mm
Height:	5 mm	9 mm
Weight:	10.7 g	12.5 g

Environment
Storage Temperature: -30°C to 85°C
Operating Temperature: -20°C to 70°C

Host Communication Interfaces/ Data Rates
UART(TTL): 9.6-115.2 kbps
SPI Mode 1: up to 10MHz
USB 2.0 Full Speed: 12 Mb/s
I²C: 100/400 KHz

Peripheral I/O Connection
7 programmable GPIO pins CF
4 programmable GPIO pins MH

Compliance[4]
FCC 15.247
EN 301-489
AS/NZS 4268:2003
HKTA 1049
MIC 2005-50
EN 302-208
EN 61000-4-3
DGT LP002
IDA TS SRD
RoHS

Transponder Communication Rate
EPC C1G1: slow & fast modes
EPC C1G2 / ISO 18000-6C: 40, 80 kbps
ISO 18000-6B: 40 kbps

Air-interface Protocols
EPC C1G1
EPC C1G2 / ISO 18000-6C
ISO 18000-6B
ISO 18000-6A[5]

Antenna Connection
50 Ω port with MMCX (female)
VSWR 1.5:1 or lower for best performance

Current Consumption
Sleep Mode: 5 mA
Idle Mode: 170 mA
Scan Mode: 800mA @ 27 dBm
650mA @ 24 dBm
500mA @ 21 dBm

Supply Voltage
4.5-5.5 V

Output Power
Adjustable 10-27 dBm @ 0.1 dB steps
Power Accuracy: ±1 dBm

Singulation Performance
Up to 50 tags/second (25-35 typical)

Read Range
Approx. 3.5m with 6 dBi linearly polarized antenna

Performance dependent on tag type, configuration, and other environmental conditions

DKM9 - SkyeModule M9 Developer Kit

The developer kit for the SkyeModule M9 includes all hardware and software components required to integrate UHF RFID technology quickly and easily into any application:

Hardware
- 1 M9 SkyeModule
- 1 Host Interface Board
- 1 860-960MHz External Antenna
- 1 9V Power Supply
- 1 RS-232 Cable
- 1 USB Cable
- SkyeTek sample tag kit
  - EPC Class1 Gen1, EPC Class1 Gen2, and ISO18000-6B label tags
  - Variety of labels and form factors

Software
- SkyeWare 4 Dev/Demo Software
- Software Libraries (API): C, .NET
- Protocol Command Builder
- Command Line Interface
- Windows DLL

Service
- Technical Support

Notes: [1]Available via firmware upgrade, [2]See Tag Support Matrix for complete details, [3]CF = CompactFlash-style form-factor; MH = Mounting Hole format, [4]Pre-scan compliant. Fit-for-use products require additional certification. [5]Future firmware release.

SkyeTek Reader Technology SkyeTek provides a variety of reader technology at both 13.56 MHz (HF) and 860- 960 MHz (UHF). ReaderDNA, a comprehensive reference design, is available for component level integration of the technology including complete design files, BOM, and test fixture. All SkyeTek readers leverage powerful firmware that drastically reduce hardware costs and are delivered in conjunction with ReaderDNA. SkyeModules are controlled via the SkyeTek Protocol, a powerful but simple communication protocol that grants the user access to all features of an RFID transponder. Further, they have been designed with flexible and modular embedded software that allows one to select only the features desired.

About SkyeTek:

SkyeTek, Inc., maker of SkyeWare™, is the leading supplier of RFID reader software and reference designs that enable the pervasive adoption of RFID technology. SkyeTek's Tagnostic® reader technology works with most industry standard tags and smart labels, its low power requirements and small form factor make it the optimal choice for embedding into new or existing products. SkyeTek's RFID reader technology is available in several formats including reader modules, hardware reference designs, and the SkyeWare software suite. SkyeTek markets to OEM customers in targeted vertical markets with several high-volume licensing options available.

For more information:
11030 Circle Point Road, Ste 300
Westminster, Colorado 80020 USA
ph: 720.565.0441
www.skyetek.com

skyetek

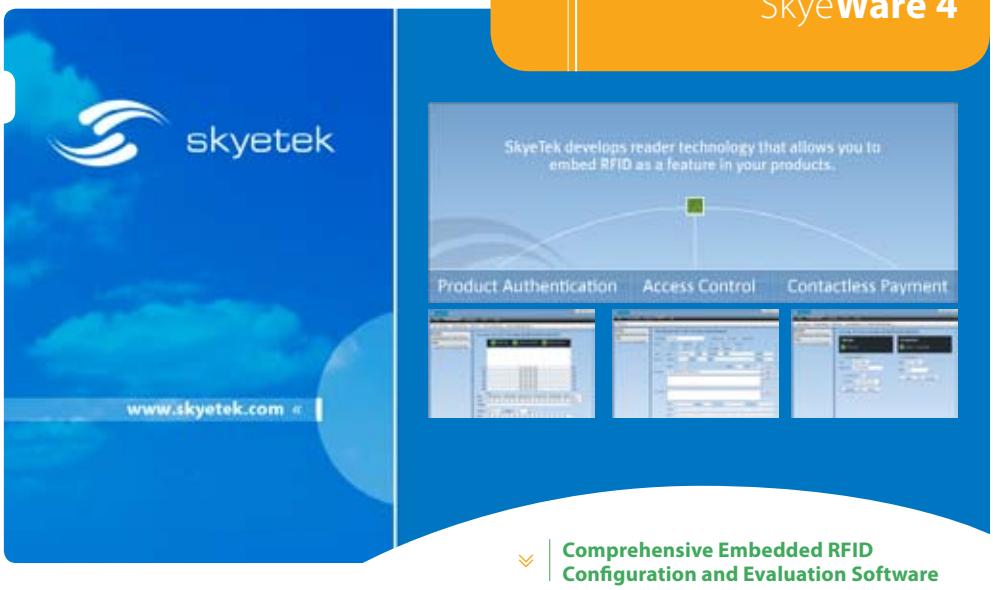
APPENDIX E

SKYEWARE4 DATASHEET

rfid as a feature

skyetek™

SkyeWare 4



Comprehensive Embedded RFID Configuration and Evaluation Software

SkyeWare4 is included in all SkyeTek Module Developer Kits.

Benefits

- Flexible enough for pilot and production designs in support of nearly any application
- Fast and easy completion of reader evaluations
- Quick launch of multiple demonstrations
- Easy-to-use, common interface for configuring and testing multiple tag types
- Clear visibility of tag data on multiple tag types

Featured Embedded Applications

- Product Authentication
- Access Control & Payment
- Patron Management
- Item-Level Inventory

Product Overview

SkyeWare™ takes another leap forward as one of the most comprehensive RFID reader application development toolkits available in the industry with the introduction of SkyeWare 4 which offers a powerful and straightforward new approach to usually awkward and clumsy developer kits.

Included with every SkyeTek module Developer Kit, SkyeWare 4 provides companies a quick and intuitive interface to evaluate embedded RFID. By connecting a SkyeModule™ reader module to a computer loaded with with SkyeWare, developers can immediatly configure the reader and begin testing code. SkyeWare 4 empowers developers to focus on designing and configuring an RFID solution – instead of focusing on how to install and run the developer kit.

Flexible pilot and production designs in support of nearly any application - provides robust testing environment to support design, integration and prouction stages for successful proof-of-concepts.

Fast and easy completion of reader evaluations - unveils the full capabilities of SkyeTek's embedded reader modules for integration into OEM products to determine fit and feasibility.

Quick launch of multiple demonstrations - showcases demos for anti-collision, read range, memory, secure memory, product authentication, payment and access control all built on SkyeTek's application programming inteface, SkyeAPI.

Easy-to-use, common interface for configuring and testing multiple tag types - optimizes read range, anti-collision and environmental considerations to select ideal tags that best fit target application.

Clear visibility of tag data on multiple tag types - explores secutiry for both proprietary security on branded tags and standards-based security on generic tags with extended memory.

SkyeWare Elements:

RFID Security: SkyeWare 4 provides developers with the same security technology that is used in DoD, Internet, and Financial applications. The Advanced Encryption Standard (AES) family of encryption provides up to 256-bit symmetric encryption and the Secure Hash Algorithm (SHA) family of hashing provides best-in-class anti-cloning protection for applications such as product authentication and anti-counterfeiting. SkyeTek implements these algorithms in industry standards fashion allowing customers to secure generic tags and avoid the 100 – 200% premium that is charged on tags that implement a proprietary form of cryptography. SkyeWare 4 also support proprietary forms of security including MIFARE, DESFire and CryptoRF and for companies desiring to integrate with existing systems or specific application requirements.

Tagnostic®: SkyeTek's reader modules support more UHF and HF tags than any other comparable reader on the market. Truly tag agnostic, SkyeWare 4 allows developers to make design and integration decisions around tag selection long before creating a prototype.

TagIQ™: SkyeTek refers to its intimate knowledge of tags and their various behaviors as TagIQ™ and it ensures that the hardware and software platform squeezes every ounce of performance out of every supported tag.

» www.skyetek.com

About Skyetek:

SkyeTek, Inc. is the leading supplier of RFID reader software and reference designs that enable the pervasive adoption of RFID technology. SkyeTek's Tagnostic™ reader technology works with most industry standard tags and smart labels, its low power requirements and a small form factor make it the optimal choice for embedding into new or existing products.

Setup Wizard

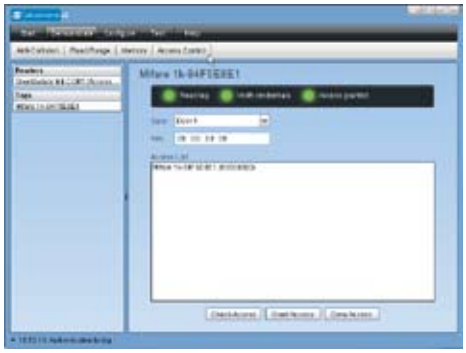
SkyeWare 4 provides a hands-on wizard to help users set up a reader and application without the overhead of wondering what to do next.

The wizard provides access to advance options such as firmware loading and configuration screens.

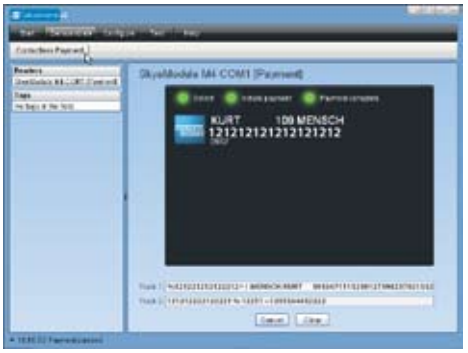
Application Demos

SkyeWare 4 supports demos for nearly any RFID application and delivers specific user interfaces for quickly setting up and demonstrating three of the most common applications:

Product Authentication. Demonstrates the features of the Product Authentication firmware personality. Tags are set up in a single screen with secure memory and then can be read only by an authorized reader.

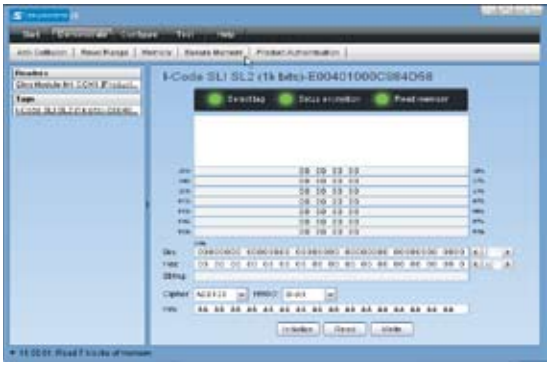


Access Control. Demonstrates using tags supported within the Access Control firmware personality to provide access to secure locations by applying different policies.



Contactless Payment. Detects contactless payment credit cards and demonstrates retrieving Track 1 and Track 2 data using the Contactless Payment firmware personality.

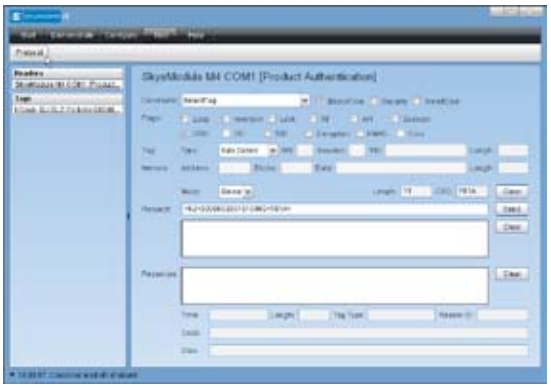
SkyeTek's RFID reader technology is available in several formats including reader modules, hardware reference designs, and the ReaderWare software suite. SkyeTek markets to OEM customers in targeted vertical markets with several high-volume licensing options available.



Memory & Security

Memory maps are easily defined, as SkyeWare 4 allows users to read and write memory of all supported tags in a common memory interface, including DESfire data and record files.

Secure Memory: Invoking security from an industry-leading standards library users can read, write, and secure tag memory by encrypting data and/or creating a digital signature. The user can switch between the Memory and Secure Memory screens to verify that the data is encrypted when secure memory is enabled.



Development & Test Tools

SkyeWare 4 includes a host of test and development tools that allow developers to determine the best configuration for their needs and to test API commands. SkyeModule reader system and radio parameters can be quickly configured and firmware can be uploaded through an easy to use interface.

skyetek™

Global Headquarters:
11030 Circle Point Road,
Ste 300
Westminster, Colorado
80020. USA
ph: 720.565.0441
www.skyetek.com