２０１１年度　卒業研究論文

QR コードの理論とプログラムの制作

岡山理科大学

総合情報学部

情報科学科

澤見研究室

I08I040　冨松 佑貴

I08I043　新田 勝啓

目次

1　はじめに

QR コードは 2 次元コードの一種であり QR コードリーダーにもなる携帯電話の普及率の増加により身近なものになってきた．そこで，私達は QR コードの仕組みを調べ，QR コードを生成するプログラムを Visual Basic 2008 により制作し，QR コードについての考察を加える．

２　QR コードについて

QR(Quick Response)コードとは，株式会社デンソーウェーブが開発したマトリックス方式の 2 次元コードである(図 1)．一方，スタック方式の 1 次元コードにバーコードがある(図 2)．QR コードとバーコードの違いは，同じデータでも QR コードのほうがコンパクトにすることができ，汚れにも強いことである．また，最大データ量が増え，どの方向からも読み取ることが可能になっている．

図 1　2 次元コード

図 2　1 次元コード

３　QR コードの仕組み

QR コードはおおまかに分けて，以下に示す五つの要素(1)位置検出パターン，(2)タイミングパターン，(3)形式情報，(4)データ・誤り訂正コード，(5)位置合わせパターンから成り立っている(図 3).

(1) 位置検出パターン(図 3 赤色部)
　左上，右上，左下の 3 か所に配置し，QR コードの位置と方向を決定する
(2) タイミングパターン(図 3 紫色部)
　モジュールの中心座標を決定する
(3) 形式情報(図 3 水色部)
　誤り訂正レベルやマスクパターンの情報を格納する
(4) データ・誤り訂正コード(図 3 黄色部)
　データと誤り訂正コードを格納する
(5) 位置合わせパターン(図 3 緑色部)
　歪みを補正する

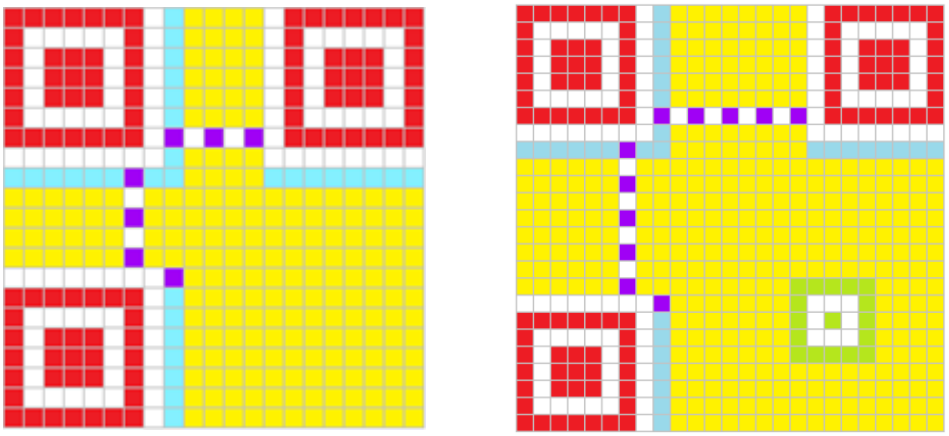


図 3　　QR コードの構造図

4　QR コード生成の手順

入力した文字列から QR コードとして出力するまでの生成手順の大まかな流れは以下のようになる(図 4).

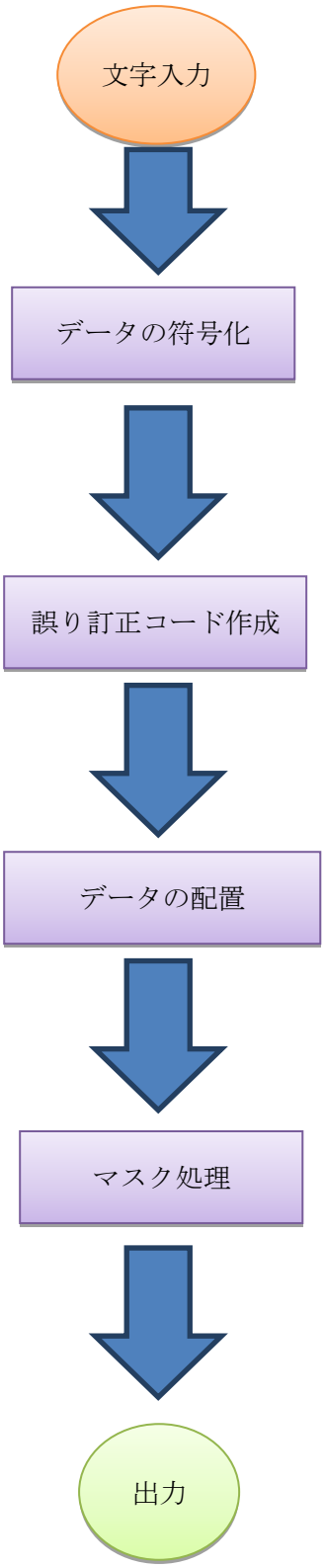


図 4　QR コード生成の手順

４．１　文字入力

QR コードを生成するには，まず入力された文字から文字種を示すモードを決定し，エラー訂正レベルと文字数からバージョン，データコード語数，誤り訂正コード語数を決定する(表 1).

表 1　QR コードのコード語数とデータ容量

バージョン	誤り訂正レベル	データコード語数	誤り訂正コード語数	数字	英数字	漢字	8bit
1	L	19	7	41	25	10	17
	M	16	10	34	20	8	14
	Q	13	13	27	16	7	11
	H	9	17	17	10	4	7
2	L	34	10	77	47	20	32
	M	28	16	63	38	16	26
	Q	22	22	48	29	12	20
	H	16	28	34	20	8	14
3	L	55	15	127	77	32	53
	M	44	26	101	61	26	42
	Q	34	36	77	47	20	32
	H	26	44	58	35	15	24
4	L	80	20	187	114	48	78
	M	64	36	149	90	38	62
	Q	48	52	111	67	28	46
	H	36	64	82	50	21	34

今回使用する例では以下のようになる.

データ：**ABCDE123**
モード：**英数字**
バージョン：**1**
誤り訂正レベル：**H**
データコード数：**9**
誤り訂正コード語数：**17**

４．２　データの符号化

データの符号化とは，データのモード従って，データをビット列に変換する際，データの先頭にモード指示子と文字数指示子をつけ，もしデータが最大文字数に達していなければデータの最後に終端パターンとして 0 を 4 ビット挿入し，それを 8 ビットごとに分割することである(表 2,3).

また分割したコードの数がデータコード数に達していなければ，データコード数になるまで 236 と 17 を交互につける.

表 2　モード指示子

モード指示子	
数字モード	0001
英数字モード	0010
8bit バイトモード	0100
漢字モード	1000

表 3　文字数指示子

文字数指示子	
数字モード	10bit
英数字モード	9bit
8bit バイトモード	8bit
漢字モード	8bit

例として入力データ ABCDE123 を符号化する．まず入力されたデータは，表 2 からモード指示子は以下の数値になる．

0010

次に文字数指示子は，入力された文字が 8 文字なので表 3 よりそれを 2 進数の 9bit で表現すると以下のようになる．

000001000

英数字モードでは，まずデータを 2 文字ずつ区切り対応表より文字を数値化する（表 4）．次に 1 文字目の数値を 45 倍したものに 2 文字目の数値を加え，その数値を 11bit で表現すると以下のようになる．なお 2 文字に満たない場合は対応する数値を 6bit で表現する．

AB	CD	E1	23
45*10+11	45*12+13	45*14+1	45*2+3
461	553	631	93
00111001101	**01000101001**	**01001110111**	**00001011101**

表 4　英数字モード対応表

0	0	A	10	K	20	U	30	+	40
1	1	B	11	L	21	V	31	−	41
2	2	C	12	M	22	W	32	.	42
3	3	D	13	N	23	X	33	/	43
4	4	E	14	O	24	Y	34	:	44
5	5	F	15	P	25	Z	35		
6	6	G	16	Q	26	[sp]	36		
7	7	H	17	R	27	$	37		
8	8	I	18	S	28	%	38		
9	9	J	19	T	29	*	39		

最後にデータが最大文字数に達していなければ終端パターンとして 0 を 4bit つける．

0000

ここまでで出たビット列を 8bit 毎に区切ると以下のようになる．

00100000 01000001 11001101 01000101 00101001 11011100 00101110 10000

最後のビット列が 8bit 未満の場合 0 で埋める．

00100000 01000001 11001101 01000101 00101001 11011100 00101110 10000000

これを 10 進数で表現すると以下のようになる．

32 65 205 69 41 220 46 128

得られたコード語数がシンボルのデータコード語数に満たない場合 236 と 17 を交互につける．今回例では表 1 よりデータコード語数が 9，得られたコード語数が 8 で満たしていないので，236 を一つ付けることになり，最終的なデータとして以下のようになる．

32 65 205 69 41 220 46 128 236

４．３　誤り訂正コード

QR コードではリード・ソロモン(以下ＲＳ)誤り訂正方式により冗長コードを付加し誤り訂正機能を持たせている．冗長コードを付加するためコード列を一定の規則に従って RS ブロックに分割する．この例では RS ブロック数＝1 なので分割は不要である．

今回使用した例では誤り訂正コード語数が 17 なので以下のような式になる(表 5).

表 5　誤り訂正コード語数対応表

誤り訂正コード語数	生成多項式 g(x)
7	$x^7 + a^{87}x^6 + a^{229}x^5 + a^{146}x^4 + a^{149}x^3 + a^{238}x^2 + a^{102}x^1 + a^{21}$
10	$x^{10} + a^{251}x^9 + a^{67}x^8 + a^{46}x^7 + a^{61}x^6 + a^{118}x^5 + a^{70}x^4 + a^{64}x^3 + a^{94}x^2 + a^{32}x^1 + a^{45}$
13	$x^{13} + a^{74}x^{12} + a^{152}x^{11} + a^{176}x^{10} + a^{100}x^9 + a^{86}x^8 + a^{100}x^7 + a^{106}x^6 + a^{104}x^5 + a^{130}x^4 + a^{218}x^3 + a^{206}x^2 + a^{140}x^1 + a^{78}$
15	$x^{15} + a^{8}x^{14} + a^{183}x^{13} + a^{61}x^{12} + a^{91}x^{11} + a^{202}x^{10} + a^{37}x^9 + a^{51}x^8 + a^{58}x^7 + a^{58}x^6 + a^{237}x^5 + a^{140}x^4 + a^{124}x^3 + a^{5}x^2 + a^{99}x^1 + a^{105}$
16	$x^{16} + a^{120}x^{15} + a^{104}x^{14} + a^{107}x^{13} + a^{109}x^{12} + a^{102}x^{11} + a^{161}x^{10} + a^{76}x^9 + a^{3}x^8 + a^{91}x^7 + a^{191}x^6 + a^{147}x^5 + a^{169}x^4 + a^{182}x^3 + a^{194}x^2 + a^{225}x^1 + a^{120}$
17	$x^{17} + a^{43}x^{16} + a^{139}x^{15} + a^{206}x^{14} + a^{78}x^{13} + a^{43}x^{12} + a^{239}x^{11} + a^{123}x^{10} + a^{206}x^9 + a^{214}x^8 + a^{147}x^7 + a^{24}x^6 + a^{99}x^5 + a^{150}x^4 + a^{39}x^3 + a^{243}x^2 + a^{163}x^1 + a^{136}$
18	$x^{18} + a^{215}x^{17} + a^{234}x^{16} + a^{158}x^{15} + a^{94}x^{14} + a^{184}x^{13} + a^{97}x^{12} + a^{118}x^{11} + a^{170}x^{10} + a^{79}x^9 + a^{187}x^8 + a^{152}x^7 + a^{148}x^6 + a^{252}x^5 + a^{179}x^4 + a^{5}x^3 + a^{98}x^2 + a^{96}x^1 + a^{153}$
20	$x^{20} + a^{17}x^{19} + a^{60}x^{18} + a^{79}x^{17} + a^{50}x^{16} + a^{61}x^{15} + a^{163}x^{14} + a^{26}x^{13} + a^{187}x^{12} + a^{202}x^{11} + a^{180}x^{10} + a^{221}x^9 + a^{225}x^8 + a^{83}x^7 + a^{239}x^6 + a^{156}x^5 + a^{164}x^4 + a^{212}x^3 + a^{212}x^2 + a^{188}x^1 + a^{190}$
22	$x^{22} + a^{210}x^{21} + a^{171}x^{20} + a^{247}x^{19} + a^{242}x^{18} + a^{93}x^{17} + a^{230}x^{16} + a^{14}x^{15} + a^{109}x^{14} + a^{221}x^{13} + a^{53}x^{12} + a^{200}x^{11} + a^{74}x^{10} + a^{8}x^9 + a^{172}x^8 + a^{98}x^7 + a^{80}x^6 + a^{219}x^5 + a^{134}x^4 + a^{160}x^3 + a^{105}x^2 + a^{165}x^1 + a^{231}$
24	$x^{24} + a^{229}x^{23} + a^{121}x^{22} + a^{135}x^{21} + a^{48}x^{20} + a^{211}x^{19} + a^{117}x^{18} + a^{251}x^{17} + a^{126}x^{16} + a^{159}x^{15} + a^{180}x^{14} + a^{169}x^{13} + a^{152}x^{12} + a^{192}x^{11} + a^{226}x^{10} + a^{228}x^9 + a^{218}x^8 + a^{111}x^7 + a^{255}x^6 + a^{117}x^5 + a^{232}x^4 + a^{87}x^3 + a^{96}x^2 + a^{227}x^1 + a^{21}$
26	$x^{26} + a^{173}x^{25} + a^{125}x^{24} + a^{158}x^{23} + a^{2}x^{22} + a^{103}x^{21} + a^{182}x^{20} + a^{118}x^{19} + a^{17}x^{18} + a^{145}x^{17} + a^{201}x^{16} + a^{111}x^{15} + a^{28}x^{14} + a^{165}x^{13} + a^{53}x^{12} + a^{161}x^{11} + a^{21}x^{10} + a^{245}x^9 + a^{142}x^8 + a^{13}x^7 + a^{102}x^6 + a^{48}x^5 + a^{227}x^4 + a^{153}x^3 + a^{145}x^2 + a^{218}x^1 + a^{70}$
28	$x^{28} + a^{168}x^{27} + a^{223}x^{26} + a^{200}x^{25} + a^{104}x^{24} + a^{224}x^{23} + a^{234}x^{22} + a^{108}x^{21} + a^{180}x^{20} + a^{110}x^{19} + a^{190}x^{18} + a^{195}x^{17} + a^{147}x^{16} + a^{205}x^{15} + a^{27}x^{14} + a^{232}x^{13} + a^{201}x^{12} + a^{21}x^{11} + a^{43}x^{10} + a^{245}x^9 + a^{87}x^8 + a^{42}x^7 + a^{195}x^6 + a^{212}x^5 + a^{119}x^4 + a^{242}x^3 + a^{37}x^2 + a^{9}x^1 + a^{123}$

$$g(x)=x^{17}+\alpha^{43}x^{16}+\alpha^{139}x^{15}+\alpha^{206}x^{14}+\alpha^{78}x^{13}$$
$$+\alpha^{43}x^{12}+\alpha^{239}x^{11}+\alpha^{123}x^{10}+\alpha^{206}x^{9}$$
$$+\alpha^{214}x^{8}+\alpha^{147}x^{7}+\alpha^{24}x^{6}+\alpha^{99}x^{5}+\alpha^{150}x^{4}$$
$$+\alpha^{39}x^{3}+\alpha^{243}x^{2}+\alpha^{163}x+\alpha^{136} \quad (1)$$

データ ABCDE123 を符号化したものは，32 65 205 69 41 220 46 128 236 なのでこれを式で表すと以下のようになる．

$$f(x)=32x^{25}+65x^{24}+205x^{23}+69x^{22}+41x^{21}+220x^{20}$$
$$+46x^{19}+128x^{18}+236x^{17} \quad (2)$$

データコードを係数とした多項式 f(x)を g(x)で除算する必要がある x のべき乗に対するが係数が対応していないことから f(x)がこのままでは除算することができないことから，次の対応表より f(x)を以下の式に変換する(表 6).

表 6　α の指数と整数の対応表

α の指数 Exponent of α	整数 Integer		整数 Integer	α の指数 Exponent of α
0	1			
1	2		1	0
2	4		2	1
3	8		3	25
4	16		4	2
5	32		5	50
6	64		6	26
7	128		7	198
8	29		8	3
9	58		9	223
10	116		10	51
11	232		11	238
12	205		12	27
13	135		13	104
14	19		14	199
15	38		15	75
16	76		16	4
17	152		17	100
18	45		18	224
19	90		19	14
20	180		20	52

$$f(x)= \alpha^{5}x^{25} + \alpha^{191}x^{24} + \alpha^{12}x^{23} + \alpha^{221}x^{22} + \alpha^{147}x^{21} + \alpha^{187}x^{20} + \alpha^{130}x^{19} + \alpha^{7}x^{18} + \alpha^{122}x^{17} \quad (3)$$

f(x)を g(x)により除算する．多項式 f(x)の g(x)による除算を簡単に表すと次のようになる．

$$g(x)*(\alpha^{5})*x^{8}$$
$$=\alpha^{5}*x^{25}+\alpha^{5}*\alpha^{43}*x^{24}+\alpha^{5}*\alpha^{139}*x^{23}$$
$$+\alpha^{5}*\alpha^{206}*x^{22}+\alpha^{5}*\alpha^{78}*x^{21}\cdots$$
$$=\alpha^{5}*x^{25}+\alpha^{48}*x^{24}+\alpha^{144}*x^{23}$$
$$+\alpha^{211}*x^{22}+\alpha^{83}*x^{21}\cdots$$

表 6 より α の指数から整数に変換すると次のようになる．

$$=32x^{25} +70x^{24} +168x^{23} +178x^{22} +187x^{21}\cdots \quad (4)$$

得られた式(2)と式(4)の各項の係数の排他論理和を計算して f(x)'を得る

$$f(x)'=7x^{24} +101x^{23} +247x^{22}+146x^{21}\cdots \quad (5)$$

このような処理を繰り返し，次は f(x)'の最初の項の係数 7 に対応する α^{198} を商の次の項の係数とし，$g(x)*\alpha^{198}*x^{7}$ を計算し係数部を整数化して排他論理和をとる．なお α のべき乗の指数が 255 を超えたときは $\alpha^{255}=1$ を利用して 255 未満にする．この結果，以下の剰余が得られる．

$$R(x)=42x^{16}+159x^{15}+74x^{14}+221x^{13}+244x^{12} +169x^{11}+239x^{10}+150x^{9}+138x^{8} +70x^{7}+237x^{6}+85x^{5}+224x^{4}+96x^{3} +74x^{2}+219x+61 \quad (6)$$

これらの計算結果から，QR コード作成に必要な最終的なデータは符号化されたデータの後ろに剰余 R(x)の各項の係数を付けた以下のようなものになる

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**

４．４　データの配置

データの配置のルールは以下のようにしている．

(1)一番左上を(0,0)とし x 行 y 列(x,y)の座標系を考える．

(2)スタートは右下としここにデータ(0 or 1)を配置する．

(3)上下の進行方向を決めておく．また，最初の進行方向は上とする．

(4)幅二つ分を基準とし，幅二つ分のうち右側にいるとき，ひとつ左に移動することを試みる．あいていればそこに移動し次のデータを配置し，もし固定のパターン等で埋まっていたら現在の進行方向（上または下）へ１つ移動しデータを配置する．

(5)幅２つ分のうち左側にいるときそこより現在の進行方向側に空きがあるか確認し，空きがあれば現在のモジュールに上下方向で最も近くかつ幅２つのうち右側に優先してデータを配置する．

(6)もし進行方向に空きが無ければ一つ左方向へ移動しそこにデータを配置する．またその後，進行方向を今までの逆にする．

４．４．１　配置の具体例

先ほどのデータ ABCDE123 符号化したデータの一部を, 6 行 4 列のマトリックスになるよう上記のルールで配置すると次のようになる(図 5 の左).

また同様の同じ大きさのマトリックスで，中央の 4 行 2 列がすでに固定パターン"＊"により埋まっている場合，上記のルールで配置すると次のようになる(図 5 の右).

0	0	0	0
1	0	1	0
1	1	0	0
0	0	0	0
1	1	0	1
1	0	0	0

1	0	0	0
0	*	*	0
0	*	*	0
0	*	*	0
0	*	*	1
1	0	0	0

図 5　配置例

４．５　マスク処理について

マスク処理とは，データを配置した状態で，黒か白の一方の色のモジュールが極端に多くなったり，位置検出パターンに類似した模様による読み込みエラーが起きるのを防ぐためのものである．

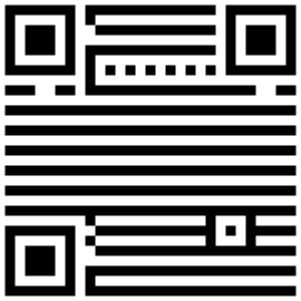
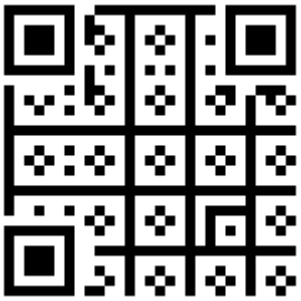
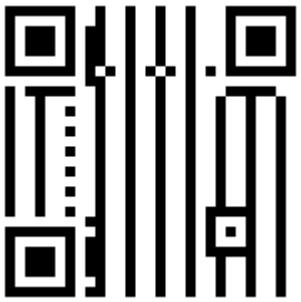
以下はマスクパターン参照子とそれに対応するマスクが掛かる位置の条件式を示したものである(表 7)．

表 7　マスクパターン参照子

マスクパターン参照子	条件
000	(x+y) mod 2 = 0
001	x mod 2 = 0
010	y mod 3 = 0
011	(x+y) mod 3 = 0
100	((x div 2)+(y div 3)) mod 2 = 0
101	(xy) mod 2 + (xy) mod 3 = 0
110	((xy) mod 2 +(xy) mod 3) mod 2 = 0
111	((xy)mod 3 + (x+y) mod 2) mod 2 = 0

各マスクパターンのマスクの掛かりかたは，マスクパターン参照子(表 7)で示された条件式から以下のようになる(表 8).

表 8　マスクの掛かり方

マスクパターン参照子	マスクパターン	マスクパターン参照子	マスクパターン
000		100	
001		101	
010		110	
011		111	

４．５．１　マスクの評価

各マスクパターンでマスク処理を実行した後，以下の特徴の評価条件に従ってマスクの評価を行い，失点の合計が少ないパターンを採用する(表 9).

なお評価は機能パターンを含むシンボル全体に対して行う.

表 9　マスクの評価

特徴	評価条件	失点
同色の行／列の隣接モジュール	モジュール数＝(5+i)	3+i
同色のモジュールブロック	ブロックサイズ　＝2×2	3
行／列における 1:1:3:1:1 (黒:白:黒:白:黒)のパターン	1:1:3:1:1 比率のパターンの前又は後ろに比率4の幅以上の白パターンが存在する。	40
全体に占める黒モジュールの割合	50±(5×k)%～ 50±(5×(k+1))%	10×k

i：同色の隣接したモジュール数が 5 個を超える分

k：シンボル内の黒モジュールの比率

４．６　形式情報

形式情報とは，誤り訂正レベルとマスクパターンを格納しておくものであり，15bit で表される．最初に誤り訂正レベルに対応する 2bit の数値を以下の表から当てはめる(表 10).

表 10　誤り訂正レベルに対応する 2 進数

誤り訂正レベル	2 進数
L	01
M	00
Q	11
H	10

ここで示す例では誤り訂正レベルは H なので 10 を当てはめる．次の 3bit に使用したマスクパターン参照子を割り当てる．今回はマスクパターン参照子として 011 を当てはめるものとする．残りの 10bit は形式情報に対する誤り訂正符号を付加する．また誤り訂正符号には BCH 符号を使用するものとする．形式情報に対する誤り訂正符号の求め方は，まず上記で得られた 5bit を項の係数とする多項式を作ると以下のような式となる．

$$f(x) = x^4+x+1 \quad (1)$$

先ほど出した式 f(x)を x^{10} 倍し以下の式を求める．

$$f(x) = x^{14}+x^{11}+x^{10} \quad (2)$$

求めた式を以下の式で割る．

$$G(x) = x^{10}+x^8+x^5+x^4+x^2+x+1 \quad (3)$$

この結果，下のような剰余を求めることができる．

$$\text{剰余 } R(x) = x^8+x^7+x^6+x \quad (4)$$

よって求めるビット列は誤り訂正レベルとマスクパターン参照子 5bit に誤り訂正符号 10bit を付加することにより，以下のようになる．

10011　0111000010

最後にできたビット列が全て白モジュールにならないようにするために以下のビット列と XOR 演算する．

101010000010010

最終的なビット列として以下のものが得られる．

001100111010000

求めたビット列は，QR コードの形式情報を格納する部分に縦と横に同じものを順番に配置する．図中に示す数値は，格納する順番を示す(図 6).

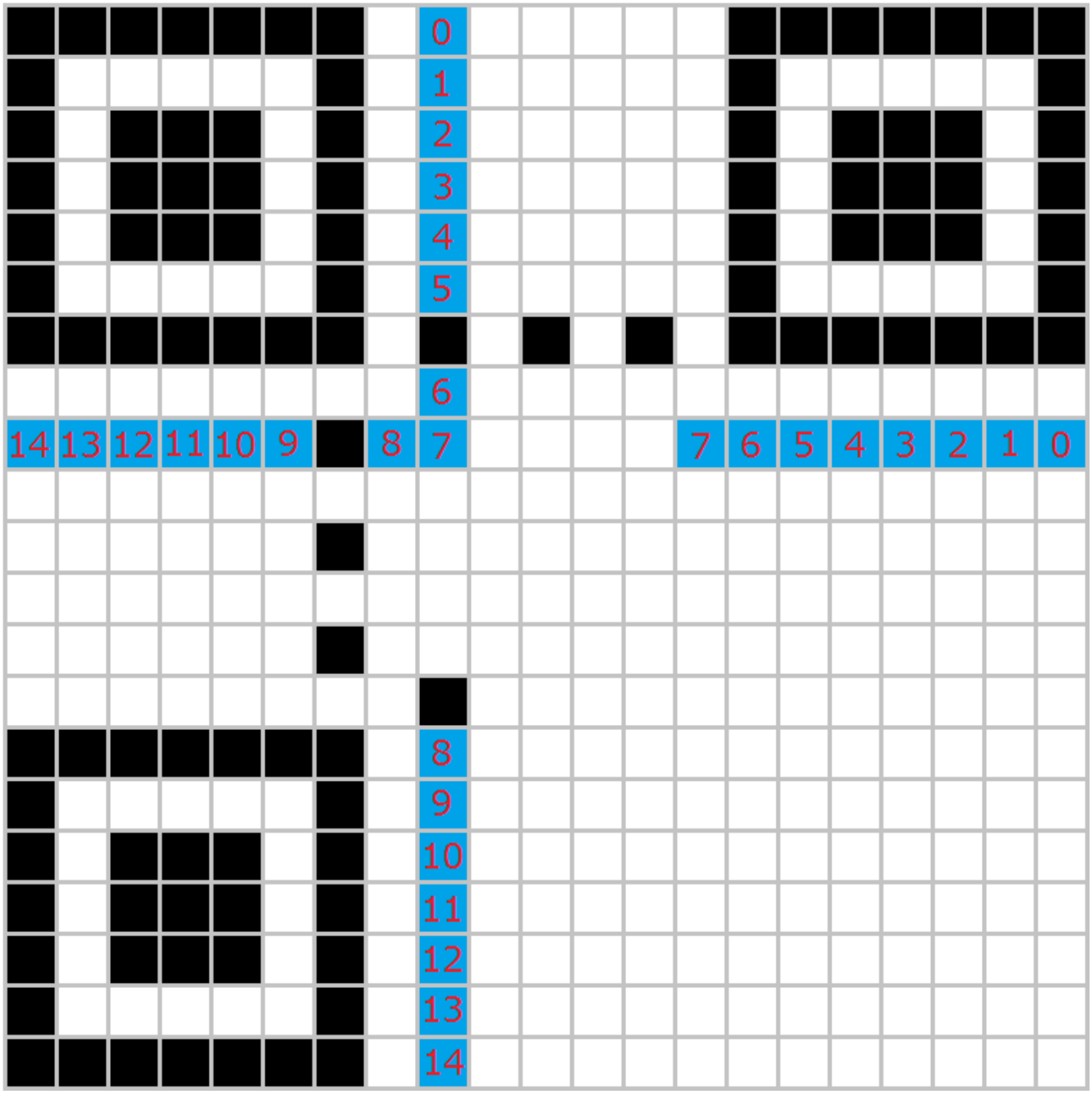


図 6　形式情報の格納位置

５　QR コード読み取りの手順

まず，(1)位置検出パターンから位置を検出する．すぐ隣の(2)形式情報からバージョンと使用したマスクと誤り訂正レベルがわかり，読み取ったマスクと同じもので(3)マスキングを行うと(4)マスク前のデータと誤り訂正コードが取得できるので，それを(5)文字列に復号する(図 7)．

また読み取り時の注意点として，QR コードを画像加工ツールなどで拡大・縮小すると，一つ一つのセルが歪んだものになってしまう可能性があり読み取れなくことがある．QR コードの周囲に，文字や絵を配置すると，マージン（余白）を確保することができなく読み取りにくい，あるいは，読み取りができなくなることがあるので注意する必要がある．

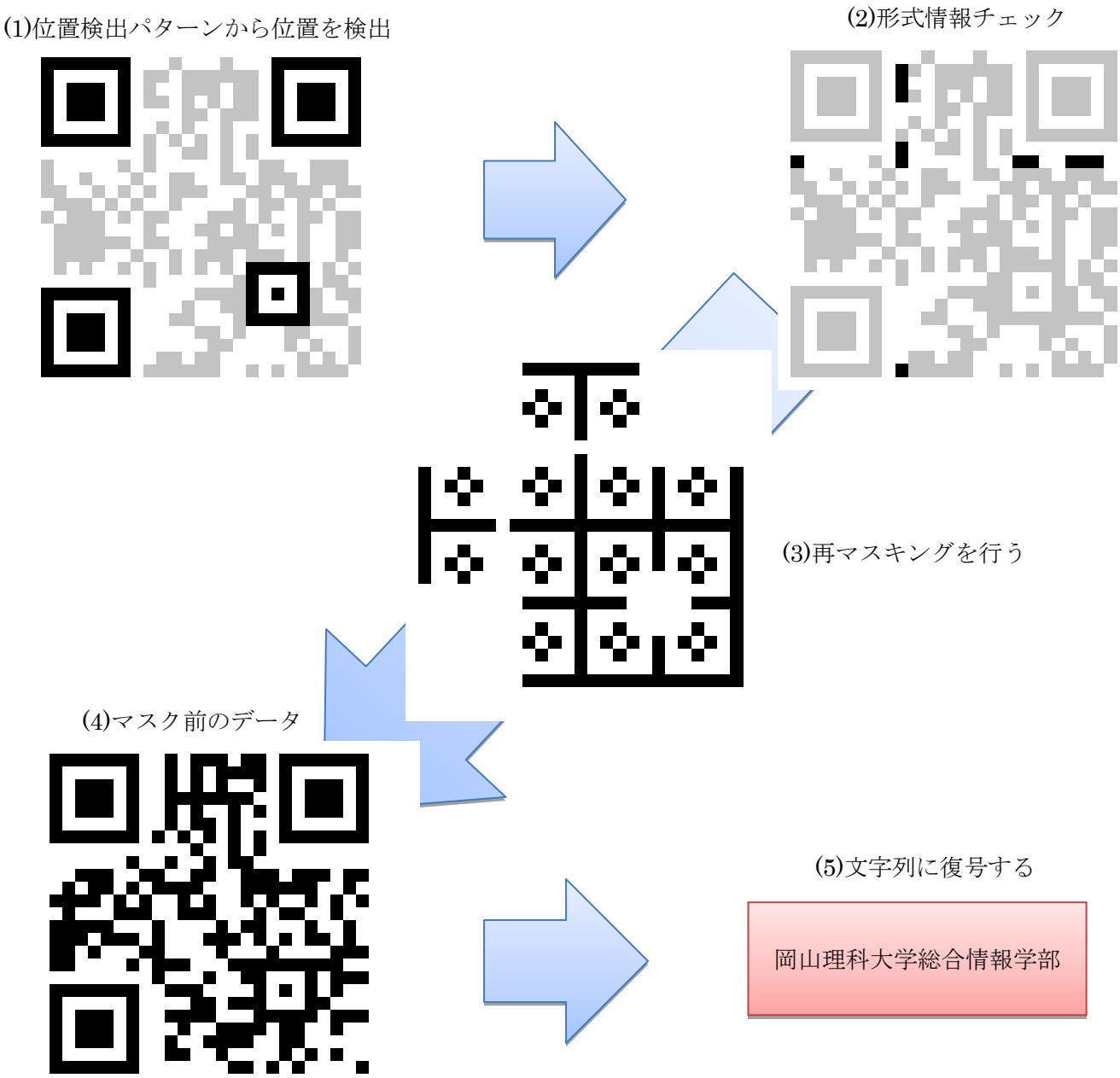


図 7　読み取り手順説明

６　制作プログラムの紹介

以下に今回作成したプログラムの機能を説明する(図 8).

(1)データ入力部

QR コードに変換したい文字を入れる

(2)マスク選択部

マスクを 8 パターンの中から選択する

(3)誤り訂正レベル選択部

誤り訂正レベルを選択する

レベル L(約 7%が復元可能)，M(約 15%が復元可能)，Q(約 25%が復元可能)，H(約 30%が復元可能)

(4)生成ボタン

データ入力部とマスク選択部と誤り訂正レベル選択部]情報を元に QR コードを作成する

(5)QR コード表示部

生成ボタンで作成した QR コードを表示する場所

(6)保存ボタン

生成した QR コードを保存することができる

(7)終了ボタン

プログラムを終了させる

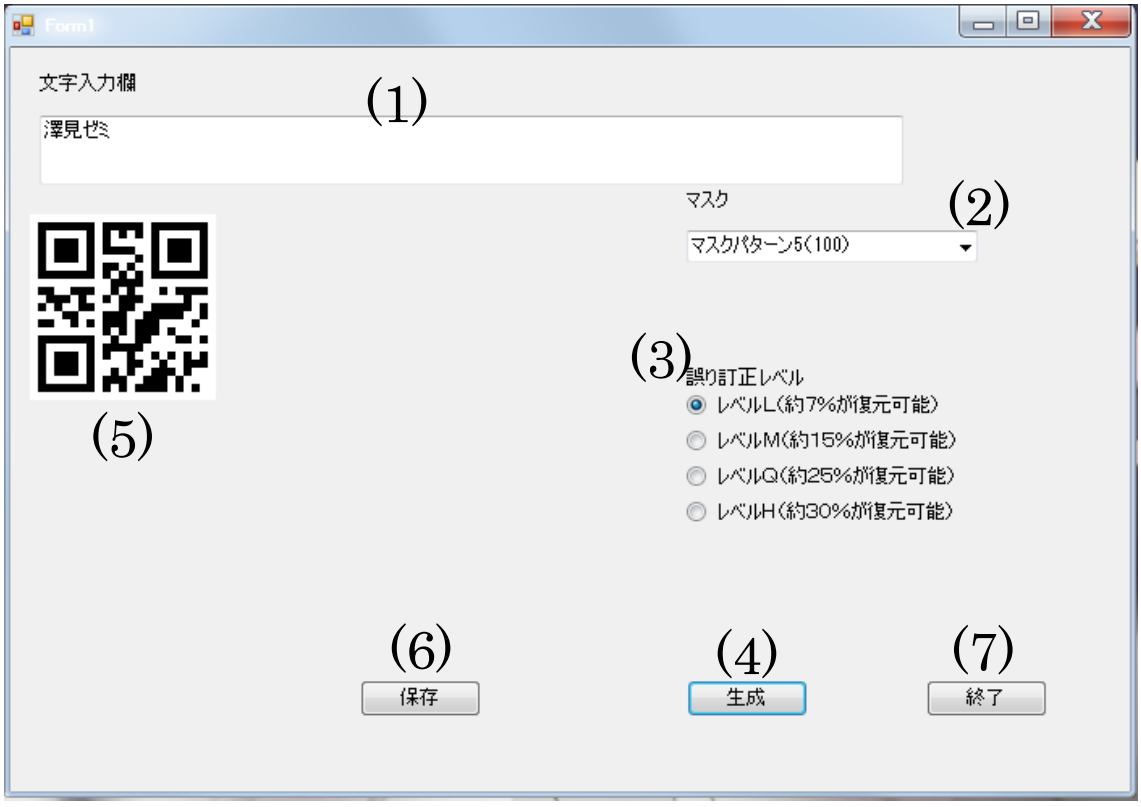


図 8　制作プログラム

7　実験例

本研究で用いたデータをもとに実際に生成した QR コードとデータとを対応させて以下に示す(図 9).

データ：**ABCDE123**
モード：**英数字**
バージョン：**1**
誤り訂正レベル：**H**
データコード数：**9**
誤り訂正コード語数：**17**

(a) ABCDE123

データ：**岡山理科大学総合情報学部**
モード：**漢字**
バージョン：**2**
誤り訂正レベル：**M**
データコード数：**28**
誤り訂正コード語数：**16**

(b) 岡山理科大学総合情報学部

データ：**澤見ゼミ**
モード：**漢字**
バージョン：**2**
誤り訂正レベル：**L**
データコード数：**19**
誤り訂正コード語数：**7**

(c) 澤見ゼミ

図 9　生成した QR コードと対応するデータ

８　考察

実際にプログラムを作成し実験することで，QR コードの仕組みや QR コードをどのように読み取っているかがわかった．また QR コードは誤り訂正レベルを適切に設定することで，ある程度図や文字を重ねても読み取ることが可能であることがわかった．

QR コードを生成するプログラムを自分たちで作り上げ，生成したコードをバーコードリーダにより読み取ることもできた．また，数字･英語･日本語に対応させることもできた．

今後の課題としては，今回作成したプログラムの高速化を図ること，混合モードでの最大文字数の見直しをすることなどが挙げられる．

謝辞

学士課程の１年間において，本論文を執筆するにあたり多くの御指導，御助言を下さいました岡山理科大学大学院総合情報研究科情報科学専攻, 澤見英男教授に心から御礼申し上げます．また, 多くのご指導, ご指摘を下さいました澤見ゼミの朝倉裕貴先輩に感謝いたします.

参考文献

[1]QR コードをつくってみる

http://www.swetake.com/qr/qr1.html

[2]Visual Basic .Net 入門教室について

http://homepage.mac.com/tuyano/VBNetTutor/index.html

[3]Visual Basic 中学校

http://homepage1.nifty.com/rucio/main/main.html

[4]バーコード生成

http://tobu-note.ver2.co.jp/barcode/

[5]浅山研究室２００３年度卒業研究

http://gemini.mis.ous.ac.jp/Rejume/2003/QR.pdf

[6]Notebook

http://void.yamicha.com/notebook/

[7]QR コードドットコム

http://www.qrcode.com/index.html