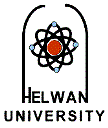
**Artificial Intelligence Project Specification**

**for**

**QR Code Generator**

**For Proff:Alaa Hamdy**

**Table of Contents**

**1. INTRODUCTION 1**

1.1 MODEL OF QRCODE 1

1.2 CHARACTERS WHICH CAN BE ENCODED IN QRCODE.(ENCODE MODE) 1

1.3 ERROR CORRECTING IN QRCODE. 1

1.4 VERSION IN QRCODE. 2

1.5 Structure of QRcode symbol 2

**2. How to encode 3**

2.1 Capacity 3

2.1 encode to data code words. 3

2.1.1 Mode indicator 4

2.1.2 Character count indicator 4

2.1.3 Encoding data in binary representation 5

2.1.4 Terminator 5

2.1.5 Encode to code words 7

2.3 Calculating error correcting code words 9

2.4 Data allocation 11

2.5 Mask pattern 11

2.5.1 Range of masking 11

2.5.2 Condition of masking 11

2.5.3 Select mask pattern 12

2.6 Format information 13

2.7 completion 14

**1. introduction**

QRcode is two dimensional code which is categorized in matrix code Sample image of QRcode is below.  


Features of QRcode are

* High-speed reading(QR is derived from "Quick Response")
* High capacity and high density
* Error correcting
* Structured Append

**1.1 Model of QRcode**

QRcode model1:Original model.  
QRcode model2:Extended model.

**1.2 Characters which can be encoded in QRcode.(encode mode)**

* numeric(0-9)  
  3 characters are encoded to 10bit length.  
  In theory, 7089 characters or less can be stored in a QRcode.
* alphanumeric(0-9A-Z $%\*+-./:)45characters  
  2 characters are encoded to 11bit length.  
  In theory, 4296 characters or less can be stored in a QRcode.
* 8bit byte data  
  In theory, 2953 characters or less can be stored in a QRcode.
* KANJI  
  A KANJI character(this is multi byte character) is encoded to 13bit length.  
  In theory, 1817 characters or less can be stored in a QRcode.

**1.3 Error correcting in QRcode.**

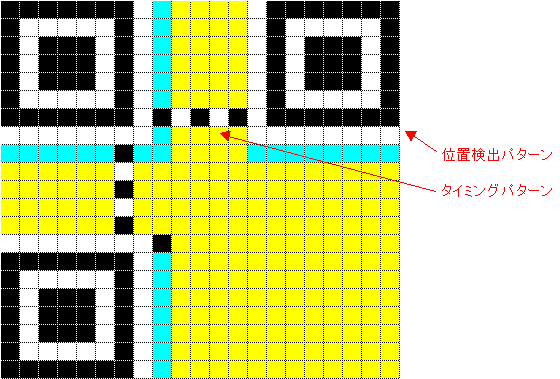
QRcode has a function of an error correcting for miss reading that white is black.  
Error correcting is defined in 4 level as below.

* level L : About 7% or less errors can be corrected.
* level M : About 15% or less errors can be corrected.
* level Q : About 25% or less errors can be corrected.
* level H : About 30% or less errors can be corrected.

**1.4 Version in QRcode.**

Size of QRcode is defined as version.  
Version is from 1 to 40.  
Version 1 is 21\*21 matrix. And 4 modules increases whenever 1 version increases. So version 40 is 177\*177 matrix.

**1.5 Structure of QRcode symbol.**

The below figure is a structure of QRcode model2 version 1. In the below figure,white or black parts are fixed in specification. They are "finder pattern" and "timing pattern". "Finder pattern" is used to help detection a position of QRcode in decoder application."Timing pattern" is used to help determine a symbol's coordinate in decoder application.  
  
In yellow color parts,encoded data (including error correct code) are stored.And in cyan color parts,information of error correcting level and mask pattern(described later) are stored.This is called format information  
  
  
  
■encoded data (including error correct code)  
■format information  
  
In addition, version2 or higher has "alignment pattern" which is used to correct skewness in decoder application.  
And version 7 or higher has "version information" which has information of version itself.

**2. How to encode**

This document is describe how to encode with an easy example, and no complex function(like structured append) is.

**2.1 Capacity**

Capacity of a QRcode is determined by version,error correcting level and encoding mode(e.g. numeric,alphanumeric etc) for example, In version 1 and error correcting level Q, 27 numeric characters can be stored or 16 alphanumeric characters can be stored. 11 byte data can also be stored. (see [table1](http://www.swetake.com/qr/qr_table0.html))Conversely ,version increases when error correct level is higher in same data. So, first we need consider error correcting level,and next we consider version if necessary.

**2.2 encode to data code words.**

8bit data is treated as a code word. We calculate or put data in this unit. In this section,"how to transform input data to code words" is described. Now we think to encode example source data "ABCDE123" to QRcode in version1-error correcting level H (below 1-H).

**2.2.1 Mode indicator**

First,mode indicator is created in 4 bit long as binary representation.  
  
numeric mode : 0001  
alphanumeric mode : 0010  
8bit byte mode: 0100  
KANJI mode : 1000  
  
Example data is alphanumeric,and we select an alphanumeric mode.  
  
**0010**

**2.2.2 Character count indicator**

Character count indicator is character counts stored in each mode.  
  
In version 1-9  
  
numeric : 10bit long  
alphanumeric : 9bit long  
8bit byte : 8bit long  
KANJI mode : 8bit long  
  
Example data have 8 characters,we encode 8 in 9bit long binary representation for alphanumeric mode.   
0010 **000001000**

**2.2.3 Encoding data in binary representation**

Next we think to encode source data to binary representation.  
  
In numeric mode ,data is delimited by 3 digits.   
For example,"123456" is delimited "123" and "456",and first data is "123",second data is "456".  
And each data is encoded in 10bit long binary representation.  
  
When length of delimited data is 1 or 2, 4bit long or 7bit long are used in each case.  
For example,"9876" is delimited "987" in 10 bit long and "6" in 4 bit long.  
Its result is "1111011011 0110"  
  
In alphanumeric mode ,each character is converted to value in rule of [table2](http://www.swetake.com/qr/qr_table1.html).  
Next we consider delimited data by 2 numbers. First value increase 45 times and second value is added to it. Result value is encoded in 11bit long binary representation. When length of delimited data is 1,6bit long are used.  
  
In example data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | "AB" | "CD" | "E1" | "23" |
|  |  | 45\*10+11 | 45\*12+13 | 45\*14+1 | 45\*2+3 |
|  |  | 461 | 553 | 631 | 93 |
| 0010 | 000001000 | **00111001101** | **01000101001** | **01001110111** | **00001011101** |

In 8bit byte mode,each value is directly encoded in 8bit long binary representation.

**2.2.4 Terminator**

We add 0000 to result data in section 2.2.3. When length of encoded data is full in this version and error correcting level,terminator is not needed.  
  
0010 000001000 00111001101 01000101001 01001110111 00001011101 **0000**

**2.2.5 Encode to code words**

Result data in section 2.2.4 are delimited by 8bit.  
  
00100000 01000001 11001101 01000101 00101001 11011100 00101110 10000  
  
If last data length is less than 8,padded 0.  
  
00100000 01000001 11001101 01000101 00101001 11011100 00101110 10000**000**  
  
If count of code words is less than symbol's capacity ([table1](http://www.swetake.com/qr/qr_table0.html)) ,then we alternately put "11101100" and "00010001" until full capacity.  
  
Now capacity of 1-H are 9,  
  
00100000 01000001 11001101 01000101 00101001 11011100 00101110 10000000 **11101100**   
  
To decimal representation...  
  
32 65 205 69 41 220 46 128 236

**2.3 Calculating error correcting code words**

Reed-solomon error correcting is used in QRcode.  
  
First, result data in previous section are delimited to RS block data by rule of [table5](http://www.swetake.com/qr/qr_table2.html).  
In example data, RS block number in 1-H is 1, we need not delimit.

Next select g(x) from [table3](http://www.swetake.com/qr/qr_table3.html).  
In example data,count of error correcting code words is 17, we select below g(x).  
  
g(x)=x17 +α43x16 +α139x15 +α206x14 +α78x13 +α43x12  
    +α239x11 +α123x10 +α206x9 +α214x8 +α147x7 +α24x6  
    +α99x5 +α150x4 +α39x3 +α243x2 +α163x +α136   
  
Above α is a primitive element on GF(28).   
  
Features of GF(28) are ...  
 1.four arithmetic operations are supported  
 2.α255=1  
 3.We can convert exponential in α to integer (or vice versa) using [table4](http://www.swetake.com/qr/qr_table4.html).  
  
  
Now polynomial f(x) which coefficients are data code words is divided by g(x).  
  
  
f(x)=32x25 +65x24 +205x23 +69x22 +41x21 +220x20 +46x19 +128x18 +236x17 <---(1)  
  
divide by g(x)  
  
Coefficient of leading term in f(x) is 32.  
For 32 is α5 from [table4](http://www.swetake.com/qr/qr_table4.html), we use  
 g(x)\*(α5)\*x8  
  
=α5\*x25 +α5\*α43\*x24 +α5\*α139\*x23 +α5\*α206\*x22 +α5\*α78\*x21.....  
  
=α5\*x25 +α48\*x24 +α144\*x23 +α211\*x22 +α83\*x21.....  
  
=32x25 +70x24 +168x23 +178x22 +187x21..... <---(2)  
  
calculate exclusive logical sum (1) and (2)  
  
f(x)'=7x24 +101x23 +247x22+146x21.....  
  
We repeat same logic until this devide calculation is over. Next, for 7 is α198,we use g(x)\*α198\*x7  
If exponent of α is over 255,then we decrease it using α255=1   
  
Finally we can get below remainder R(x).   
  
R(x)=42x16 +159x15 +74x14 +221x13 +244x12 +169x11+239x10   
　　　 +150x9 +138x8 +70x7 +237x6 +85x5 +224x4 +96x3 +74x2 +219x +61  
  
(see [table6](http://www.swetake.com/qr/qr_ecc_calc_sample_en.html))   
  
So we can get   
  
32 65 205 69 41 220 46 128 236 **42 159 74 221 244 169 239 150 138 70 237 85 224 96 74 219 61**

**2.4 Data allocation**

In QRcode,1 module means 1 bit. Result data in previous section are encoded to binary representation, and we allocate these encoded data. In addition when RS block number (see [table5](http://www.swetake.com/qr/qr_table2.html)) is 2 or higher, we should allocate data in interleaved. In example data, for 1-H has 1 RS block,we skipped how to interleave.We allocate fixed pattern which is "finding pattern" and "timing pattern" in advance.   
  
Rule for allocation

1. Now we think coordinate which has i lines j columns, and upper left corner is (0,0).  
For example, in version 1 this has modules from (0,0) to (20,20).  
  
2. Start module is lower right corner.  
In example data (1-H), start module is (20,20), and we put a data (0 or 1).  
  
3. Direction of movement (upper or lower) is kept. Direction is upper in first.  
  
4. We think 2 modules width.  
If we are in right module of 2 modules width....  
If left module is blank (not fixed pattern or version information etc), we move left module and put data.  
If left module is not blank, we move in direction which is kept, and put data  
  
If we are in left module....  
We check that blank module is in direction which is kept. If blank module is, we put data in right module in priority to left module of 2 modules width.  
If blank module is not,we move to a left module, and put data there.Then we turn  
direction which is kept.

Example  
  
If we have data "01234567 89ABCDEF GHIJKLMN" (We really have 0 or 1 in QRcode...) and we put this data in 6\*4 matrix...

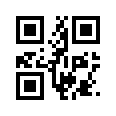
|  |  |  |  |
| --- | --- | --- | --- |
| D | C | B | A |
| F | E | 9 | 8 |
| H | G | 7 | 6 |
| J | I | 5 | 4 |
| L | K | 3 | 2 |
| N | M | 1 | 0 |

(^first direction)

In same data, fixed pattern "\*" is existed in 4\*2 center...

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | 8 | 7 | 6 |
| A | \* | \* | 5 |
| B | \* | \* | 4 |
| C | \* | \* | 3 |
| D | \* | \* | 2 |
| F | E | 1 | 0 |

(^first direction)

Below figure is put finder pattern,timing pattern and code words in example data.  
  


**2.5 Mask pattern**

If one color modules are much than other color or pattern like "finder pattern" exists, decoder application can have many mistake. To prevent above case,best mask pattern which is selected from among 8 pattern is covered in QRcode.

**2.5.1 Range of masking**

Range of masking is code words which is excepted "finder pattern","timing pattern" etc.

**2.5.2 Condition of masking**

We have 8 mask pattern in QRcode. Mask pattern indicator is 3 bit long binary representation.

|  |  |
| --- | --- |
| **mask pattern indicator** | **condition** |
| 000 | (i+j) mod 2 = 0 |
| 001 | i mod 2 = 0 |
| 010 | j mod 3 = 0 |
| 011 | (i+j) mod 3 = 0 |
| 100 | (( i div 2)+(j div 3)) mod 2 = 0 |
| 101 | (ij) mod 2 + (ij) mod 3 = 0 |
| 110 | ((ij) mod 2 +(ij) mod 3) mod 2 = 0 |
| 111 | ((ij)mod 3 + (i+j) mod 2) mod 2 = 0 |

"mod" means remainder calculation, "div" means integer divide

in mask pattern 000....  
at (20,20) : (20+20) mod 2 = 0 and reverse bit  
at (19,20) : (20+19) mod 2 = 1 and do not reverse bit

**2.5.3 Select mask pattern**

We estimate result of above mask pattern to select. We select a mask pattern which has least down point to calculate in which below table. In addition conditions are applied to whole symbol.

|  |  |  |
| --- | --- | --- |
| **characteristics** | **condition** | **down point** |
| concatenation of same color in a line or a column | count of modules=(5+i) | 3+i |
| module block of same color | block size 2\*2 | 3 |
| 1:1:3:1:1(dark:bright:dark:bright:dark)pattern in a line or a column |  | 40 |
| ration of dark modules in whole | from 50±(5+k)% to 50±(5+(k+1))% | 10\*k |

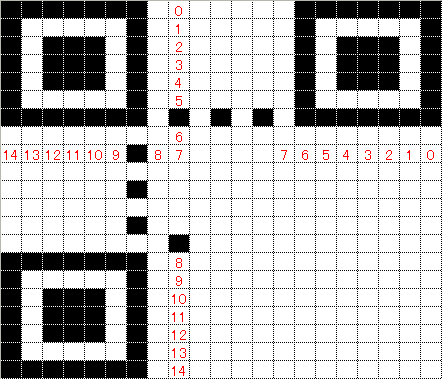
*I think it is not very important to select mask pattern exactly.  
In fact there is different between result of JIS X0510(1999) appendix 8 and JIS X0510(2004) appendix G in same data. And below result may is not exactly good.*   
  
We select a mask pattern "011" in example data.

**2.6 Format information**

Format information includes error correcting level and mask pattern indicator in 15 bit long.  
First 2 bit are error correcting level in below table.

|  |  |
| --- | --- |
| **error correcting level** | **indicator** |
| L | 01 |
| M | 00 |
| Q | 11 |
| H | 10 |

We select "10" in example data.  
  
**10**   
  
In next 3 bit, we put mask pattern indicator which is selected in previous section.  
  
10　**011**  
  
We put error correcting data which is Bose-Chaudhuri-Hocquenghem(BCH)(15,5) in right 10 bit.  
  
First, polynomial F(x) which coefficients are above 5 bit and x10 times is divided by below G(x).   
  
G(x)=x10+x8+x5+x4+x2+x+1  
  
In example data,5 bit data is "10011" and F(x) is below. F(x)=x14+x11+x10  
  
divide by G(x)....   
  
Remainder R(x)=x8+x7+x6+x   
  
So we get result as 10011 **0111000010**   
  
Finally,we calculate exclusive logical sum "101010000010010" and above result to avoid that result data is not all 0.   
  
Format information is "001100111010000".

We put results in number of below figure. (left side of bit data is upper bit[14]. )   
  


### 2.7 completion

