이차원 Data Matrix 바코드에서 Base 256 모드의 디코딩 알고리즘

한희준[1], 이효창[1], 이종연[2]
[1]충북대학교 컴퓨터교육과, [2]충북대학교 디지털정보융합과

Algorithm of Decoding the Base 256 mode in Two-Dimensional Data Matrix Barcode

Hee June Han[1], Hyo Chang Lee[1], and Jong Yun Lee[2]
[1]Department of Computer Education, Chungbuk National University
[2]Department of Digital Informations and Convergence, Chungbuk National University

요 약 기존의 바코드는 정보 배열이 나란히 나열된 선 모양을 가지며 이를 1차원 바코드라 부른다. 이에 반해 2차원 바코드는 점자방식 또는 모자이크방식 코드로 작은 정사각형 도는 직사각형 안에 정보를 표현한다. 2차원 바코드는 기존의 1차원 바코드보다 작은 공간에 많은 데이터를 표현 가능함으로써 보다 효율적인 바코드의 구현이 가능하다. 현재 ISO 국제 표준화된 2차원 바코드는 총 4가지로 분류되는데 QR Code, Data Matrix, PDF417, MaxiCode가 있다. 본 논문에서는 ISO 국제 표준화된 바코드 중 하나인 Data Matrix의 Base 256 모드에 대한 기본 개념, 구성 방법, 인코딩 및 디코딩 방법을 상세히 제안한다. Data Matrix 심벌에 저장된 데이터를 보다 효율적으로 구성하기 위해 숫자, Alphanumeric 문자, 이진법에 따라 다른 인코딩, 디코딩 방법을 사용하게 되는데 본 논문에서는 이를 고려한 디코딩 방법에 초점을 맞춰 기술할 것이다.

• **주제어** : 2차원 바코드, 데이터측정, 인코딩, 디코딩, 베이스256

Abstract Conventional bar code has the appearance of line bars and spaces, called as one-dimensional bar code. In contrast, the information in two-dimensional bar code is represented by either a small, rectangular or square with the types of mosaic and Braille. The two-dimensional bar code is much more efficient than one-dimensional bar code because it can allow to store and express large amounts of data in a small space and so far there is also a little information about decoding the Data Matrix in base 256 mode. According to the ISO international standards, there are four kinds of bar code: QR code, Data Matrix, PDF417, and Maxi code. In this paper, among them, we focus on describing the basic concepts of Data Matrix in base 256 mode, how to encode and decode them, and how to organize them in detail. In addition, Data Matrix can be organized efficiently depending on the modes of numeric, alphanumeric characters, and binary system and expecially, we focus on describing how to decode the Data Matrix code by four modes.

• **Key Words** : 2D Barcode, Data Matrix, Encoding, Decoding, Base256

*교신저자: 이종연(jongyun@chungbuk.ac.kr)
*본 논문은 미래창조과학부 및 정보통신산업진흥원의 산학협력 특성화 지원사업의 연구결과로 수행되었음(NIPA-2013010918).
접수일 2013년 8월 21일 수정일 2013년 9월 6일 게재확정일 2013년 9월 9일

1. 서론

바코드는 1차원 바코드와 2차원 바코드로 분류된다. 유통 부문에서 흔히 볼 수 있는 검정색 선(line)과 흰색여백의 형태가 1차원 바코드이다. 2차원 바코드는 점자방식 또는 모자이크방식 코드로 정사각형 안에 정보를 표현한다. 선과 여백의 굵기에 따라 가로 방향으로만 정보를 표현하는 1차원 바코드와는 달리, 2차원 바코드는 가로 세로 양방향 모두 정보를 표현할 수 있다. 따라서 2차원 바코드는 기존 1차원 바코드의 빈약한 정보용량에 비해 100배 이상의 고밀도 정보 저장이 가능하다. 또한 정보가 훼손되어도 상당부분 복구가 가능하며 어느 방향에서나 인식이 가능하다는 것이 장점이다. 현재 ISO 규격화되어 쓰이고 있는 바코드는 4가지로서 QR코드, Data Matrix, PDF417, MaxiCode가 있다. 본 논문에서는 Data Matrix를 소개하고 Data Matrix의 여러 가지 인코딩과 디코딩 방식 중 Base256방식에 대해 중점적으로 기술하고자 한다.

Data Matrix 코드는 규격화된 정사각형이나 직사각형 크기의 데이터 코드라고도 불리는 심벌로지로 구성되어 있다. 2차원 바코드 중에서도Data Matrix는 최대 3116개의 숫자, 2335개의 Alphanumeric 문자 그리고 1555바이트의 이진법 정보까지 인코딩 및 디코딩이 가능한 높은 밀도의 2차원 바코드이다. 또한 코드의 물리적 손상을 복구하기 위한 나선형(Convolutional)과 Reed-Solomon 유형의 오류정정 및 수정기능을 포함한다. [그림1]은 Data Matrix의 사용 예로 제품 박스와 물품의 라벨에 이용하는 예이다.

[Fig. 1] Example of Data Matrix

Data Matrix의 심벌은 2가지 기본 형태로 제공된다. 첫 번째 형태는 10x10에서 144x144사이의 지정된 모듈(Module) 개수에 맞춘 정사각형 형태로 제공되며, 두 번째 형태는 8x16, 16x48의 지정된 모듈 개수를 가진 직사각형 형태로 제공된다. 심벌은 사각형의 도트들로 구성되는데 그 사각형 도트의 크기인 모듈 또한 0.001 * 0.001 인치(inch)부터 14*14인치까지 사용자 지정이 가능하다. 모듈의 사이즈 조절에 따라 심벌의 크기도 사용자가 임의로 지정이 가능하다.

Data Matrix는 또한 오류정정 및 수정기능에 따라 2가지로 크게 분류된다. 나선형(Convolutional) 오류정정 및 수정기능을 가진 ECC 000-140과 Reed-solomon 오류정정 및 수정기능을 가진 ECC 200으로 분류된다. 하지만 본 논문에서는 현재 가장 일반적으로 사용되고 있는 ECC 200을 기준으로 오류정정 및 수정기능을 소개할 것이다. 또한 ECC 200이 적용된 Data Matrix의 인코딩 방식은 [표 1]과 같다.

[Table 1] Latch Codeword [1]

Codeword	Data of function
1 - 128	ASCII data (ASCII value + 1)
129	Pad
130 - 229	2-digit data 00-99 (Numeric Value + 130)
230	Latch to C40 encodation
231	**Latch to Base 256 encodation**
232	FNC1
233	Structured Append
234	Reader programming
235	Upper Shift (shift to Extended ASCII)
236	05 Macro
237	06 Macro
238	Latch to ANSI X 12 encodation
239	Latch to Text encodation
240	Letch to ECIFACT encodation
241	ECI Character
242 - 255	Not to be used in ASCII encodation

여기서 자주 사용하는 인코딩과 디코딩 방식으로는 ASCII, C40, Text, EDIFACT(Base64), Base 256이 있으며 ASCII, C40, Text, EDIFACT(Base64) 인코딩과 디코딩 방식은 소개된 바 있지만 Base256 과정은 아직까지 별도로 소개된 바가 없다. 또한 Data Matrix의 Base 256 모드의 디코딩은 다음과 같은 이점을 제공한다. 첫째 Base 256 인코딩과 디코딩 기법은 널리 쓰이는 Data Matrix의 ASCII, TEXT, C40, X12, Base 64 인코딩, 디코딩방법과 달리 코드를 전부 디코딩하지 않아도 변환

* ISO(International Organization for Standardization): 나라마다 다른 공업규격을 조정, 통일하고 물자 및 서비스의 국제적 교류를 원활히 하기 위한 국제기구

코드워드 다음 1개 또는 2개의 코드워드들만을 가지고 인코딩 그리고 Base 256으로 디코딩을 해야 하는 길이를 사전에 알 수 있다는 장점이 있다. 둘째, 이로 인해 Base 256으로 인코딩과 디코딩을 해야 하는 부분을 사전에 미리 추출 인지하고 정확하게 인코딩과 디코딩길이에 맞춰 작업이 이루어지도록 할 수 있다. 이로 인해 디코딩에 필요한 부분만 추출을 하여 작업을 수행할 수 있다.

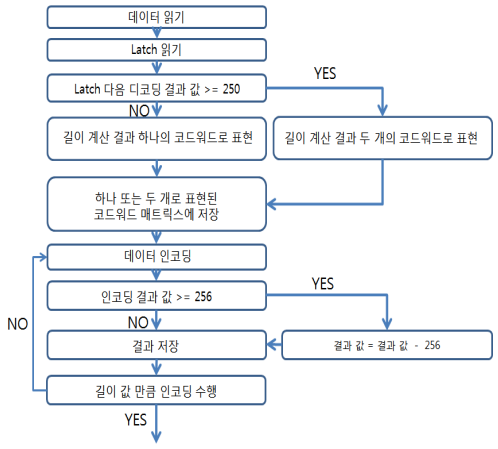
따라서 본 논문에서는 Data Matrix의 Base256 모드에 대한 인코딩과 디코딩 방식과 알고리즘을 상세히 제안하고자 한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 ECC 200이 적용된 Data Matrix 이차원 바코드의 여러 인코딩과 디코딩 방식 중 Base 256의 인코딩 알고리즘과 인코딩 방식을 예를 들어 기술한다. 3장에서는 Base 256의 디코딩 알고리즘과 디코딩 방식을 예를 들어 기술한다. 그리고 4장에서 결론을 요약할 것이다.

2. Base 256의 인코딩 방법

Base 256 인코딩 과정은 [표 1]에 있는 ASCII, C40, TEXT, X12 그리고 EDIFACT(Base 64)의 인코딩 방식과 다르게 길이를 지정해 주는 코드워드 값이 있으며 인코딩된 데이터의 길이를 나타내는 코드워드는 데이터의 길이에 따라 1개 또는 2개의 코드워드를 이용하여 나타내며 데이터의 길이가 '250'개보다 크거나 같을 경우에 2개의 코드워드를 이용하여 나타낸다. 코드워드의 길이를 나타내는 1개 또는 2개의 코드워드는 Base 256 인코딩의 시작을 나타내는 Latch(변환) 코드워드인 10진수 '231' 바로 다음에 나타난다.

[그림 2]는 Base 256 모드의 인코딩 알고리즘이며, 그 처리과정은 다음과 같다. 첫째, 인코딩할 데이터를 읽는다. 둘째, Latch 코드워드를 읽는다. 셋째, Latch 다음 코드워드의 결과 값에 따라 다르게 처리된다. 먼저 읽은 데이터 길이가 '250' 미만이면 하나의 코드워드로 저장된다. 또한 읽은 데이터 길이가 '250'이상 이면, 두 개의 코드워드로 저장된다. 넷째, 인코딩 결과값에 따라 다르게 처리된다. 값이 '256'이상일 경우 그 값에서 '255'를 뺀 값을 저장하고 그렇지 않을 경우 바로 저장한다. 마지막으로 인코딩할 길이만큼 인코딩 했을 경우 Base 256 인코딩을 종료 하고 그렇지 않을 경우 반복 시행 한다.



[Fig. 2] Base 256 Encoding Algorithm

Base 256으로 인코딩할 경우 $Randomizenumber$를 생성하며, [식1]에 $CodewordPosition\,Value$에 코드워드의 순번을 대입한다. 세 번째 코드워드 자리에 Base 256으로 인코딩된 값이 들어가게 되거나 길이를 나태는 코드워드가 들어가게 될 경우 $CodewordPosition\,Value$ 자리에 세 번째 순번을 나타내는 숫자 '3'을 네 번째 코드워드 자리에 들어가게 될 경우 $CodewordPosition\,Value$ 자리에 네 번째 순번을 나타내는 숫자 '4'를 대입한다.

$$Randomizenumber=$$
$$((149\times CodewordPosition\,Value)\,\%\,255)+1$$
$$(식1)$$

예로, 만약 변환 코드워드를 나타내는 두 번째 코드워드 다음인 세 번째 코드워드 자리에 인코딩된 데이터 길이의 값을 넣고 인코딩 데이터를 구하게 될 경우 (식1)을 이용해 먼저 코드워드의 순번 값을 이용하여 $Randomizenumber$를 구할 수 있다. 즉,

$$Randomizenumber=((149*3)\,\%\,256)+1$$

그리고 (식2)를 이용하여 인코딩할 데이터를 계산할 수 있다. 먼저 인코딩 데이터 길이를 계산하고, 앞에서 구한 $Randomizenumber$를 (식2)에 대입하여 인코딩 데이터($Encodingdata$)를 계산할 수 있다.

$$Encodingdata = Data\,Length + Randomizenumber \quad (식2)$$

또한 위 데이터를 인코딩할 경우 (식2)에서 *DataLength*의 자리에 인코딩하고자 하는 데이터가 나타내는 10진수 값을 ASCII테이블에서 찾아 10진수 값을 *DataLength*자리에만 대입하면 된다.

예를 들어, 순번이 3인 세 번째 코드워드에 값이 '6'인 숫자데이터를 넣는 경우 (식2)를 이용하기 전에 먼저 ASCII테이블에서 숫자 '6'이 나타내는 10진수 값을 찾아보면 '54'임을 알 수 있다. (식2)를 이용하기 위해서는 먼저 (식1)을 이용하여 *Randomizenumber*를 구해야 하는데 순번이 '3'이므로 (식1)의 *CodewordPosition Value*에 '3'을 입력하면 '193'이라는 *Randomizenumber*값을 구한다. (식2)에 *Randomizenumber*과 ASCII에서 값 '3'에 대한 십진수 '54'을 더하면 $54+193=247_{(10)}=1111\,0111_{(2)}$ 이 된다.

위 경우는 인코딩된 데이터 길이가 250개 미만이라는 가정 하에 식을 적용한 사례이다. 250개 이상의 데이터가 인코딩될 경우에는 위 식(1)을 수행하고 식(2)하기 이전에 한 단계를 추가적으로 더 수행을 해주어야 한다. 앞에서 설명했지만 250개 이상일 경우에는 코드워드 2개를 이용하여 값을 표현한다. 앞서 설명한 것과 같이 2개의 코드워드를 이용하여 이를 나타내게 되는데 이때 (식3)을 사용한다.

$$First\,Codeword = (Data length / 250) + 249$$
$$Second\,Codeword = Data length \,\%\, 250 \quad (식3)$$

예로, 인코딩할 데이터 길이가 250개라고 할 경우, 위 (식3)을 이용해 값을 구하면 다음과 같이 250과 0결과가 산출된다.

$$First\,Codeword = (250/250) + 249 = 250$$
$$Second\,Codeword = 250 \,\%\, 250 = 0$$

이렇게 2개의 코드워드를 이용해 값을 나타낼 경우 (식1)과 (식2)를 두 번 사용해야 한다.

먼저 길이를 계산해서 얻은 '250'을 이용하여 (식2)의 계산결과를 얻고자 한다. 여기서 코드워드의 순번을 '2'로 가정할 경우 (식1)에 대입하여 *Randomizenumber*을 구하면 '44'라는 값이 계산된다.

$$Randomize\,Nmber = ((149*2)\,\%\,255)+1 = 44$$

여기에 길이를 나타내는 첫 번째 코드워드 값 '250'을 (식2)에 대입하여 결과를 구하면 '294'라는 값이 나온다.

$$Encoding\,Data = 250+44 = 294$$
$$[First\,Codeword = 250]$$

그런데 여기서 (식2)를 이용해 나온 결과 값이 '255'를 초과하는 경우, (식4)를 이용하여 값에 변화를 주어야 한다.

$$IF(Encoding Data \geqq 256)\,then$$
$$Encoding Data = Encoding Data - 256; \quad (식4)$$

(식2)를 이용하여 나온 결과 값이 '255'를 초과하기 때문에 (식4)를 이용하여 값에 변화를 준다. 즉,

$$IF(294 \geqq 255)$$
$$294-255 = 38$$

이유는 Data Matrix에서 하나의 코드워드는 8비트로 표현이 되는데 8비트로 표현할 수 있는 값은 '255'가 최대 값이기 때문이다.

(식3)의 첫 번째 식을 이용해서 나온 값으로 첫 번째 코드워드에 저장하게 될 값은 '38'이 되며 (식 3)의 두 번째 식을 이용하여 나온 값을 두 번째 코드워드에 저장될 값은 '193'이 된다.

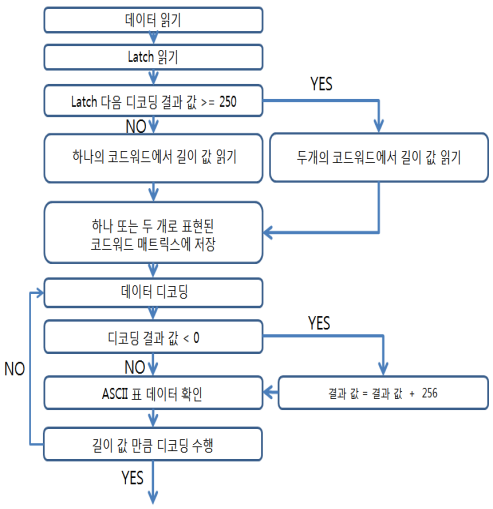
$$Randomize\,Nmber = ((149*3)\,\%\,255)+1 = 193$$
$$Encoding\,Data = 0+193 = 193$$
$$[Second\,Codeword = 0]$$

길이를 나타낸 이후부터는 (식1)의 *Codeword*

*Position Value*에는 순번을 (식2)의 *Data Length*에는 데이터가 ASCII테이블에서 나타내는 10진수 값을 입력하여 계산한 결과 값을 이진수로 변형하여 해당 순번의 코드워드에 저장한다. 이를 데이터의 길이만큼 반복해서 시행해 인코딩 한다.

3. Base 256의 디코딩 방법

Base 256의 디코딩 과정은 Data Matrix의 다른 디코딩 모드들과 동일하게 인코딩 과정을 거꾸로 수행한다.



[Fig. 3] Base 256 Decoding Algorithm

[그림 3]은 Data Matrix에서 Base 256 모드의 디코딩 과정이다. 첫째, 디코딩할 데이터를 읽는다. 둘째, 디코딩하는 과정에서 [표 1]에서 확인했던 Latch to Base 256에 해당하는 십진수 값 '231'이 나올 경우 이 다음 코드워드부터 Base 256 모드로 디코딩 한다. 셋째, Latch 코드워드 다음에 오는 코드워드를 디코딩해 길이를 알아낸다. 먼저 디코딩한 값이 '250'미만일 경우 하나의 코드워드로부터 길이 값을 읽어 온다. 하지만 '250'이상일 경우 두 개의 코드워드를 읽는다. 넷째, 디코딩을 하다 결과 값이 '0'보다 작을 경우 '256'을 더해주며 그렇지 않을 경우 ASCII 테이블로부터 데이터 값을 저장한다. 다섯째, 마지막으로 인코딩된 데이터 길이만큼 디코딩이 됐는지를 확인하고 그렇지 않을 경우 디코딩을 반복해서 수행한다.

Base 256의 상세 디코딩 과정은 앞의 2절 인코딩 방법에서 보았던 수식들에 대해 거꾸로 수행하면 된다. (식1)에서 사용되었던 수식은 디코딩 과정에서도 똑같이 사용되며 (식2)의 수식은 디코딩 과정에서는 (식5)을 이용한다.

$$DataLength = ScannedData - Randomizenumber \quad (식5)$$

ASCII 모드로 디코딩 수행 중 Base 256 Latch(변환) 코드워드인 십진수 '231'을 만나면 다음에 나올 한 개 또는 두개의 코드워드는 Base 256으로 인코딩된 데이터 길이를 나타낸다. 여기서 만약 Latch(변환) 코드워드 다음에 나오는 코드워드의 디코딩된 10진수 값이 '250'미만일 경우 *DataLength* 값이 하나의 코드워드로 나타내어진 것을 의미하며, Latch(변환) 코드워드의 다음 코드워드 10진수 값이 '250'보다 크거나 같을 경우 *DataLength* 값은 두 개의 코드워드로 나타낸 것을 의미한다. 후자의 경우 길이를 나타내는 두개의 코드워드의 디코딩된 10진수 값을 더해줘야 Base 256 모드로 디코딩된 길이를 알 수 있다.

예로, Latch(변환) 코드워드 다음 코드워드의 10진수 값이 '47'이고 *CodewordPosition Value* 값이 '2'라고 하자. *CodewordPosition Value* '2'를 (식1)을 통해 구한 $((149*2)\%255)+1=44$ 이고 (식5)를 이용해 코드워드의 10진수 값을 *ScannedData*에 대입시켜 나온 값 $47-44=3$ 이다. 디코딩된 값 '3'이 250보다 작기 때문에 이는 '3'개의 데이터가 Base 256 인코딩 방식으로 인코딩 됐다는 것을 의미한다. 만약 (식5)의 수행결과 값이 '250'이상이면 그 다음번 코드워드도 (식1)과 (식5)를 거쳐 디코딩 값을 구하고 이 값을 앞에서 구한 '250'이상 값에 더해주어 Base 256 으로 디코딩해야하는 총 데이터의 길이를 구한다. 이렇게 길이를 나타내는 코드워드 다음부터는 순번을 (식1)대입해 구한 결과 값 *Randomiznumber*를 해당 순번 코드워드의 10진수 값을 (식5)의 *ScannedData*에 대입시켜 반복 시행하여 디코딩한다.

앞에서 보았던 *CodewordPosition Value* 값이 '2'였던 예에서 (식5)의 결과 값이 '3'이 나왔기 때문에 세 번째 코드워드부터 다섯 번째 코드워드까지 총 3개의 코드워드는 원래 데이터 값들이 저장되어 있으며 Base 256 모드로 디코딩하면 인코딩되기 이전 원래 데이터 값이 나오게 된다.

인코딩되기 이전의 원래 데이터 값은 (식5)에서 *Data Length*를 구하는 것과 같은 방식으로 계산한다.

예로, 세 번째 코드워드가 나타내는 10진수 값이 '242'이고 *CodewordPosition Value* 값이 '3' 이라고 가정하자. 먼저 *Randomizenumber*값을 구한다. 풀이 과정을 살펴보면 *CodewordPosition Value* 값은 '3'이므로 (식1)에 대입하면 '193'이라는 값을 얻을 수 있다. 즉,

$$Randomize\ Number = ((149*3)\%255)+1 = 193$$

세 번째 코드워드의 10진수 값이 '242'이라 할 경우 (식5)에 대입하고 *Randomizenumber*값 193을 빼주면 (242 − 193 = 49), 해당 코드워드가 Base 256으로 인코딩되기 이전의 데이터 값인 '49'가 나타나게 된다. 이렇게 나온 '49'를 ASCII 표에서 십진수 '49'를 찾으면 숫자 '1' 값이 나오며 이 값이 Base 256을 이용하여 인코딩되기 이전의 원래 데이터이다. 만약 결과값이 음수값이 나오면 (식6)을 이용하여 '256'이라는 숫자를 더해주고 결과값을 ASCII 표에서 찾아보면 인코딩되기 이전의 원래 데이터를 알 수 있다.

$$IF(Data\ Length < 0)\quad Data\ Length = Data\ Length + 256 \quad (식\ 6)$$

예로, *CodewordPosition Value*값이 '5'이고 다섯 번째 코드워드를 10진수로 표현했을 때 값이 '31'이라고 하자. *CodewordPosition Value* '5'를 (식1)에 대입하면, *Randomizenumber* = ((149 * 5) % 255) + 1 은 '236'이 된다. 다섯 번째 코드워드를 십진수로 나타낸 값 '31'을 (식5)의 *ScannedData*에 대입하여 구한 값은 31 − 236 = −205 이다. 이와 같이 (식5) 결과값이 음수이면 2장의 (식4)와 같이 유사한 과정을 거쳐야 하며 이를 (식6)으로 표현한다. (식6)의 과정을 거치면 -205 + 256 = 51 이라는 십진수 값이 나오며 이 결과를 ASCII 테이블에서 찾아보면 숫자 '3'을 나타낸다. 이 값이 Base 256으로 인코딩되기 이전의 원래 데이터 값이다. 네 번째와 다섯 번째 코드워드 또한 같은 방식으로 디코딩하면 총 데이터의 값이 모두 계산되며 여섯 번째 코드워드부터는 ASCII 디코딩 방식으로 다시 디코딩한다. 여기서 중요한 부분은 디코딩 과정에서 변환 코드워드 다음에 디코딩한 코드워드의 값이 '250'이상일 경우 바로 다음번 코드워드의 디코딩 값을 더해줘야 한다. 이렇게 더해준 결과 값이 Base 256으로 디코딩된 총 데이터의 길이를 나타낸다. 디코딩 과정에서는 코드워드의 순번 값(*CodewordPosition Value*)을 (식1)에 입력하여 *Randomizenumber*값을 구한 뒤 그 순번에 해당하는 코드워드가 나타내는 값에서 *Randomizenumber*를 빼준 값이 인코딩되기 이전 값이며 만약 음수가 나왔을 경우 (식6)을 이용하여 양수로 변환을 해줘야 한다.

4. 결론

본 논문에서는 ECC 200이 적용된 Data Matrix의 인코딩과 디코딩 방법 중 하나인 Base 256 코드워드 인코딩과 디코딩 기법에 대해 제안하였다. 따라서 본 논문의 연구결과는 다음과 같은 이점을 제공할 것이다. 첫째, 기존에 나와 있는 논문들은 Base 256을 제외한 ASCII, Text, C40, X12 그리고 EDIFACT(Base 64)에 관한 논문이 대부분이다. 이 때문에 사용자가 Base 256을 이해하는데 있어서 많은 어려움이 있었지만 이 기법에 대해 제안함으로서 Base 256의 흐름과 원리를 이해하는데 도움이 될 것이다. 둘째, Data Matrix는 큰 데이터들을 보다 효율적으로 저장하기 위해 인코딩과 디코딩시에 여러 가지 모드를 호환하며 사용하는데 본 논문에서 제안한 Base 256 모드 또한 이러한 모드들 중 하나로써 큰 데이터 저장 시 다른 모드들과 호환이 되어 가장 효율적인 Data Matrix 구성에 도움이 될 것이다. 끝으로 향후 연구과제로는 숫자, Alphanumeric문자, 이진법 이외에 한글도 입력이 가능한 코드워드 기법이 없는데 숫자와 영문자뿐만이 아닌 한글 데이터도 입력이 가능한 인코딩 및 디코딩 방식이 만들어져야 할 것이다.

REFERENCES

[1] ISO/IEC FCD 16022 Information technology - Automatic identification and data capture techniques - Bar code symbology specifications - Data Matrix, Jan, 05.

[2] Ho-Keun Oh, Latest Barcode Application Technoloty, 1997.

[3] GS1, Introduction to GS1 Data Matrix, 2010.

[4] Jin-Hee Hwang and Hee-il Hahn "Data Matrix barcode decoding algorithm, the implementation of

a two-dimensional", Korea Intelligent Information System Society, 2001.
[5] Jin-Hee Hwang and Hee-il Hahn, "Extracting Symbol Informations from Data Matrix two dimensional Barcode Image", the institute of electronics engineers of korea, 2002.
[6] "Data Matrix Barcode Search Algorithm on Post Envelope and Decoding Program Development", Information Technologies, Management and Society, Vol. 2, No. 1, pp. 13-22, 2009.
[7] Tadeja Muck, Branka Lozo, Arjana Žitnik "Multi-color 2d Data matrix codes With Poorly Readable Colors", Journal of Graphic Engineering and Design, Vol. 1, 2010.
[8] http://en.wikipedia.org/wiki/Data_matrix_(computer)

저자소개

한 희 준(Hee-June Han)

・2008년 3월 ~ 현재 : 충북대학교 컴퓨터교육과 학사과정

<관심분야> : 데이터베이스시스템, 과학 데이터베이스, 바코드 검증기

이 효 창(Hyo-Chang Lee)

・2009년 3월 ~ 현재 : 충북대학교 컴퓨터교육과 학사과정

<관심분야> : 데이터베이스시스템, 과학 데이터베이스, 바코드 검증기

이 종 연(Jong-Yun Lee) [종신회원]

・1985년 2월 : 충북대학교 전자계산공학과(공학사)
・1987년 2월 : 충북대학교 대학원 전자계산기공학과(공학석사)
・1999년 2월 : 충북대학교 전자계산학과(이학박사)
・1990년 2월 ~ 1996년 5월 : 현대전자산업(주) SW연구소 및 현대정보기술(주) CIM사업부 근무(책임연구원)
・1999년 3월 ~ 2003년 3월 : 강원대학교 삼척캠퍼스 정보통신공학과조교수
・2003년 3월 ~ 현재 : 충북대학교 컴퓨터교육과 및 디지털정보융합학과 교수로 재직 중

<관심분야> : 질의 처리 및 최적화, 시공간 데이터베이스, u-learning 및 평가모델, 유통물류, GIS

・E-Mail : jongyun@chungbuk.ac.kr