ภาคผนวก

ผนวก ก

ผลการทดสอบระยะการอ่านของเครื่องอ่าน

การทดสอบระยะการอ่านของเครื่องอ่านที่ใช้ชิป AS3990 ของบริษัท Austriamicrosystems นั้นผู้วิจัยได้ทำการระยะการอ่านที่สามารถอ่านได้จริง โดยใช้ป้าย 2 ประเภทคือ ป้าย ISO 180000 – 6 ประเภท B และ C ในการวัดนั้นผู้วิจัยได้ทำการวัดในแนวแกนตั้งและแนวแกนนอน โดยทำการหมุนทิศทางของเครื่องอ่านครั้งละ 10 องศา จนครบ 360 องศา

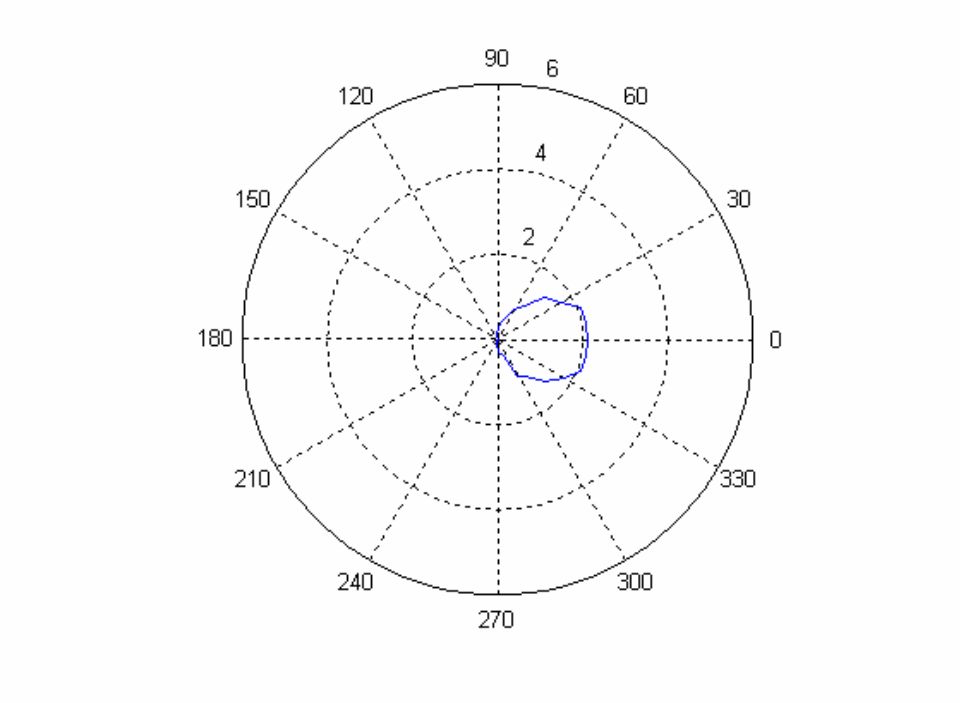
รูปที่ ก.1 แสดงชุดวัดมุมของขาตั้งเครื่องอ่าน

รูปที่ ก.2 แสดงการติดตั้งสายอากาศของเครื่องอ่านแนวนอน

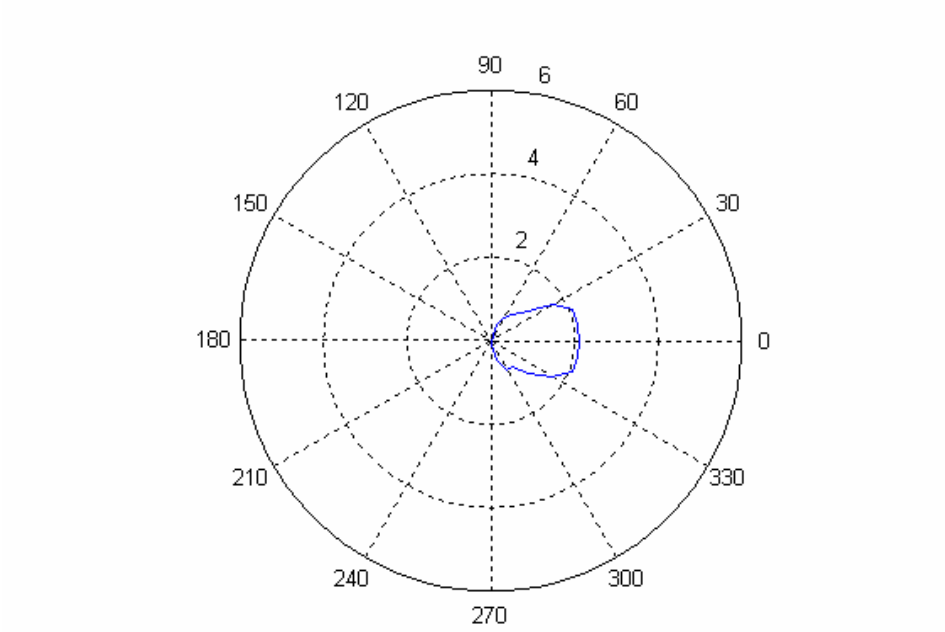
รูปที่ ก.2 แสดงการติดตั้งสายอากาศของเครื่องอ่านแนวตั้ง

1) ป้าย ISO 18000 – 6 ประเภท B ในแนวแกนนอน

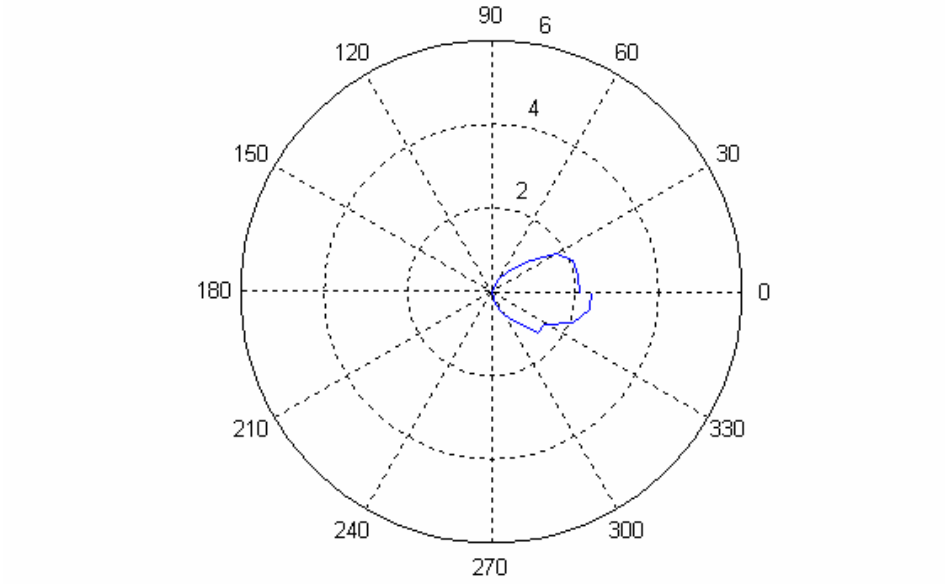
ในการวัดระยะการอ่านของเครื่องอ่านในแนวแกนนอนนั้นแบ่งออกเป็น 2 ส่วนคือ ระยะการอ่านของป้าย 18000 – 6 ประเภท B จำนวน 1 และ 2 ป้าย โดยตัวเลขรอบนอกแสดงองศาที่ทำการหมุนเครื่องอ่าน และตัวเลขของแต่ละวงแสดงระยะห่างจากเครื่องอ่านมีหน่วยเป็นเมตร



รูปที่ ก.3 ระยะทางที่สามารถอ่านป้าย ISO 18000 – 6 ประเภท B 1 ป้ายในแนวแกนนอน



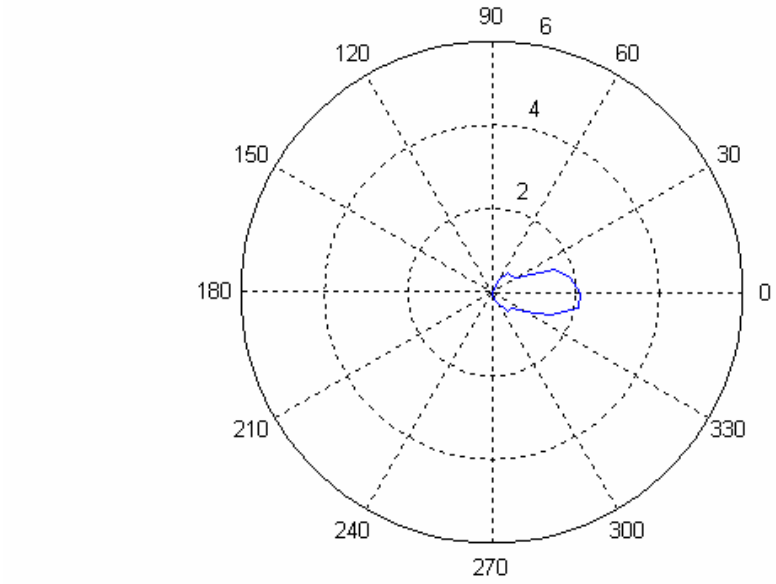
รูปที่ ก.3 ระยะทางที่สามารถอ่านป้าย ISO 18000 – 6 ประเภท B 2 ป้ายในแนวแกนนอน



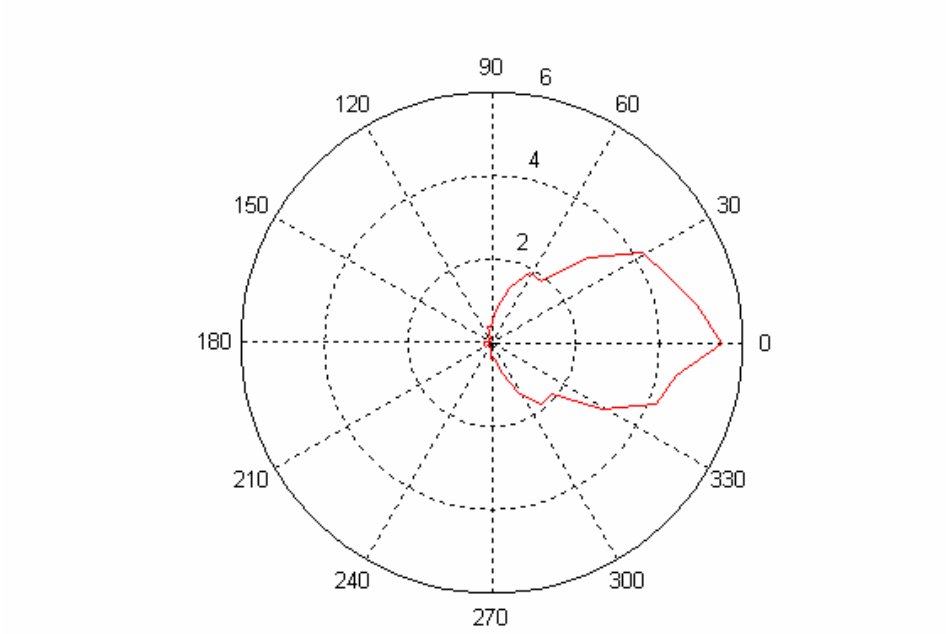
รูปที่ ก.4 ระยะทางที่สามารถอ่านป้าย ISO 18000 – 6 ประเภท B 1 ป้ายในแนวแกนตั้ง

2) ป้าย ISO 18000 – 6 ประเภท B ในแนวแกนตั้ง

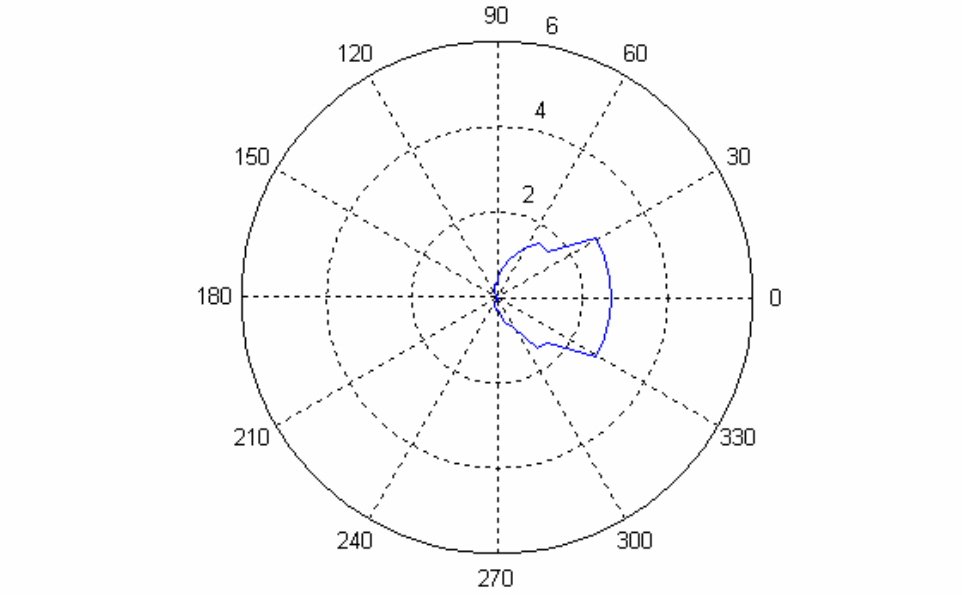
ในการวัดระยะการอ่านของเครื่องอ่านในแนวแกนตั้งนั้นแบ่งออกเป็น 2 ส่วนคือ ระยะการอ่านของป้าย 18000 – 6 ประเภท B จำนวน 1 และ 2 ป้าย โดยตัวเลขรอบนอกแสดงองศาที่ทำการหมุนเครื่องอ่าน และตัวเลขของแต่ละวงแสดงระยะห่างจากเครื่องอ่านมีหน่วยเป็นเมตร



รูปที่ ก.5 ระยะทางที่สามารถอ่านป้าย ISO 18000 – 6 ประเภท B 2 ป้ายในแนวแกนตั้ง



รูปที่ ก.6 ระยะทางที่สามารถอ่านป้าย ISO 18000 – 6 ประเภท C 1 ป้ายในแนวแกนนอน



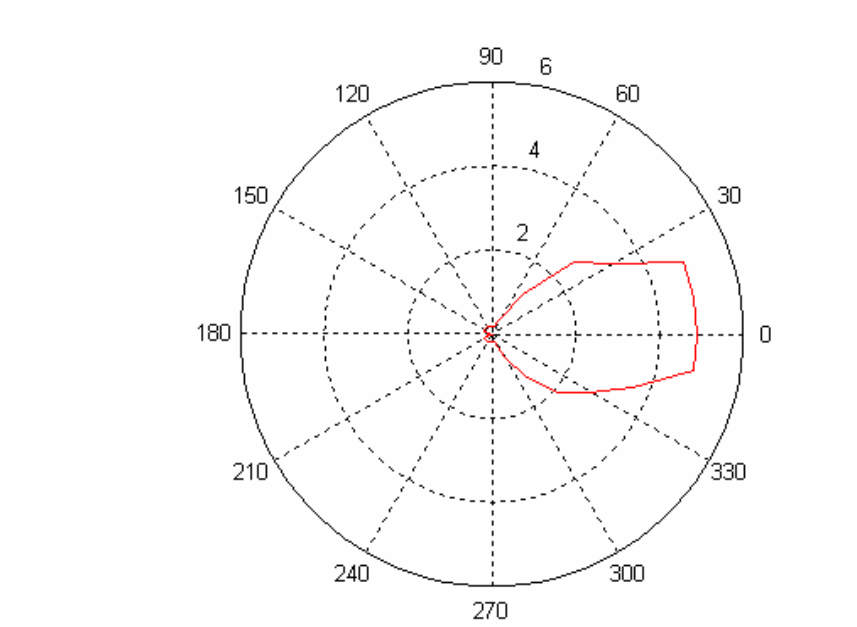
รูปที่ ก.7 ระยะทางที่สามารถอ่านป้าย ISO 18000 – 6 ประเภท C 2 ป้ายในแนวแกนนอน

3) ป้าย ISO 18000 – 6 ประเภท C ในแนวแกนนอน

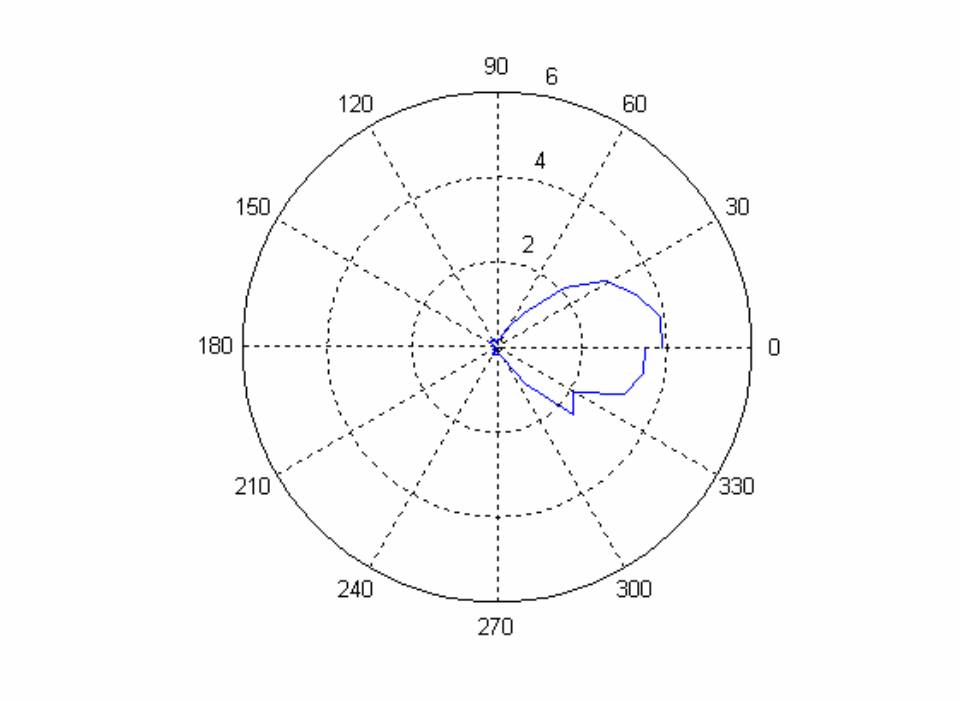
ในการวัดระยะการอ่านของเครื่องอ่านในแนวแกนนอนนั้นแบ่งออกเป็น 2 ส่วนคือ ระยะการอ่านของป้าย 18000 – 6 ประเภท C จำนวน 1 และ 2 ป้าย โดยตัวเลขรอบนอกแสดงองศาที่ทำการหมุนเครื่องอ่าน และตัวเลขของแต่ละวงแสดงระยะห่างจากเครื่องอ่านมีหน่วยเป็นเมตร

4) ป้าย ISO 18000 – 6 ประเภท C ในแนวแกนตั้ง

ในการวัดระยะการอ่านของเครื่องอ่านในแนวแกนตั้งนั้นแบ่งออกเป็น 2 ส่วนคือ ระยะการอ่านของป้าย 18000 – 6 ประเภท C จำนวน 1 และ 2 ป้าย โดยตัวเลขรอบนอกแสดงองศาที่ทำการหมุนเครื่องอ่าน และตัวเลขของแต่ละวงแสดงระยะห่างจากเครื่องอ่านมีหน่วยเป็นเมตร



รูปที่ ก.8 ระยะทางที่สามารถอ่านป้าย ISO 18000 – 6 ประเภท C 1 ป้ายในแนวแกนตั้ง



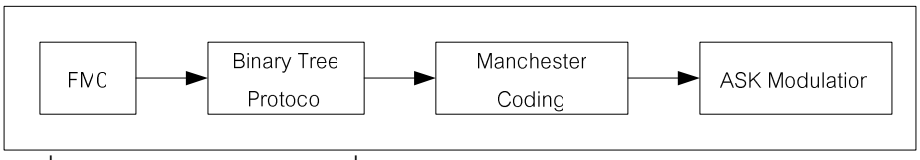
รูปที่ ก.9 ระยะทางที่สามารถอ่านป้าย ISO 18000 – 6 ประเภท C 2 ป้ายในแนวแกนตั้ง

ผนวก ข

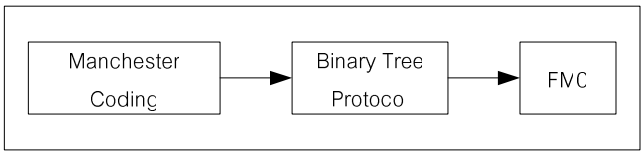
สถาปัตยกรรมของเครื่องอ่านป้ายอาร์เอฟไอดีตามมาตรฐาน ISO 18000 – 6

ข-1 สถาปัตยกรรมของมาตรฐาน ISO 18000 – 6 ประเภท B และ C

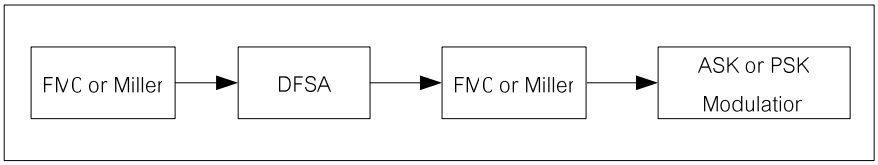
สถาปัตยกรรมของมาตรฐาน ISO 18000 – 6 ประเภท B และ C นั้นมีความแตกต่างกันในส่วนของการเข้ารหัสและอัลกอริทึมที่ใช้ในการป้องกันการชนกันของข้อมูลดังรูปที่ ข. 1, ข. 2, ข. 3 และ ข. 4 ซึ่งในมาตรฐาน ISO 18000 – 6 ประเภท B นั้นใช้ Binary Tree Protocol ในการป้องกันการชนกันของข้อมูล ส่วนมาตรฐาน ISO 18000 – 6 ประเภท C นั้นใช้ DFSA ในการป้องกันการชนกันของข้อมูล ส่วนการเข้ารหัสข้อมูลนั้น ISO 18000 – 6 ประเภท B ใช้ FM0 และ Manchester ในการเข้ารหัส แต่มาตรฐาน ISO 18000 – 6 ประเภท C นั้นสามารถเลือกใช้ได้ทั้ง FM0 และ Miller ในการเข้ารหัสของข้อมูล อีกทั้งยังสามารถเลือกการมอดูเลชันได้ทั้ง ASK และ PSK



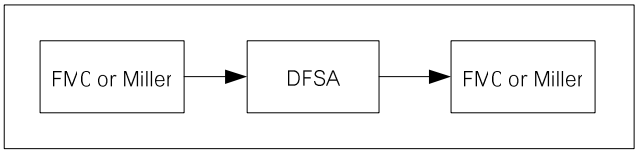
รูปที่ ข.1 สถาปัตยกรรมของเครื่องอ่านตามมาตรฐาน ISO 18000 – 6 ประเภท B



รูปที่ ข.2 สถาปัตยกรรมของป้ายตามมาตรฐาน ISO 18000 – 6 ประเภท B



รูปที่ ข.3 สถาปัตยกรรมของเครื่องอ่านตามมาตรฐาน ISO 18000 – 6 ประเภท C



รูปที่ ข.4 สถาปัตยกรรมของป้ายตามมาตรฐาน ISO 18000 – 6 ประเภท C

ข-2 คำสั่งที่ใช้ในมาตรฐาน ISO 18000 – 6 ประเภท B

ตารางที่ ข.1 แสดงคำสั่งที่ใช้งานในอัลกอริทึมป้องกันการชนกันของข้อมูลตามมาตรฐาน ISO18000 – 6 ประเภท B

คำสั่ง	รหัสคำสั่ง	Preamble	Delimiter	Command	Address	Mask	WORD_DATA	CRC-16
GROUP_SELECT_EQ	"00"	√	√	8 bits	8 bits	8 bits	64 bits	16 bits
GROUP_UNSELECT_EQ	"04"	√	√	8 bits	8 bits	8 bits	64 bits	16 bits
FAIL	"08"	√	√	8 bits	-	-	-	16 bits
SUCCESS	"09"	√	√	8 bits	-	-	-	16 bits
RESEND	"15"	√	√	8 bits	-	-	-	16 bits
INITIALIZE	"0A"	√	√	8 bits	-	-	-	16 bits
DATA_READ	"0B"	√	√	8 bits	-	-	-	16 bits
READ	"0C"	√	√	8 bits	-	-	-	16 bits

ตารางที่ ข.2 ข้อมูลที่ป้ายตอบสนองตามมาตรฐาน ISO 18000 – 6 ประเภท B

คำสั่ง	Preamble	ID	CRC-16
GROUP_SELECT_EQ	√	64 bits	16 bits
GROUP_UNSELECT_EQ	√	64 bits	16 bits
FAIL	√	64 bits	16 bits
SUCCESS	√	64 bits	16 bits
RESEND	√	64 bits	16 bits
INITIALIZE	√	64 bits	16 bits
DATA_READ	√	64 bits	16 bits
READ	√	64 bits	16 bits

ข-3 คำสั่งที่ใช้ในมาตรฐาน ISO 18000 – 6 ประเภท C

ตารางที่ ข.7 แสดงคำสั่งที่ใช้งานในอัลกอริทึมป้องกันการชนกันของข้อมูลตามมาตรฐาน ISO 18000 – 6 ประเภท C

คำสั่ง	**รหัสคำสั่ง**	CRC	**ความยาวของคำสั่ง**
QueryRep	"00"	X	4
ACK	"01"	X	18
Query	"1000"	5 bits	22
QueryAdjust	"1000"	X	9
Select	"1000"	16 bits	>44
NAK	"1000"	X	8
Req_RN	"1000"	16 bits	40

ตารางที่ ข.8 ข้อมูลที่ป้ายตอบสนองตามมาตรฐาน ISO 18000 – 6 ประเภท B

คำสั่ง	RN 16	ID	CRC-16
QueryRep	√	-	-
ACK	-	21 - 528 bits	16 bits
Query	√	-	-
QueryAdjust	√	-	-
Select	-	-	-
NAK	-	-	-
Req_RN	√	-	16 bits

ผนวก ค

ผลงานตีพิมพ์ทางวิชาการ

- S. Makwimanloy, P. Kovintavewat, U. Ketprom, C. Tantibundhit and C. Mitrpant, “A New Anti-Collision Based on A-Priori Information,” in *Proc. of ECTI-CON* 2008, Krabi, Thailand, vol. II, pp. 733 – 736, May 14 – 16, 2008.
- S. Makwimanloy, P. Kovintavewat, U. Ketprom and C. Tantibundhit, “A Novel Anti-Collision Algorithm for High-Density RFID Tags,” in *Proc. of ECTI-CON* 2009, Thailand, vol. xx, pp. xxx – xxx, May 6 – 9, 2009.

Proceedings of ECTI-CON 2008

A New Anti-Collision Based on A-Priori Information

Sarawut Makwimanloy[1], Piya Kovintavewat[2], Urachada Ketprom[3], Charturong Tantibundhit[4], Chaichana Mitrpant[5]

[1,4] *Electrical and Computer Engineering Department, Thammasat University, Thailand*
[2] *RFID Technology and Applications Research Unit, Nakhon Pathom Rajabhat University, Nakhon Pathom, Thailand*
[3,5] *RFID Program, National Electronics and Computer Technology Center (NECTEC), Thailand*

Email: [1]makwimanloy@hotmail.com, [2]piya@npru.ac.th, [3]urachada.ketprom@nectec.or.th,
[4]tchartur@engr.tu.ac.th, [5]chaichana.mitrpant@nectec.or.th

***Abstract*— A collision occurs when more than two tags present in the reader's field of a radio frequency identification system. Anti-collision algorithms such as binary trees and dynamic framed slotted aloha (DFSA) have been employed to prevent such a collision. The identification number of tag consists of 64 bits and certain parts of 64 bits can be considered a priori-information. This paper proposes a new anti-collision algorithm based on a-priori information about the manufacturer code. This prior-information reduces the number of bits to analyze in the algorithm, hence reduces the operation time for the faster read-performance. Results indicate that the proposed anti-collision algorithm required a less number of used time slots, thus minimizing the operation time more than 50% comparing to the existing ones.**

I. Introduction

Radio-frequency identification (RFID) system has been introduced to uniquely identify the object of interest. The RFID system basically consists of a reader and a tag, communicating via radio frequency waves. Currently, the RFID system has been employed in a variety of applications, such as transportation, ticketing, access control, animal identification, and so forth.

When more than one tag in the reader's field communicates with the reader at the same time, a *collision* will occur, resulting in the failure of that communication. In this case, each tag has to restart communication with the reader. To prevent this problem, an anti-collision algorithm must be used. Based on the International Standards Organization (ISO) and EPCglobal (EPC), there are 3 types of anti-collision algorithms, namely, Binary Tree (BT) [1, 2], Framed Slotted ALOHA (FSA) [1], and Dynamic Framed Slotted ALOHA (DFSA) [1, 3] algorithms.

Many improved anti-collision algorithms have recently been proposed in the literature. For example, reference [1] presents the analysis and simulation of several RFID anti-collision algorithms and partitioning of tags for near-optimum RFID anti-collision performance. Partitioning technique enabling a faster accurate estimation on the number of contending tags, which yields much higher throughput against previous non-partitioning approaches, was proposed in [4].

Figure 1 shows a structure of a tag's ID number consisting of 64 bits. This first 32 bits (the 1-st bit to the 32-nd bit) represents a serial number of each tag, whereas the last 32 bits indicates the manufacturer code (the numbers of registered company and the type of product) which is never changed. Normally, the manufacturer code is hidden from the users. Thus, all existing anti-collision algorithms use all 64 bits in a tag's ID number to process, starting from the 64-th bit to the 1-st bit [2, 3]. However, for a special case where a-priori information about the manufacturer code is known,

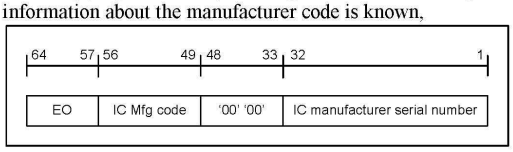


Fig. 1. A structure of a tag's ID number used in ISO 18000-6 [7].

there is no anti-collision algorithm that exploits such information to improve its performance. In this paper, we therefore propose a new anti-collision algorithm based on a-priori information, which performs better than the existing algorithms in terms of the number of used time slots (the less the used time slot, the faster the algorithm). The performance comparison of different anti-collision algorithms used in ISO and EPC standards is also provided to serve as a guideline for users to decide which algorithm should be utilized for a given condition.

The rest of this paper is organized as follows: Section II briefly describes some anti-collision algorithms used in ISO and EPC standards. A new anti-collision algorithm based on a-priori information is explained in Section III. Section IV compares the performance of different anti-collision algorithms. Finally, Section V concludes this paper.

II. Existing Anti-Collision Algorithms

This section briefly describes how the anti-collision algorithms (i.e., BT, FSA, and DFSA) perform.

A. Binary Tree Algorithm

A Binary Tree (BT) algorithm is employed in ISO 18000-6 Type B and EPC Class 1 [2]. For ISO 18000-6 Type B, it divides tags into two groups based on the most significant bit of the tag's ID number, denoted as MSBID (i.e., the 64-th bit in Fig. 1), which consists only of bits "0" and "1". To search a tag, a dividing process continues adding up the number "0"

978-1-4244-2101-5/08/$25.00 ©2008 IEEE

and "1" into each group, until finding a tag [1, 5, 6]. Note that we consider only the case where the tags do not support a random generator in hardware for group selection [7], meaning that the BT algorithm operates on the tag's ID number. Fig. 2 shows how the BT algorithm works. Suppose there are 3 tags in the reader's field, namely, "011," "101," and "110," where the first digit is MSB. To obtain all tags, the reader begins a search by sending bit "0" (step 1) to all tags and waits for the response. There is one response sent to

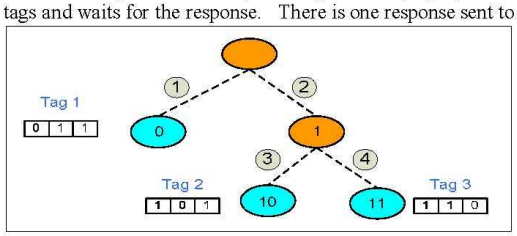


Fig. 2. How Binary Tree algorithm works.

reader because there is only one tag beginning (on the left hand) with bit "0." Now, the reader recognizes Tag 1. Next, the reader sends bit "1" (step 2) to the other two tags, i.e., "101" and "110". In this case, a collision occurs because two tags respond back at the same time. Then, the reader sends another bit "0" (step 3) to these two tags. At this time, the reader can recognize Tag 2 because the reader receives only one response. Then, the reader sends another bit "1" (step 4) to the remaining tag, which results in only one response from Tag 3 sent to the reader. This means there is no other tags in the reader's field, thus implying the end process of the BT algorithm.

To compare the performance of different anti-collision algorithms, we use the required total number of commands sent from the reader to the tag as a criterion. Each command is referred to as one *used time slot* (or, in short, *slot*). Assuming that each slot uses the same processing time, the algorithm that requires a large number of slots will operate slow. For example, in Fig. 2, the total number of slots that the reader requires to recognize all three tags is 4 slots. This means that the number of slots is increased one slot every time when the reader sends out each one bit, i.e., "0" or "1."

For the BT algorithm used in EPC Class 1, the searching procedure is similar to that used in ISO 18000-6 Type B, but the BT algorithm in EPC Class 1 will divide a group into 8 subgroups based on 3 bits at each step [2]. There are both advantages and disadvantages between these two BT algorithms as illustrated in Section IV.

B. *Framed Slotted ALOHA (FSA)*

This algorithm developed from the Slotted Aloha algorithm is used in ISO 18000-6 Type A [7]. It divides tags into many groups according to the number of slots specified by a reader. All tags will random the slot number, and the tags having the same number will be in the same group.

First, the reader sends an "Init_round" command to tags for setting the number of slots within one frame. Next, tags randomly pick a slot number between 0 to "slot_number," and record it into a "slot_count." If the "slot_count" equals to the required "slot_number," the tag will respond to the reader. Then, three possible outcomes could happen:

1) No Tag response
   Reader sends a "Close_slot" command to all tags to increase a "slot_count."
2) One Tag response
   Reader passes a "Next_slot" command to the responded tag so as not to respond the reader in the next frame.
3) Multiple Tags response
   Reader recognizes a collision and will send a "Close_slot" to the collided tags to increase a "slot_count."

This procedure repeats until the reader can identify all tags completely [6]. In FSA, the total number of slots is equal to all slots used in the FSA algorithm.

C. *Dynamic Framed Slotted ALOHA (DFSA)*

This algorithm developed from FSA is utilized in EPC Class 1 Generation 2. It works similar to FSA, except that the number of slots in each frame can be adjusted based on a Q-parameter [3, 4]. In DFSA, a reader sends a command to tags for specifying a Q-parameter. Next, tags randomly select and record a value between 0 and $2^{\text{Q-parameter}}$ -1 into a "slot_counter." The tag with a "slot_counter" equal to 0 will respond back to the reader. Then, the reader sends a "Query" command to decrease the value of a "slot_counter," and also sends a "QueryAdjust" command to adjust the value of Q-parameter. However, if there are empty or collided slots more than the number of accepted slots, tags will repeat all steps until the reader can identify all tags.

III. Proposed Anti-Collision Algorithm

When some a-priori information about the tags is known, we can exploit such information to improve the performance of the existing anti-collision algorithms. In this paper, we consider three types of a-priori information, i.e.,

1) Suppose a-priori information about the total number of tag's manufacturers is known. We found that for each application, if possible, it is preferable to employ all tags from one manufacturer in the RFID system.
2) Suppose the total number of tags needed to identify is known. In this case, we found that there is no significant performance improvement when we use this information in the anti-collision algorithm.
3) Suppose the manufacturer code of tags is known. In this case, we can use this information in the anti-collision algorithm to reduce the time required to identify all tags.

In a searching process, all anti-collision algorithms begins with the MSBID (i.e., the 64-th bit in Fig. 1), and continues to the 1-st bit. The proposed anti-collision algorithm is the existing anti-collision algorithm that exploits a-priori information. This means that if we know a manufacturer code (i.e., ranging from the 64-th bit to the 33-th bit in Fig. 1), the proposed anti-collision algorithm can start the searching process at the 32-th bit, instead of the 64-th bit. Clearly, this

will reduce the time required to identify all tags. As shown in simulation, the proposed anti-collision algorithm identifies all tags much faster than other algorithms.

IV. Simulation

Performance comparison of the existing anti-collision algorithms has been investigated in [1, 6]. Here, we compare the performance of the proposed anti-collision algorithm with the existing algorithms in different aspects as follows.

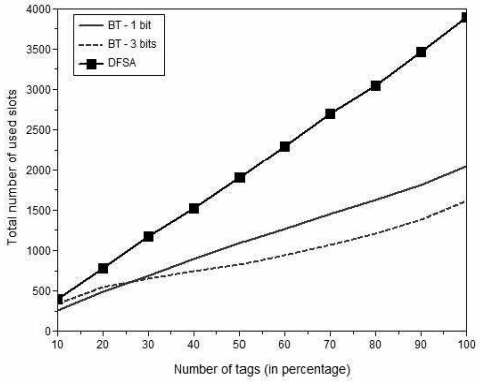


Fig. 3. Performance comparison of BT 1-bit, BT 3-bit, and DFSA.

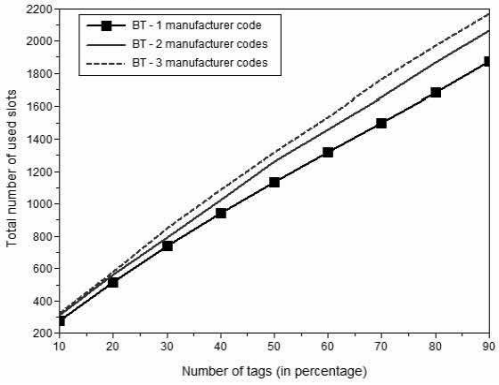


Fig. 4. Performance of the BT with multiple manufacturer codes.

A. DFSA and Binary Tree

We compare the performance of three algorithms, i.e., Binary Tree 1 bit (BT 1-bit), Binary Tree 3 bits (BT 3-bit), and DFSA, and assume that the tag's ID number consists of 10 bits (all random bits). Note that we cannot simulate the tag's ID number of 64 bits due to the limitation of memory requirement. Figure 3 compares the performance of different algorithms, where the x-axis represents the number of used tags in percentage, and the y-axis is the total number of used slots. The less the number of used slots, the faster the algorithm. It is clear that the BT performs better than the DSFA, especially when the number of tags is large. This is because the DFSA divides groups of tags randomly into slots. Thus, tags are more likely to collide, especially when a large number of tags present in the reader's field. Furthermore, the BT 1-bit performs better than the BT 3-bit when the number of used tags is less than 25%, but worse than the BT 3-bit when the number of used tags is larger than 25%. Therefore, the selected algorithm depends on the number of used tags for a given application.

B. Binary Tree with multiple manufacturer codes

In Figure 3, we assume that the tag's ID number consists of 20 bits. Here, we consider the case where the IC manufacturer code is known and can be divided into one, two, and three groups (i.e. the first 10 bits are the same for each group, the last 10 bits are random numbers). We expected that the number of groups affects the performance of the algorithms. Figure 4 compares the performance of the BT with 1, 2, and 3 manufacturer codes, where each point is averaged by 10 data sets.

It is apparent from Fig. 4 that the BT with 1 manufacturer code performs better than that with 2 and 3 manufacturer codes. As expected, the results confirm that the more the difference in the manufacturer code, the more the number of slots required to identify all tags. Consequently, for a given application, it is preferable to use all tags from one manufacturer if possible.

C. Smart Binary Tree algorithm

Here, we compare the performance of the proposed anti-collision algorithm with the existing ones. We consider two cases of a-priori information, i.e., when the total number of tags is known and when the manufacturer code is known.

The proposed algorithm that knows when the total number of tags needed to identify is the *normal* anti-collision algorithm, but it will stop the searching process when all tags are identified. We observed that there is no significant performance improvement (not shown here) when the reader knows the total number of tags needed to identify. This is because the *normal* algorithm will also stop the searching processing when no tag responds after querying. However, if a-priori information about the manufacturer code is known, we can then improve the performance of the anti-collision algorithms. Let us denote "Smart BT *n*-bit" as the BT *n*-bit algorithm that exploits such a-priori information. We also assume that the tag's ID number consists of 20 bits (the first 10 bits represent a manufacturer code and the last 10 bits represent a random ID number). Again, we cannot simulate the tag's ID number of 64 bits because of the limitation of memory requirement. Then, with the Smart BT algorithm, the searching process skips the 10-bit manufacturer code, and starts the *normal* BT algorithm at the 10-th bit.

Figure 5 compares the performance of the BT and the Smart BT algorithms with one manufacturer code. Clearly, the Smart BT performs better than the BT. For the Smart BT algorithm, the decision point to decide whether or not 1-bit or 3-bit searching process should be used is roughly at 50% of

the number of used tags, whereas for the BT algorithm, the decision point is at 26% of the number of used tags.

Table I shows the total number of slots used in the Smart and the *normal* BT algorithm (extracted from Fig. 5). The Smart BT algorithm requires the number of slots less than the BT algorithm, approximately 50%. We also compare the performance of the BT and the Smart BT algorithms with three manufacturer codes as depicted in Fig. 6.

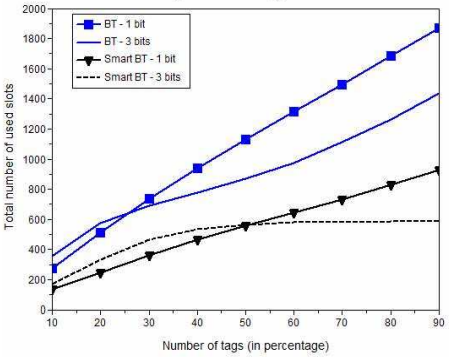


Fig. 5. Performance of the BT and the Smart BT with one manufacturer code.

TABLE I
COMPARISON OF THE NUMBER OF USED SLOTS

% of tags	One manufacturer code (total slots)				Percentage of slot reduction (%)	
	Normal		Smart			
	BT	BT 3 bit	BT	BT 3 bit	BT	BT 3 bit
30%	736	844	362	462	50.815	45.26
50%	1133	1316	557	564	50.838	57.142
70%	1497	1762	733	581	51.035	67.026

Clearly, the performance improvement is not significant. The Smart BT algorithm performs well when the numbers of tags are known prior to data communication, but the manufacturer codes of three companies have no role in time slot reduction. It is not possible for the reader to know beforehand which tags of three companies will be first read and thus keep sending the new command until no collision occurs. However, the smart BT algorithm will in general perform better than the *normal* BT algorithm. The performance comparison of existing anti-collision algorithms is summarized in Table II. The speed refers to the operation time used in each algorithm, while the complexity refers to the system request memory, computation, and other functions on tags.

V. CONCLUSIONS

The anti-collision algorithms are crucial to the application that uses a large numbers of tags. In general, the Binary Tree algorithm performs faster than the DFSA algorithm as shown in Fig. 3. Furthermore, one should employ tags with one manufacture code in each application to expedite the identification process. The proposed algorithm that exploits a-priori information performs better than the existing anti-collision algorithm in terms of the number of used time slots, resulting in the faster read.

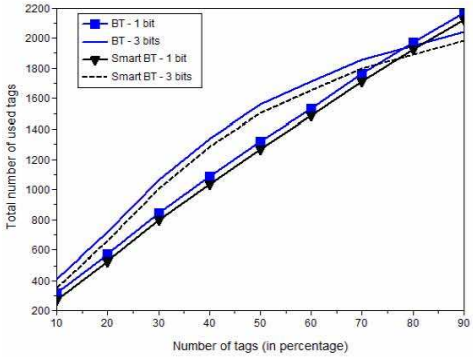


Fig. 6. Performance of BT and Smart BT with three manufacturer codes.

TABLE II
DETAILS OF EACH ALGORITHM

Type / Details	FSA	DFSA	BT 1-bit	BT 3-bit
1) Speed	slow	normal	fast	normal
2) Ability to add tags while working	√	√	X	X
3) Complexity	normal	highest	low	low
4) Security of tag's IDs	√	√	X	X

ACKNOWLEDGMENT

This work was supported by National Science and Technology Development Agency (NSTDA) and the RFID Program, National Electronics and Computer Technology Center (NECTEC), Thailand, under grant TG-44-21-50-098M.

REFERENCES

[1] T. Cheng and L. Jin, "Analysis and Simulation of RFID Anti-collision Algorithm," *IEEE Advanced Communication Technology*, vol. 1, pp. 697 – 701, Mar. 2007.

[2] EPC Global. 860MHz~930MHz Class I Radio Frequency Identification Tag Radio Frequency & Logical Communication Interface Specification Candidate Recommendation, Version 1.0.1.

[3] EPC Global. EPC™ Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz~960MHz, Version 1.0.9.

[4] W. J. Shin and J. G. Kim, "Partitioning of Tags for Near-Optimum RFID Anti-collision Performance," *IEEE Wireless communications and Networking Conference*, pp. 1673-1678, Mar. 2007.

[5] C. Abraham, V. Ahuja, A. K. Ghosh, and P. Pakanati, "Inventory Management using Passive RFID Tags: A Survey," Department of Computer Science thesis, University of Texas at Dallas, Richardson, Texas.

[6] R. Ahmed, "Performance Comparison of RFID Tag Anti-collision Algorithm using Simulation and Real Testing Based," M. Eng. thesis, Asian Institute of Technology, Thailand, May.2007.

[7] ISO/IEC 18000-6:2003(E), Part 6: Parameters for air inter-face communications at 860-960 MHz, Nov. 26, 2003.

[8] K. Finkenzeller, *RFID handbook*, John Wiley & Sons, West Sussex, 2003.

A Novel Anti-Collision Algorithm for High-Density RFID Tags

Sarawut Makwimanloy[1], Piya Kovintavewat[2], Urachada Ketprom[3], Charturong Tantibundhit[4]

[1,4] *Electrical and Computer Engineering Department, Thammasat University, Thailand*
[2] *RFID Technology and Applications Research Unit, Nakhon Pathom Rajabhat University, Nakhon Pathom, Thailand*
[3] *RFID Program, National Electronics and Computer Technology Center (NECTEC), Thailand*
Email: [1] makwimanloy@hotmail.com, [2]piya@npru.ac.th, [3]urachada.ketprom@nectec.or.th, [4]tchartur@engr.tu.ac.th

Abstract— **In a radio frequency identification (RFID) system, when more than one tag communicates with a reader at the same time, a collision will occur resulting in the failure of that communication. Many anti-collision algorithms, such as Binary Tree (BT), FSA, and DFSA have been used in ISO and EPC standards to prevent such a collision. This paper develops a new anti-collision algorithm based on the BT and the DFSA algorithms. Specifically, all tags are divided into many groups using the DSFA algorithm. Then, the tags in each group are identified using the BT algorithm. Results indicate that the proposed algorithm performs better than the existing ones in terms of the number of used time slots (the less the used time slot, the faster the algorithm).**

I. Introduction

Radio frequency identification (RFID) is a technology for automated identification. Typically, an RFID system consists of a reader and tags, which communicate with one another via radio frequency waves. Recently, RFID has been widely used in many applications, such as transport systems, electronic ticketing, access control, animal identification, logistics, and supply chain management [1].

In the application, where many tags are present in the reader's field, if more than one tag communicates with a reader at the same time, a *collision* will occur resulting in the failure of that communication. Thus, each tag has to re-send all information to the reader. To prevent this problem, an anti-collision algorithm must be used. Based on the International Standards Organization (ISO) and EPCglobal (EPC), there are 3 types of anti-collision algorithms, namely, binary tree (BT) [, Framed Slotted ALOHA (FSA), and Dynamic Framed Slotted ALOHA (DFSA)algorithms [2-4]. However, these algorithms take a lot of time to identify tags [2].

Many improved anti-collision algorithms have recently been proposed in the literature. For example, Cheng and Jin [2] presented the analysis and simulation of several RFID anti-collision algorithms and partitioning of tags for near-optimum RFID anti-collision performance. Shin and Kim [5] proposed a partitioning technique, which enables a faster accurate estimation on the number of contending tags, and yields much higher throughput against previous non-partitioning approaches. Cho *et al.* [6] proposed an anti-collision algorithm using parity bit (ACPB) in RFID system. The ACPB identifies tags without checking all bits in the tags. Then, the reader uses the parity bit, which is added to the tag's ID number. Clearly, ACPB can reduce the number of the requests from the reader. Thus, it can shorten the time of identifying all tags in the reader's field. In this paper, we propose a novel anti-collision algorithm, which is based on the BT and DFSA algorithms. The proposed algorithm can estimate the number of tags in the reader's field and identifies all tags faster than the existing anti-collision algorithms.

The rest of this paper is organized as follows. Section II briefly describes how BT and DFSA algorithms work. A new anti-collision algorithm is explained in Section III. Section IV compares the performance of different anti-collision algorithms. Finally, Section V concludes this paper.

II. Existing Anti-Collision Algorithms

This section briefly describes how BT and DFSA perform because their performances are compared with the proposed anti-collision algorithm.

A. *Binary Tree* (BT)

The BT algorithm or the Query Tree algorithm divides tags into two groups based on the most significant bit (MSB) of the tag's ID number, which consists only of bits "0" and "1"[6]. To search a tag, a dividing process continues adding up the number "0" and "1" into each group, until finding a tag [2, 7, 8]. Note that we consider only the case where the tags do not support a random generator in hardware for group selection, meaning that the BT algorithm operates on the tag's identification (ID) numbers [9].

To obtain all tags, the reader begins a search by sending a prefix bit "0" or "1" to all tags and waits for the response. If there is only one response, the reader then can identify that tag. However, if more than one tag responds back at the same time, a collision will occur. In this case, the reader will add another bit ("0" or "1") to a prefix bit and send the new prefix bits to the remaining tags until there is only one response. The reader will do this process until all tags are identified.

To compare the performance of different anti-collision algorithms, we count the required number of commands sent from the reader to the tag as a criterion. Each command is referred to as one *time slot* or slot. . Assuming that each slot

978-1-4244-3388-9/09/$25.00 ©2009 IEEE

uses the same processing time, the algorithm that requires more number of slots will operate slower.

B. *Dynamic Framed Slotted ALOHA* (DFSA)

Dynamic Framed Slotted ALOHA developed from FSA is utilized in Class 1 Generation 2 of EPC [4]. It divides tags into many groups according to the number of slots specified by a reader. Each tag will random a slot number between 0 and the maximum number of slots, and the tags having with the same slot number will be in the same group.

First, the reader sends a command with a parameter called "slot_number." Note that the "slot_number" will be set to 0 at first, and it will then increase by 1 for every round. If the tag has a group number equal to the "slot_number," that tag will respond to the reader. Then, if there is only one response at this time, the reader will identify that tag. If there is a collision, the reader will increase the "slot_number" by 1 and send it to all remaining tags. The reader repeats this process until the "slot_number" is equal to the maximum number of slots.

When the reader finishes sending a command with the "slot_number" between 0 the maximum number of slots, we assume that the operation time is one frame. If the reader cannot identify all tags in the reader's filed, the reader will begin the new frame. The reader can adjust the number of slots in the new frame based on a Q-parameter [4, 5]. The reader will do this process until it can identify all tags in the reader's field.

III. Proposed Anti-Collision Algorithm

Reference [10] studied the condition that the BT algorithm is more efficient than FSA and DFSA because the BT algorithm uses a less number of slots when the number of tags in the system is small. Practically, when the system has a large number of tags, the BT algorithm tends to perform worse because it uses a lot of slots to identify all tags if compared to DFSA [10].

The proposed algorithm is developed based on the BT and the DFSA algorithms. The existing algorithm is illustrated in Fig.1. We first divide tags into many groups using the DFSA algorithm as illustrated in Fig. 2. Then, all tags in each group are identified using the BT algorithm. To achieve this, we assume that the tag can generate a 9-bit uniform random number and has a function to select a group according to that random number. To make the proposed algorithm more efficient, the number of groups must coincide with the number of tags. Specifically, the less the number of tags, the less the number of groups. Therefore, we must first estimate the number of tags in the reader's field so as to determine the number of groups used in the proposed algorithm. To do this, we use the number of tags in each group to estimate the total number of tags in the reader's field since each group should have an equal probability to have the same number of tags.

Fig.3 shows how the proposed anti-collision algorithm works. We determine the number of groups from the estimated total number of tags in the reader's field. If the estimated number of tags is within the range of 0 to 1,000 tags,

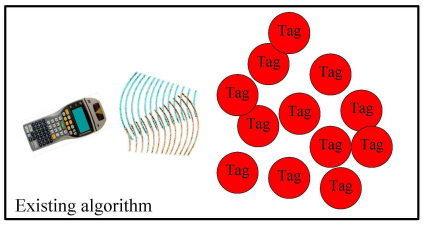


Figure 1. How the existing algorithm works.

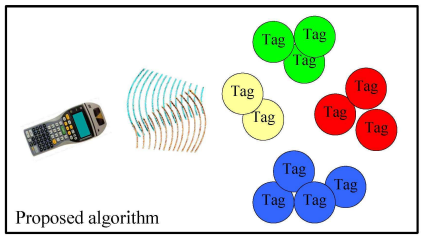


Figure 2. How the proposed algorithm works.

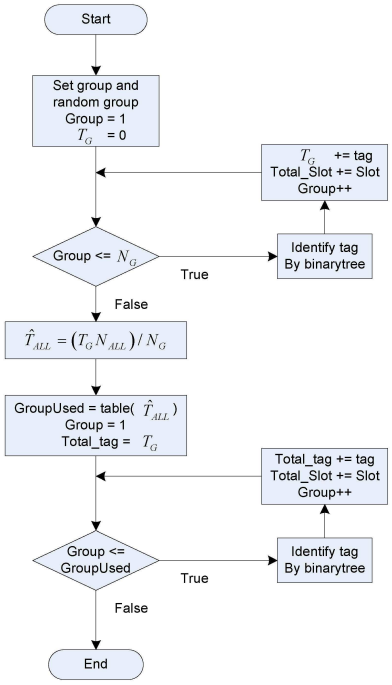


Figure 3. A flowchart of the proposed anti-collision algorithm.

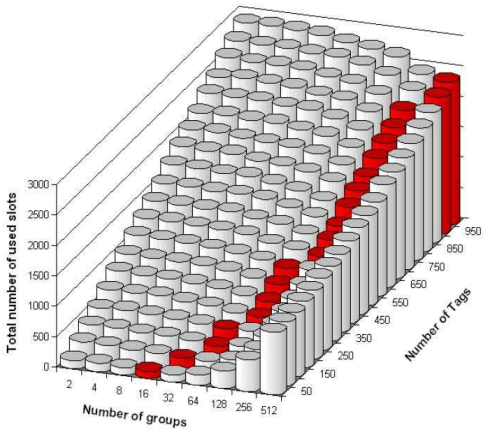


Figure 4. The number of used slots for different number of tags and groups.

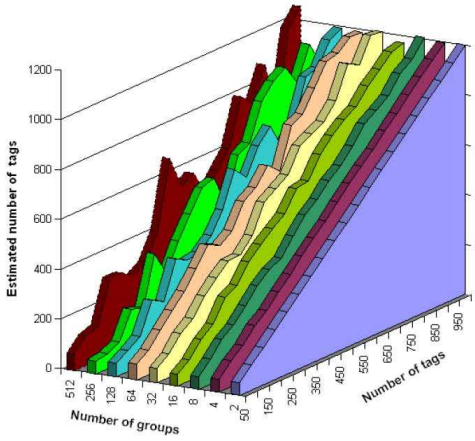


Figure 5. The estimated number of tags for different number of tags and groups (for $N_G = 3$).

we recommend the number of starting groups at 32 groups (N_{ALL} = 32). Next, we randomly pick three groups out of 32 groups in order to identify tags based on the BT algorithm. Then, the total number of tags in the reader's fields can be estimated according to

$$\hat{T}_{ALL} = (T_G N_{ALL}) / N_G \quad (1)$$

where $\hat{T}_{ALL}$ is the estimated total number of tags in the reader's field, T_G is the number of identified tags in the selected three groups, N_G is the number of selected groups used to find $\hat{T}_{ALL}$ (e.g., N_G = 3), and N_{ALL} is the total number of groups in the reader's field (e.g., N_{ALL} = 32).

Once we have an estimate of the total number of tags in the reader's field, we can now choose the suitable number of groups to identify tags. Then, we use a regular BT algorithm to identify tags in each group with the assumption that the tag's ID number consisting of 64 bits (all random bits).

IV. Simulation Result

Fig. 4 shows the total number of used slots to identify all tags for different number of tags and groups, where the x-axis represents the number of groups, the y-axis indicates the number of tags, and the z-axis represents the number of used slots. It is a practical criterion to identify the algorithm performance because the less the number of used slots, the faster the algorithm. It is apparent that for a given number of tags, there is the suitable number of groups (i.e., the shaded columns) that yields the lowest number of used slots. Therefore, the proposed algorithm must first estimate the total number of tags in the reader's field so as to determine the suitable number of groups.

Fig. 5 illustrates the estimated number of tags for different number of tags and groups, where the x-axis represents the number of groups, the y-axis indicates the number of tags, and the z-axis represents the estimated number of tags. Clearly, the less number of groups will result in a better estimation of the total number of tags. For example, the number of groups of 2 will give 100% accuracy of the estimated total number of tags. However, based on exhaustive search, we found that the number of groups of 32 is the maximum number of groups, which yields minimum error of the estimation under specified condition. For example, for N_G = 3 and N_{ALL} = 32, the total number of tags from 0 to 200 tags will give an error of 31% - 37%, but for N_G = 31 and N_{ALL} = 32, the total number of tags from 0 to 200 tags will give an error of 0.05% - 0.15%. Thus, the chosen parameter for N_G will depend strongly on the error threshold requirement.

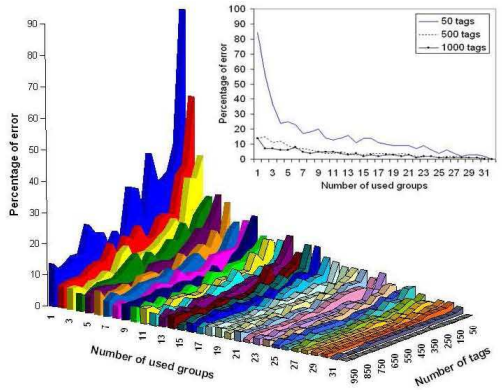


Figure 6. The percentage of error between the actual number of tags and the estimated number of tags (for N_{ALL} = 32).

Fig. 6 compares the percentage of error between the actual number of tags and the estimated number of tags obtained

from our proposed method, where the x-axis represents the number of used groups for estimating tags, the number of used groups for estimating tags, the y-axis indicates the number of actual tags, and the z-axis represents the percentage of error. We first set the total number of groups of 32 (i.e., N_{ALL} = 32). Then, we vary the number of used groups from 1 to 32 (i.e., N_G = 1 to 32) so as to estimate the total number of tags in the reader's field. If we use a large number of used groups, the estimation error will be small, but the proposed algorithm will require a lot of number of used slots, which implies low efficiency. Conversely, if we use a small number of used groups, the estimation error will be large, resulting in unacceptable estimate. Based on Fig. 6, we set the number of used groups to be 3 because if the larger number of group is utilized, the number of used slots will significantly increase even though the percentage of error between the estimated tags and the actual tags is decreased.

In this paper, we compare the performance of the four algorithms, namely, Binary Tree, Binary Tree 3 bits, DFSA, and the proposed algorithm (with 32 groups), assuming that the tag's ID number consists of 64 bits (all random bits).

Fig. 7 illustrates the performance comparison as the plot between the number of tags (x-axis) and the total number of used slots (y-axis). The smaller the number of used slots, the faster the algorithm. The proposed algorithm outperforms the other algorithms, i.e., at the considering total number of used slots, the proposed algorithm uses a smaller number of tags. The advantage of the proposed algorithm is more visible as the increase of the number of tags and could be explained as follow. The DFSA divides groups of tags into slots randomly. Thus, tags are more likely to collide especially when a large number of tags are presented in the reader's field. While in the case of BT and BT 3-bit, the more numbers of tags presented in the reader's field, the more identical of the most significant bit ID of the tags. Therefore, more collisions occur resulting in higher used slots.

V. Conclusions

The anti-collision algorithms are crucial to the application that uses a large number of tags. In general, the BT algorithm performs well when number of tags present is small while the DFSA algorithm performs well when the environment of tag present is dynamic. The proposed algorithm exploits the advantage of both the BT and the DFSA algorithms. Specifically, all tags are divided into many groups based on the DFSA algorithm, and the tags in each group are identified using the BT algorithm. It is clear from simulation that the proposed anti-collision algorithm performs better than the existing ones in terms of the number of used time slots, which implies faster identification process.

Acknowledgment

This work is supported by National Science and Technology Development Agency (NSTDA) and the RFID Program, National Electronics and Computer Technology Center (NECTEC), Thailand, under grant TG-44-21-50-098M.

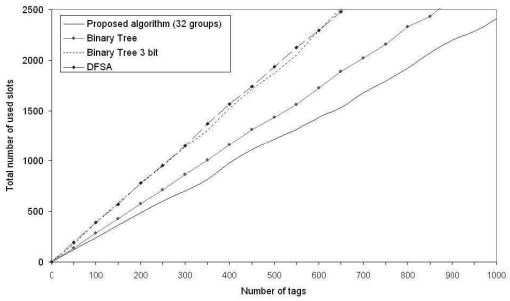


Figure 7. Performance comparison of different anti-collision algorithms.

References

[1] K. Finkenzeller, *RFID handbook*, John Wiley & Sons, West Sussex, 2003.

[2] T. Cheng and L. Jin, "Analysis and Simulation of RFID Anti-collision Algorithm," *IEEE Advanced Communication Technology*, vol. 1, pp. 697 – 701, Mar. 2007.

[3] EPC Global. 860MHz~930MHz Class I Radio Frequency Identification Tag Radio Frequency & Logical Communication Interface Specification Candidate Recommendation, Version 1.0.1.

[4] EPC Global. EPC™ Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz~960MHz, Version 1.0.9.

[5] W. J. Shin and J. G. Kim, "Partitioning of Tags for Near-Optimum RFID Anti-collision Performance," *IEEE Wireless communications and Networking Conference*, pp. 1673-1678, Mar. 2007.

[6] J. S. Cho, J. D. Shin and S. K. Kim, "RFID Tag Anti-Collision Protocol: Query Tree with Reversed IDs," *ICACT*, pp. 225-230, Mar. 2008.

[7] C. Abraham, V. Ahuja, A. K. Ghosh, and P. Pakanati, "Inventory Management using Passive RFID Tags: A Survey," Department of Computer Science thesis, University of Texas at Dallas, Richardson, Texas.

[8] R. Ahmed, "Performance Comparison of RFID Tag Anti-collision Algorithm using Simulation and Real Testing Based," M. Eng. thesis, Asian Institute of Technology, Thailand, May.2007.

[9] ISO/IEC 18000-6:2003(E), Part 6: Parameters for air inter-face communications at 860-960 MHz, Nov. 26, 2003.

[10] S. Makwimanloy, P. Kovintavewat, U. Ketprom, C. Tantibundhit and C. Mitrpant, "A New Anti-Collision Based on A-Priori Information," in *Proc. of ECTI-CON* 2008, Krabi, Thailand, vol. II, pp. 733 – 736, May 14 – 16, 2008.