INTERNATIONAL STANDARD

ISO/IEC 18004

First edition
2000-06-15

Information technology — Automatic identification and data capture techniques — Bar code symbology — QR Code

Technologies de l'information — Techniques d'identification automatique et de capture de données — Symboles de codes à barres — Code QR

Reference number
ISO/IEC 18004:2000(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 18004 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*, in collaboration with AIM Inc.[1)].

Annexes A to F form a normative part of this International Standard. Annexes G to M are for information only.

1) AIM Inc., 634 Alpha Drive, Pittsburgh, PA 15238-2802, U.S.A.

Introduction

QR Code is a matrix symbology consisting of an array of nominally square modules arranged in an overall square pattern, including a unique finder pattern located at three corners of the symbol and intended to assist in easy location of its position, size and inclination. A wide range of sizes of symbol is provided for together with four levels of error correction. Module dimensions are user-specified to enable symbol production by a wide variety of techniques. QR Code Model 1 is the original specification for QR Code; QR Code Model 2 is an enhanced form of the symbology with additional features and can be auto-discriminated from Model 1. Since Model 2 is the recommended model for new, open systems application of QR Code, this International Standard describes Model 2 fully, and specifies the features in which Model 1 QR Code differs from Model 2 in an annex.

Information technology — Automatic identification and data capture techniques — Bar code symbology — QR Code

1 Scope

This International Standard specifies the requirements for the symbology known as QR Code. It specifies the QR Code Model 2 symbology characteristics, data character encodation, symbol formats, dimensional characteristics, error correction rules, reference decoding algorithm, production quality requirements, and user-selectable application parameters, and defines in an annex the features of Model 1 symbols which differ from Model 2.

2 Conformance

QR Code symbols (and equipment designed to produce or read QR Code symbols) shall be considered as meeting this specification if they meet the requirements defined for either QR Code Model 2 or Model 1. It should be noted, however, that Model 2 is the form of the symbology recommended for new and open systems applications.

3 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this International Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 15424, *Information technology — Automatic identification and data capture techniques — Data carrier/symbology identifiers.*

ISO/IEC 15416, *Information technology — Automatic identification and data capture techniques — Bar code print quality test specifications — Linear symbols.*

EN 1556, *Bar Coding — Terminology.*

JIS X 0201, *JIS 8-bit Character Set for Information Interchange.*

JIS X 0208-1997, *Japanese Graphic Character Set for Information Interchange.*

ANSI X3.4, *Coded Character Sets — 7-bit American National Standard Code for Information Interchange (7-bit ASCII).*

AIM International Technical Specification, *Extended Channel Interpretations: Part 1: Identification scheme and protocol* (referred to as "AIM ECI specification").

4 Terms and definitions

For the purposes of this International Standard, the terms and definitions given in EN 1556 and the following apply.

4.1
Alignment Pattern
fixed reference pattern in defined positions in a matrix symbology, which enables the decode software to re-synchronise the coordinate mapping of the image modules in the event of moderate amounts of distortion of the image

4.2
Character Count Indicator
bit sequence which defines the data string length in a mode

4.3
ECI designator
six-digit number identifying a specific ECI assignment

4.4
encoding region
region of the symbol not occupied by function patterns and available for encodation of data and error correction codewords

4.5
Extended Channel Interpretation (ECI)
protocol used in some symbologies that allows the output data stream to have interpretations different from that of the default character set

4.6
Extension Pattern
in Model 1 symbols, a function pattern which does not encode data

4.7
Format Information
function pattern containing information on the error correction level applied to the symbol and on the masking pattern used, essential to enable the remainder of the encoding region to be decoded

4.8
function pattern
overhead component of the symbol required for location of the symbol or identification of its characteristics to assist in decoding

4.9
Mask Pattern Reference
three-bit identifier of the masking patterns applied to the symbol

4.10
masking
process of XORing the bit pattern in the encoding region with a masking pattern to provide a symbol with more evenly balanced numbers of dark and light modules and reduced occurrence of patterns which would interfere with fast processing of the image

4.11
mode
method of representing a defined character set as a bit string

4.12
Mode Indicator
four-bit identifier indicating in which mode the next data sequence is encoded

4.13
Padding Bit
0 bit, not representing data, used to fill empty positions of the final codeword after the Terminator in a data bit string

4.14
Position Detection Pattern
one of three identical components of the Finder Pattern

4.15
Remainder Bit
0 bit, not representing data, used to fill empty positions of the symbol encoding region after the final symbol character, where the encoding region does not divide exactly into eight-bit symbol characters

4.16
Remainder Codeword
Pad Codeword used to fill empty codeword positions to complete the symbol if the total number of data and error correction codewords does not exactly fill its nominal capacity

NOTE The Remainder codewords come after the error correction codewords.

4.17
segment
sequence of data encoded according to the rules of one ECI or encodation mode

4.18
Separator
function pattern of all light modules, one module wide, separating the Position Detection Patterns from the rest of the symbol

4.19
Terminator
bit pattern 0000 used to end the bit string representing data

4.20
Timing Pattern
alternating sequence of dark and light modules enabling module coordinates in the symbol to be determined

4.21
Version
size of the symbol represented in terms of its position in the sequence of permissible sizes from 21 × 21 modules (Version 1) to 177 × 177 (Version 40) modules

NOTE May also indicate the error correction level applied to the symbol.

4.22
Version Information
in Model 2 symbols, a function pattern containing information on the symbol version together with error correction bits for this data

5 Symbols (and abbreviated terms)

Mathematical symbols used in formulae and equations are defined after the formula or equation in which they appear.

For the purposes of this specification, the mathematical operations which follow shall apply:

div is the integer division operator

mod is the integer remainder after division

XOR is the exclusive-or logic function whose output is one only when its two inputs are not equivalent. It is represented by the symbol ⊕.

6 Conventions

6.1 Module positions

For ease of reference, module positions are defined by their row and column coordinates in the symbol, in the form (*i, j*) where *i* designates the row (counting from the top downwards) and *j* the column (counting from left to right) in which the module is located, with counting commencing at 0. Module (0, 0) is therefore located at the upper left corner of the symbol.

6.2 Byte notation

Byte contents are shown as hexadecimal values.

6.3 Version references

Symbol versions are referred to in the form Version V-E where V identifies the version number (1 - 40) and E indicates the error correction level (L, M, Q, H).

7 Symbol description

The clauses and subclauses of this International Standard define the specifications applicable to Model 2 QR Code symbols. Unless indicated otherwise in Annex M they also apply to Model 1 symbols.

7.1 Basic characteristics

QR Code is a matrix symbology with the following characteristics:

a) Encodable character set:

1) numeric data (digits **0 - 9**);

2) alphanumeric data (digits **0 - 9**; upper case letters **A -Z**; nine other characters: **space, $ % * + - . / :**);

3) 8-bit byte data (JIS 8-bit character set (Latin and Kana) in accordance with JIS X 0201);

4) Kanji characters (Shift JIS character set in accordance with JIS X 0208 Annex 1 Shift Coded Representation. Note that Kanji characters in QR Code can have values 8140_{HEX} -$9FFC_{HEX}$ and $E040_{HEX}$ - $EBBF_{HEX}$, which can be compacted into 13 bits.)

b) Representation of data:

A dark module is a binary one and a light module is a binary zero.

c) Symbol size (not including quiet zone):

21 × 21 modules to 177 × 177 modules (Versions 1 to 40, increasing in steps of 4 modules per side)

d) Data characters per symbol (for maximum symbol size – Version 40-L):

1) numeric data: 7 089 characters

2) alphanumeric data: 4 296 characters

3) 8-bit byte data: 2 953 characters

4) Kanji data: 1 817 characters

e) Selectable error correction:

Four levels of error correction allowing recovery of:

L 7%

M 15%

Q 25%

H 30%

of the symbol codewords.

f) Code type:

Matrix

g) Orientation independence:

Yes

Figure 1 illustrates a Version 1 QR Code symbol.

Figure 1 — Example of QR Code symbol

7.2 Summary of additional features

The following additional features are either inherent or optional in QR Code:

a) Structured append (optional)

This allows files of data to be represented logically and continuously in up to 16 QR Code symbols. These may be scanned in any sequence to enable the original data to be correctly reconstructed.

b) Masking (inherent)

This enables the ratio of dark to light modules in the symbol to be approximated to 1:1 whilst minimizing the occurrence of arrangements of adjoining modules which would impede efficient decoding.

c) Extended Channel Interpretations (optional)

This mechanism enables data using character sets other than the default encodable set (e.g. Arabic, Cyrillic, Greek) and other data interpretations (e.g. compacted data using defined compression schemes) or other industry-specific requirements to be encoded.

7.3 Symbol structure

Each QR Code symbol shall be constructed of nominally square modules set out in a regular square array and shall consist of a encoding region and function patterns, namely finder, separator, timing patterns, and alignment patterns. Function patterns shall not be used for the encodation of data. The symbol shall be surrounded on all four sides by a quiet zone border. Figure 2 illustrates the structure of a Version 7 QR Code symbol.

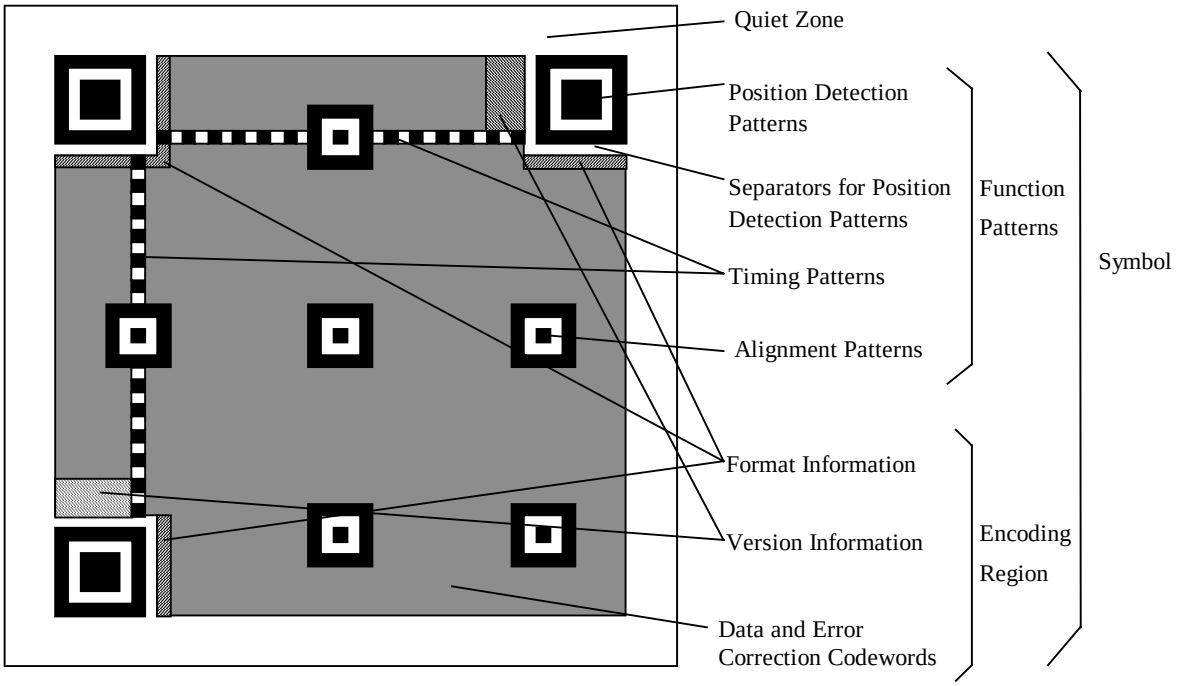


Figure 2 — Structure of a QR Code symbol

7.3.1 Symbol Versions and sizes

There are forty sizes of QR Code symbol referred to as Version 1, Version 2 ... Version 40. Version 1 measures 21 modules × 21 modules, Version 2 measures 25 modules × 25 modules and so on increasing in steps of 4 modules per side up to Version 40 which measures 177 modules × 177 modules. Figures 3 to 8 illustrate the structure of Versions 1, 2, 6, 7, 14, 21 and 40.

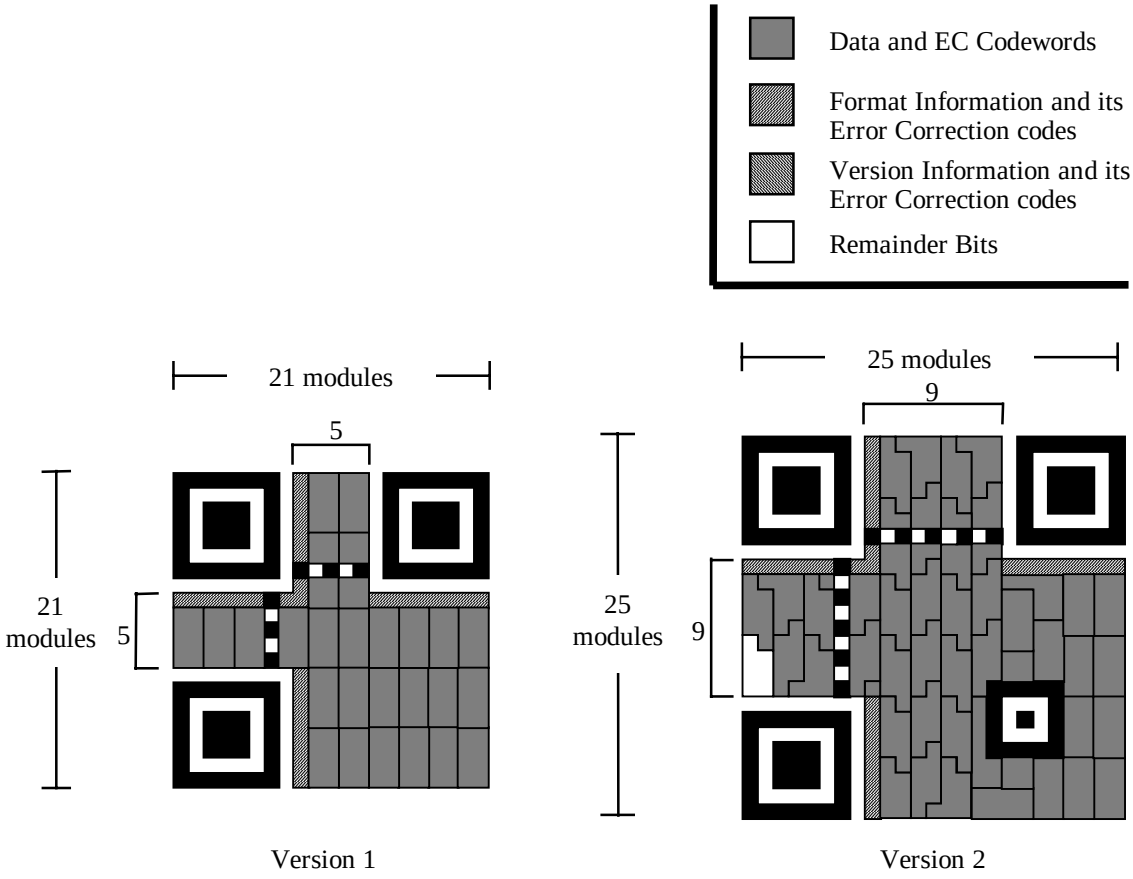
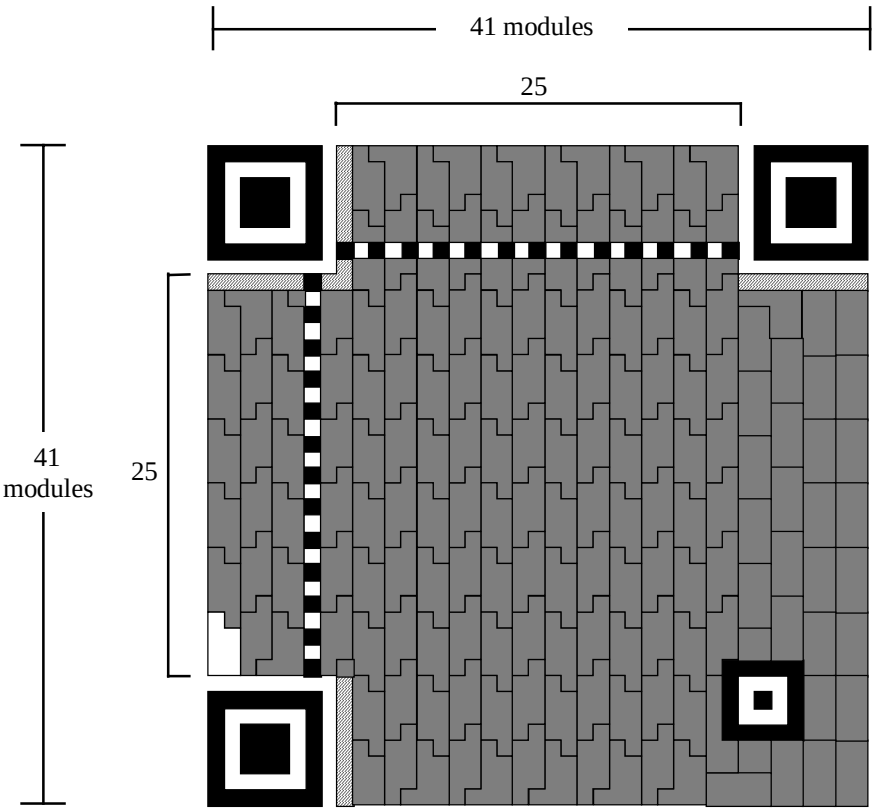
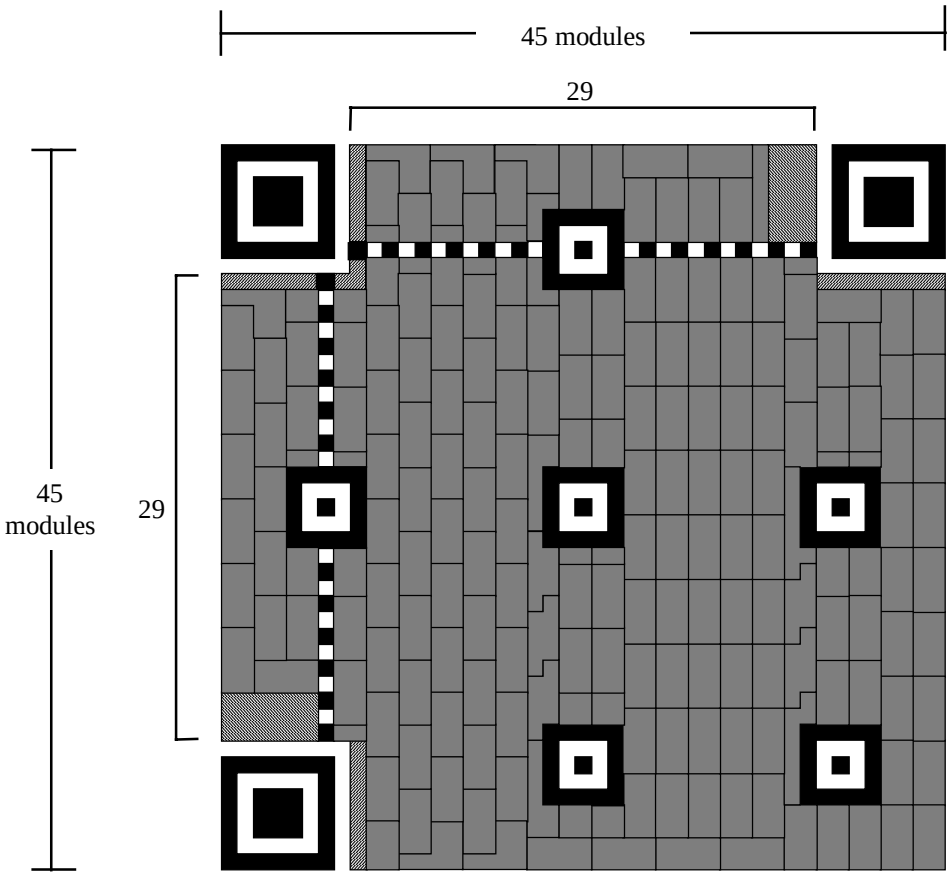


Figure 3 — Version 1 and 2 symbols



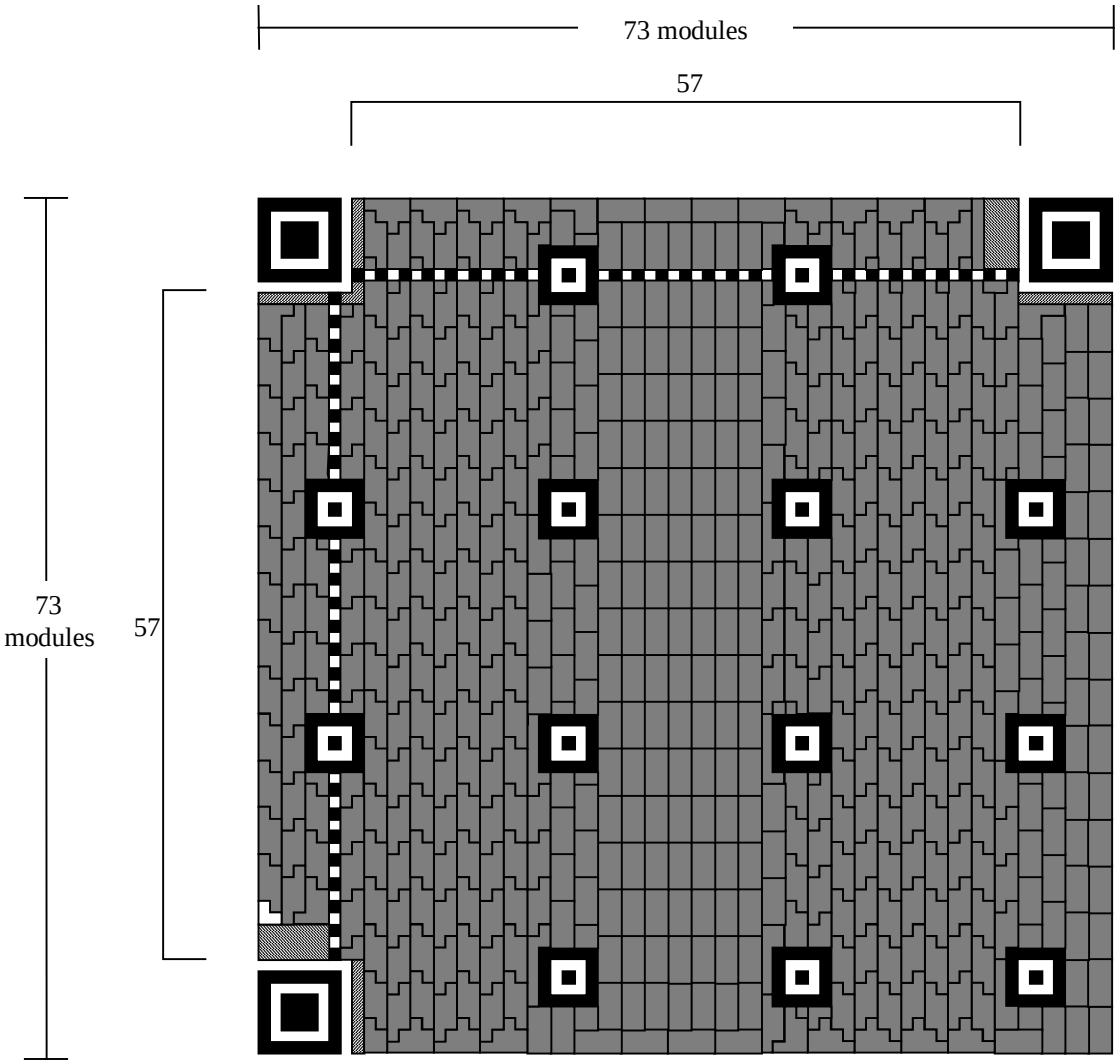
Version 6

Figure 4 — Version 6 symbol



Version 7

Figure 5 — Version 7 symbol



Version 14

Figure 6 — Version 14 symbol

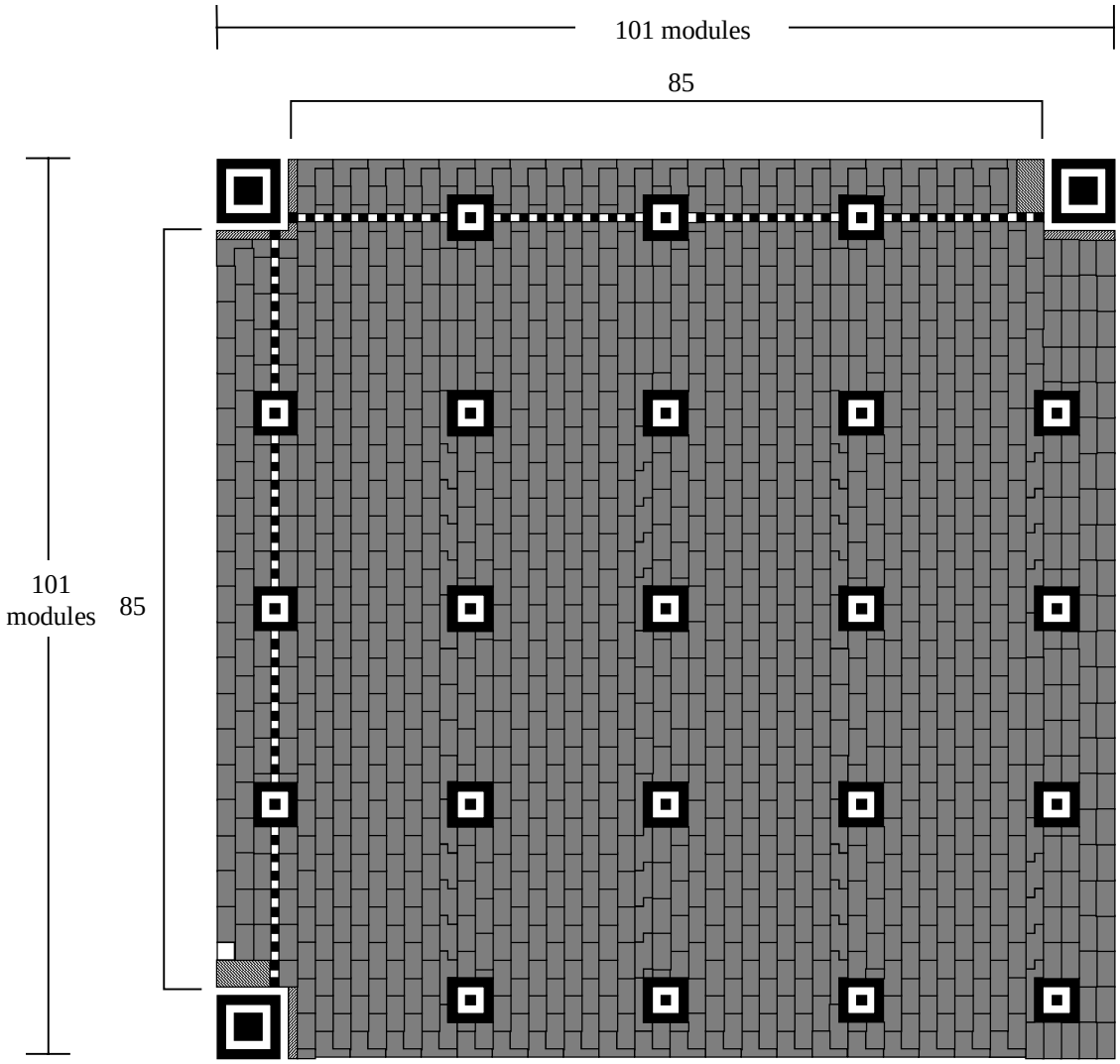


Figure 7 — Version 21 symbol

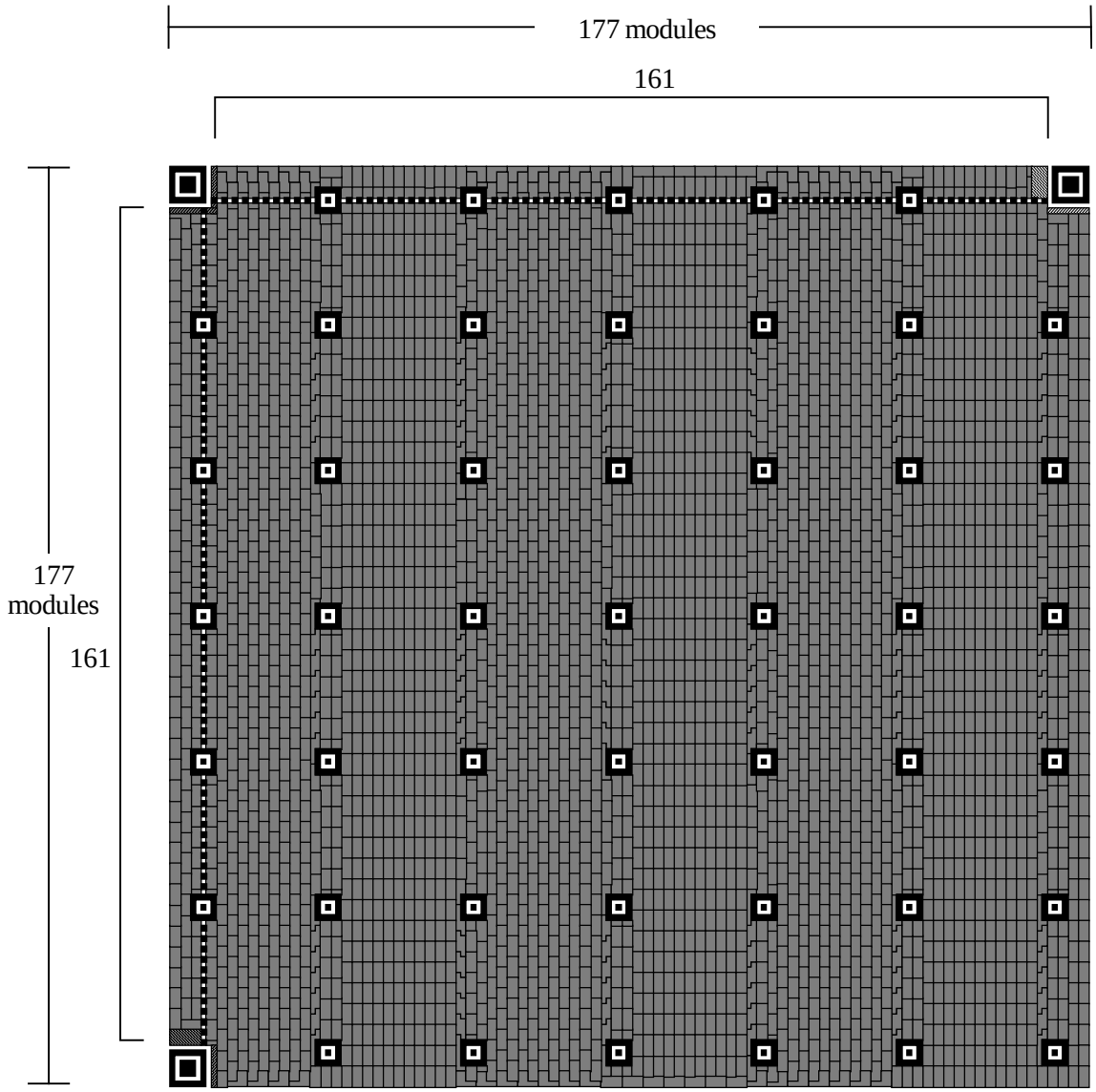


Figure 8 — Version 40 symbol

7.3.2 Finder pattern

The finder pattern shall consist of three identical Position Detection Patterns located at the upper left, upper right and lower left corners of the symbol respectively as illustrated in Figure 2. Each Position Detection Pattern may be viewed as three superimposed concentric squares and is constructed of dark 7 × 7 modules, light 5 × 5 modules and dark 3 × 3 modules. The ratio of module widths in each Position Detection Pattern is 1:1:3:1:1 as illustrated in Figure 9. The symbol is preferentially encoded so that similar patterns have a low probability of being encountered elsewhere in the symbol, enabling rapid identification of a possible QR Code symbol in the field of view. Identification of the three Position Detection Patterns comprising the finder pattern then unambiguously defines the location and orientation of the symbol in the field of view.

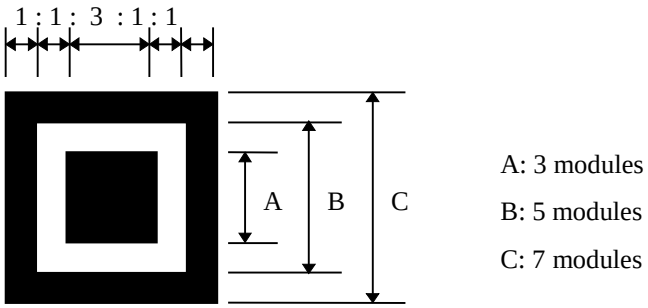


Figure 9 — Structure of Position Detection Pattern

7.3.3 Separators

A one-module wide Separator is placed between each Position Detection Pattern and Encoding Region, as illustrated in Figure 2, and is constructed of all light modules.

7.3.4 Timing Pattern

The horizontal and vertical Timing Patterns respectively consist of a one module wide row or column of alternating dark and light modules, commencing and ending with a dark module. The horizontal Timing Pattern runs across row 6 of the symbol between the separators for the upper Position Detection Patterns; the vertical Timing Pattern similarly runs down column 6 of the symbol between the separators for the left-hand Position Detection Patterns. They enable the symbol density and version to be determined and provide datum positions for determining module coordinates.

7.3.5 Alignment Patterns

Each Alignment Pattern may be viewed as three superimposed concentric squares and is constructed of dark 5 × 5 modules, light 3 × 3 modules and a single central dark module. The number of Alignment Patterns depends on the symbol version and they shall be placed in all Model 2 symbols of Version 2 or larger in positions defined in Annex E.

7.3.6 Encoding region

This region shall contain the symbol characters representing data, those representing error correction codewords, the Version Information and Format Information. Refer to 8.7.1 for details of the symbol characters. Refer to 8.9 for details of the Format Information. Refer to 8.10 for details of the Version Information

7.3.7 Quiet zone

This is a region 4X wide which shall be free of all other markings, surrounding the symbol on all four sides. Its nominal reflectance value shall be equal to that of the light modules.

8 Requirements

8.1 Encode procedure overview

This section provides an overview of the steps required to convert input data to a QR Code symbol.

Step 1 Data analysis

Analyze the input data stream to identify the variety of different characters to be encoded. QR Code supports the Extended Channel Interpretation feature, enabling data differing from the default character set to be encoded. QR Code includes several modes (see 8.3) to allow different sub-sets of characters to be converted into symbol characters in efficient ways. Switch between modes as necessary in order to achieve the most efficient conversion of data into a binary string. Select the required Error Detection and Correction Level. If the user has not specified the symbol version to be used, select the smallest version that will accommodate the data. A complete list of symbol versions and capacities is shown in Table 1.

Step 2 Data encodation

Convert the data characters into a bit stream in accordance with the rules for the mode in force, as defined in 8.4.1 to 8.4.5, inserting Mode Indicators as necessary to change modes at the beginning of each new mode segment, and a Terminator at the end of the data sequence. Split the resulting bit stream into 8-bit codewords. Add Pad Characters as necessary to fill the number of data codewords required for the version.

Step 3 Error correction coding

Divide the codeword sequence into the required number of blocks (as defined in Tables 13 to 22) to enable the error correction algorithms to be processed. Generate the error correction codewords for each block, appending the error correction codewords to the end of the data codeword sequence.

Step 4 Structure final message

Interleave the data and error correction codewords from each block as described in 8.6 (step 3) and add remainder bits as necessary.

Step 5 Module placement in matrix

Place the codeword modules in the matrix together with the Finder Pattern, Separators, Timing Pattern, and Alignment Patterns.

Step 6 Masking

Apply the masking patterns in turn to the encoding region of the symbol. Evaluate the results and select the pattern which optimizes the dark/light module balance and minimizes the occurrence of undesirable patterns.

Step 7 Format and Version Information

Generate the Format and (where applicable) Version Information and complete the symbol.

Table 1 — Data capacity of all versions of QR Code

Version	No. of Modules/ side (A)	Function pattern modules (B)	Format and Version Information modules (C)	Data modules except (C) ($D=A^2-B-C$)	Data capacity [codewords][a] (E)	Remainder Bits
1	21	202	31	208	26	0
2	25	235	31	359	44	7
3	29	243	31	567	70	7
4	33	251	31	807	100	7
5	37	259	31	1 079	134	7
6	41	267	31	1 383	172	7
7	45	390	67	1 568	196	0
8	49	398	67	1 936	242	0
9	53	406	67	2 336	292	0
10	57	414	67	2 768	346	0
11	61	422	67	3 232	404	0
12	65	430	67	3 728	466	0
13	69	438	67	4 256	532	0
14	73	611	67	4 651	581	3
15	77	619	67	5 243	655	3
16	81	627	67	5 867	733	3
17	85	635	67	6 523	815	3
18	89	643	67	7 211	901	3
19	93	651	67	7 931	991	3
20	97	659	67	8 683	1 085	3
21	101	882	67	9 252	1 156	4
22	105	890	67	10 068	1 258	4
23	109	898	67	10 916	1 364	4
24	113	906	67	11 796	1 474	4
25	117	914	67	12 708	1 588	4
26	121	922	67	13 652	1 706	4
27	125	930	67	14 628	1 828	4
28	129	1 203	67	15 371	1 921	3
29	133	1 211	67	16 411	2 051	3
30	137	1 219	67	17 483	2 185	3
31	141	1 227	67	18 587	2 323	3
32	145	1 235	67	19 723	2 465	3
33	149	1 243	67	20 891	2 611	3
34	153	1 251	67	22 091	2 761	3
35	157	1 574	67	23 008	2 876	0
36	161	1 582	67	24 272	3 034	0
37	165	1 590	67	25 568	3 196	0
38	169	1 598	67	26 896	3 362	0
39	173	1 606	67	28 256	3 532	0
40	177	1 614	67	29 648	3 706	0

[a] All codewords shall be 8 bits in length.

8.2 Data analysis

Analyze the input data string to determine its content and select the default or other appropriate ECI and the appropriate mode to encode each sequence as described in 8.4. Each mode in sequence from Numeric mode to

Kanji mode progressively requires more bits per character. It is possible to switch from mode to mode within a symbol in order to minimize the bit stream length for data, parts of which can more efficiently be encoded in one mode than other parts, e.g. numeric sequences followed by alphanumeric sequences. It is in theory most efficient to encode data in the mode requiring the fewest bits per data character, but as there is some overhead in the form of Mode Indicator and Character Count Indicator associated with each mode change, it may not always result in the shortest overall bit stream to change modes for a small number of characters. Guidance on this is given in Annex H. Also, because the capacity of symbols increases in discrete steps from one version to the next, it may not always be necessary to achieve the maximum conversion efficiency in every case.

8.3 Modes

The modes defined below are based on the character values and assignments associated with the default ECI. When any other ECI is in force, the byte values rather than the specific character assignments shall be used to select the optimum data compaction mode. For example, Numeric Mode would be appropriate if there is a sequence of data byte values within the range 30_{HEX} to 39_{HEX} inclusive. In this case the compaction is carried out using the default numeric or alphabetic equivalents of the byte values.

8.3.1 Extended Channel Interpretation (ECI) Mode

The Extended Channel Interpretation (ECI) protocol allows the output data stream to have interpretations different from that of the default character set. The ECI protocol is defined consistently across a number of symbologies. Four broad types of interpretation are supported in QR Code:

a) international character sets (or code pages)

b) general purpose interpretations such as encryption or compaction

c) user-defined interpretations for closed systems.

d) control information for structured append in unbuffered mode

The ECI protocol is fully defined in the AIM ECI specification. The protocol provides a consistent method to specify particular interpretations of byte values before printing and after decoding.

The default interpretation for QR Code is ECI **000020** representing the JIS8 and Shift JIS character sets.

8.3.2 Numeric Mode

Numeric mode encodes data from the decimal digit set (**0 - 9**) (ASCII values 30_{HEX} to 39_{HEX}) at a normal density of 3 data characters per 10 bits.

8.3.3 Alphanumeric Mode

Alphanumeric Mode encodes data from a set of 45 characters, i.e. 10 numeric digits (**0 - 9**) (ASCII values 30_{HEX} to 39_{HEX}), 26 alphabetic characters (**A - Z**) (ASCII values 41_{HEX} to $5A_{HEX}$) , and 9 symbols (**SP**, **$**, **%**, *****, **+**, **-**, **.**, **/**, **:**) (ASCII values 20_{HEX}, 24_{HEX}, 25_{HEX}, $2A_{HEX}$, $2B_{HEX}$, 2D to $2F_{HEX}$, $3A_{HEX}$ respectively). Normally, two input characters are represented by 11 bits.

8.3.4 8-bit Byte Mode

The 8-bit byte mode handles the 8-bit Latin/Kana character set in accordance with JIS X 0201 (character values 00_{HEX} to FF_{HEX}). In this mode data is encoded at a density of 8 bits/character.

8.3.5 Kanji Mode

The Kanji mode handles Kanji characters in accordance with the Shift JIS system based on JIS X 0208. The Shift JIS values are shifted from the JIS X 0208 values. Refer to JIS X 0208 Annex 1 Shift Coded Representation for detail. Each two-byte character value is compacted to a 13 bit binary codeword.

8.3.6 Mixing modes

The QR Code symbol may contain sequences of data in a combination of any of the modes described in 8.3.1 to 8.3.5

Refer to Annex H for guidance on selecting the most efficient way of representing a given input data string in Mixing Mode.

8.3.7 Structured Append Mode

Structured Append mode is used to split the encodation of the data from a message over a number of QR Code symbols. All of the symbols require to be read and the data message can be reconstructed in the correct sequence. The Structured Append header is encoded in each symbol to identify the length of the sequence and the symbol's position in it, and verify that all the symbols read belong to the same message. Refer to 9 for details of encodation in Structured Append mode.

8.3.8 FNC1 Mode

FNC1 mode is used for messages containing data formatted either in accordance with the UCC/EAN Application Identifiers standard or in accordance with a specific industry standard previously agreed with AIM International.

8.4 Data encodation

Input data is converted into a bit stream consisting of an ECI header if the initial ECI is other than the default ECI, followed by one or more segments each in a separate mode. In the default ECI, the bit stream commences with the first Mode Indicator.

The ECI header (if present) shall comprise:

— ECI Mode Indicator (4 bits)

— ECI Designator (8, 16 or 24 bits)

The remainder of the bit stream is then made up of segments each comprising:

— Mode Indicator (4 bits)

— Character Count Indicator

— Data bit stream.

The ECI header shall begin with the first (most significant) bit of the ECI Mode Indicator and end with the final (least significant) bit of the ECI Designator. Each Mode segment shall begin with the first (most significant) bit of the Mode Indicator and end with the final (least significant) bit of the data bit stream. There shall be no explicit separator between segments as their length is defined unambiguously by the rules for the mode in force and the number of input data characters.

To encode a sequence of input data in a given mode, the steps defined in sections 8.4.1 to 8.4.6 shall be followed. Table 2 defines the Mode Indicators for each mode. Table 3 defines the length of the Character Count Indicator, which varies according to the mode and the symbol version in use.

Table 2 — Mode indicators

Mode	Indicator
ECI	**0111**
Numeric	**0001**
Alphanumeric	**0010**
8-bit Byte	**0100**
Kanji	**1000**
Structured Append	**0011**
FNC1	**0101** (First position) **1001** (Second position)
Terminator (End of Message)	**0000**

Table 3 — Number of bits in Character Count Indicator

Version	Numeric Mode	Alphanumeric Mode	8-bit Byte Mode	Kanji Mode
1 to 9	**10**	**9**	**8**	**8**
10 to 26	**12**	**11**	**16**	**10**
27 to 40	**14**	**13**	**16**	**12**

The end of the data in the complete symbol is indicated by a 4 bit terminator **0000**, which is omitted or abbreviated if the remaining symbol capacity after the data bit stream is less than 4 bits. The terminator is not a Mode Indicator as such.

8.4.1 Extended Channel Interpretation (ECI) Mode

This mode, used for encoding data subject to alternative interpretations of byte values (e.g. alternative character sets) in accordance with the AIM ECI specification which defines the pre-processing of this type of data, is invoked by the use of Mode Indicator **0111.** There is no need to invoke the default Extended Channel Interpretation for QR Code (ECI 000020, corresponding to the JIS8/Shift JIS character sets) specifically at the beginning of any symbol.

The Extended Channel Interpretation can only be used with readers enabled to transmit the Symbology Identifier. Readers that cannot transmit the Symbology Identifier cannot transmit the data from any symbol containing an ECI.

Input ECI data shall be handled by the encoding system as a series of 8-bit byte values.

Data in an ECI sequence may be encoded in whatever mode or modes permit the most efficient encoding of the byte values of the data, irrespective of their significance. For example, a sequence of bytes in the range 30_{HEX} to 39_{HEX} could be encoded in Numeric Mode (see 8.4.2) as though it were a sequence of digits 0 – 9 even though it might not actually represent numeric data. In order to determine the value of the Character Count Indicator, the number of bytes (or, in Kanji Mode, of byte pairs) shall be used.

8.4.1.1 ECI Designator

Each Extended Channel Interpretation is designated by a six-digit assignment number which is encoded in the QR Code symbol as the first one, two or three codewords following the ECI Mode Indicator. The encodation rules are defined in Table 4. The ECI Designator appears in the data to be encoded as ASCII/JIS8 character $5C_{HEX}$ (\ or backslash in ISO 646 IRV, ¥ or yen sign in JIS8) followed by the six digit assignment number. Where ASCII/JIS8 character $5C_{HEX}$ appears as true data it shall have been doubled in the data string before encoding in symbols to which the ECI protocol applies.

On decoding, the binary pattern of the first ECI Designator codeword (i.e. the codeword following the Mode Indicator in ECI Mode), determines the length of the ECI Designator sequence. The number of **1** bits before the first **0** bit defines the number of additional codewords after the first used to represent the ECI Assignment number. The bit sequence after the first **0** bit is the binary representation of the ECI Assignment number. The lower numbered ECI assignments may be encoded in multiple ways, but the shortest way is preferred.

Table 4 — Encoding ECI Assignment Number

ECI Assignment Value	No. of Codewords	Codeword values
000000 to 000127	1	**0**bbbbbbb
000000 to 016383	2	**10**bbbbbb bbbbbbbb
000000 to 999999	3	**110**bbbbb bbbbbbbb bbbbbbbb
		where b ... b is the binary value of the ECI Assignment number

Example

Assume data to be encoded is in Greek, using character set ISO 8859-7 (ECI 000009) in version 1-H symbol.

Data to be encoded: ΑΒΓΔΕ (character values $A1_{HEX}$, $A2_{HEX}$, $A3_{HEX}$, $A4_{HEX}$, $A5_{HEX}$)

Bit sequence in symbol:

ECI Mode Indicator **0111**

ECI Assignment number (000009) **00001001**

Mode indicator (8-bit byte) **0100**

Character count indicator (5) **00000101**

Data: **10100001 10100010 10100011 10100100 10100101**

Final bit string: **0111 00001001 0100 00000101 10100001 10100010 10100011 10100100 10100101**

See 15.2 for example of transmission of this data following decoding.

8.4.1.2 Multiple ECIs

Refer to the AIM ECI specification for the rules defining the effect of a subsequent ECI Designator in an ECI data segment. For example, data to which a character set ECI has been applied may also be subject to encryption or compaction using a non-character set ECI which may co-exist with the initial ECI, or the second ECI may have the effect of cancelling the first ECI and starting a new ECI segment. Where any ECI Designator appears in the data, it shall be encoded in the QR Code symbol in accordance with 8.4.1.1 and shall commence a new Mode segment.

8.4.1.3 ECIs and Structured Append

Any ECI(s) invoked shall apply subject to the rules defined above and in the AIM ECI specification until the end of the encoded data or a change of ECI (signaled by Mode Indicator **0111**). If the encoded data in the ECI(s) extends through two or more symbols in Structured Append Mode, it is necessary to provide an ECI header consisting of ECI Mode Indicator and ECI Designator number for each ECI in force, immediately following the Structured Append header, in subsequent symbols in which the ECI continues in force.

8.4.2 Numeric Mode

The input data string is divided into groups of three digits, and each group is converted to its 10 bit binary equivalent. If the number of input digits is not an exact multiple of three, the final one or two digits are converted to 4 or 7 bits respectively. The binary data is then concatenated and prefixed with the Mode Indicator and the Character Count Indicator. The Character Count Indicator in the Numeric Mode has 10, 12 or 14 bits as defined in Table 3. The number of input data characters is converted to its 10, 12 or 14 bit binary equivalent and added after the Mode Indicator and before the binary data sequence.

Example 1 (for Version 1-H symbol)

Input data: **01234567**

1. Divide into groups of three digits: **012 345 67**

2. Convert each group to its binary equivalent: **012 → 0000001100**

345 → 0101011001

67 → 1000011

3. Connect the binary data in sequence: **0000001100 0101011001 1000011**

4. Convert Character Count Indicator to binary (10 bits for version 1-H):

No. of input data characters: **8 → 0000001000**

5. Add Mode Indicator **0001** and Character Count Indicator to binary data:

0001 0000001000 0000001100 0101011001 1000011

Example 2 (for Version 1-H symbol)

Input data: **0123456789012345**

1. Divide into groups of three digits: **012 345 678 901 234 5**

2. Convert each group to its binary equivalent: **012 → 0000001100**

345 → 0101011001

678 → 1010100110

901 → 1110000101

234 → 0011101010

5 → 0101

3. Connect the binary data in sequence:

0000001100 0101011001 1010100110 1110000101 0011101010 0101

4. Convert Character Count Indicator to binary (10 bits for version 1-H):

No. of input data characters: **16 → 0000010000**

5. Add Mode Indicator **0001** and Character Count Indicator to binary data:

0001 0000010000 0000001100 0101011001 1010100110 1110000101 0011101010 0101

For any number of data characters the length of the bit stream in Numeric Mode is given by the following formula:

$B = 4 + C + 10(D \text{ DIV } 3) + R$

where:

B = number of bits in bit stream

C = number of bits in Character Count Indicator (from Table 3)

D = number of input data characters

R = 0 if (D MOD 3) = 0

R = 4 if (*D* MOD 3) = 1

R = 7 if (*D* MOD 3) = 2

8.4.3 Alphanumeric Mode

Each input data character is assigned a character value V from 0 to 44 according to Table 5.

Table 5 — Encoding/decoding table for Alphanumeric Mode

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

Input data characters are divided into groups of two characters which are encoded to 11-bit binary codes. The character value of the first character is multiplied by 45 and the character value of the second digit is added to the product. The sum is then converted to an 11 bit binary number. If the number of input data characters is not a multiple of two, the character value of the final character is encoded to a 6-bit binary number. The binary data is then concatenated and prefixed with the Mode Indicator and the Character Count Indicator. The Character Count Indicator in the Alphanumeric Mode has 9, 11 or 13 bits as defined in Table 3. The number of input data characters is converted to its 9, 11 or 13 bit binary equivalent and added after the Mode Indicator and before the binary data sequence.

Example (for Version 1-H symbol)

Input data: **AC-42**

1. Determine character values according to Table 5. **AC-42 → (10,12,41,4,2)**

2. Divide the result into groups of two decimal values: **(10,12) (41,4) (2)**

3. Convert each group to its 11-bit binary equivalent: **(10,12) 10*45+12 → 462 → 00111001110**

(41,4) 41*45+4 → 1849 → 11100111001

(2) → 2 → 000010

4. Connect the binary data in sequence: **00111001110 11100111001 000010**

5. Convert Character Count Indicator to binary (9 bits for version 1-H):

No. of input data characters: **5 → 000000101**

6. Add Mode Indicator **0010** and Character Count Indicator to binary data:

0010 000000101 00111001110 11100111001 000010

For any number of data characters the length of the bit stream in Alphanumeric Mode is given by the following formula:

$B = 4 + C + 11(D \text{ DIV } 2) + 6(D \text{ MOD } 2)$

where:

B = number of bits in bit stream

C = number of bits in Character Count Indicator (from Table 3)

D = number of input data characters

8.4.4 8-bit Byte Mode

In this mode, one 8 bit codeword directly represents the JIS8 character value of the input data character as shown in Table 6, i.e. a density of 8 bits/character. In ECIs other than the default ECI, it represents an 8-bit byte value directly.

Table 6 — Encoding/decoding table for JIS8 character set

Char.	Hex	Char.	Hex	Char.	Hex	Char.	Hex	Char.	Hex	Char.	Hex	Char.	Hex	Char.	Hex
NUL	00	SP	20	@	40	`	60		80		A0	タ	C0		E0
SOH	01	!	21	A	41	a	61		81	。	A1	チ	C1		E1
STX	02	"	22	B	42	b	62		82	「	A2	ツ	C2		E2
ETX	03	#	23	C	43	c	63		83	」	A3	テ	C3		E3
EOT	04	$	24	D	44	d	64		84	、	A4	ト	C4		E4
ENQ	05	%	25	E	45	e	65		85	・	A5	ナ	C5		E5
ACK	06	&	26	F	46	f	66		86	ヲ	A6	ニ	C6		E6
BEL	07	'	27	G	47	g	67		87	ァ	A7	ヌ	C7		E7
BS	08	(	28	H	48	h	68		88	ィ	A8	ネ	C8		E8
HT	09	)	29	I	49	I	69		89	ゥ	A9	ノ	C9		E9
LF	0A	*	2A	J	4A	j	6A		8A	ェ	AA	ハ	CA		EA
VT	0B	+	2B	K	4B	k	6B		8B	ォ	AB	ヒ	CB		EB
FF	0C	,	2C	L	4C	l	6C		8C	ャ	AC	フ	CC		EC
CR	0D	-	2D	M	4D	m	6D		8D	ュ	AD	ヘ	CD		ED
SO	0E	.	2E	N	4E	n	6E		8E	ョ	AE	ホ	CE		EE
SI	0F	/	2F	O	4F	o	6F		8F	ッ	AF	マ	CF		EF
DLE	10	0	30	P	50	p	70		90	ー	B0	ミ	D0		F0
DC1	11	1	31	Q	51	q	71		91	ア	B1	ム	D1		F1
DC2	12	2	32	R	52	r	72		92	イ	B2	メ	D2		F2
DC3	13	3	33	S	53	s	73		93	ウ	B3	モ	D3		F3
DC4	14	4	34	T	54	t	74		94	エ	B4	ヤ	D4		F4
NAK	15	5	35	U	55	u	75		95	オ	B5	ユ	D5		F5
SYN	16	6	36	V	56	v	76		96	カ	B6	ヨ	D6		F6
ETB	17	7	37	W	57	w	77		97	キ	B7	ラ	D7		F7
CAN	18	8	38	X	58	x	78		98	ク	B8	リ	D8		F8
EM	19	9	39	Y	59	y	79		99	ケ	B9	ル	D9		F9
SUB	1A	:	3A	Z	5A	z	7A		9A	コ	BA	レ	DA		FA
ESC	1B	;	3B	[	5B	{	7B		9B	サ	BB	ロ	DB		FB
FS	1C	<	3C	¥	5C	|	7C		9C	シ	BC	ワ	DC		FC
GS	1D	=	3D	]	5D	}	7D		9D	ス	BD	ン	DD		FD
RS	1E	>	3E	^	5E	¯	7E		9E	セ	BE	゛	DE		FE
US	1F	?	3F	_	5F	DEL	7F		9F	ソ	BF	゜	DF		FF

NOTE 1 In the JIS8 character set byte values 80_{HEX} to $9F_{HEX}$ and $E0_{HEX}$ to FF_{HEX} are not assigned but are reserved values. Some of those values are used as the first byte in the Shift JIS character set and may be used to distinguish between the JIS8 and Shift JIS character sets. Refer to JIS X 0208 Annex 1 Shift Coded Representation for detail.

NOTE 2 Byte values 00_{HEX} to $7F_{HEX}$ in the JIS8 character set correspond to ISO 646 IRV, except values $5C_{HEX}$ and $7E_{HEX}$.

The binary data is then concatenated and prefixed with the Mode Indicator and the Character Count Indicator. The Character Count Indicator in the 8-bit Byte Mode has 8 or 16 bits as defined in Table 3. The number of input data characters is converted to its 8 or 16 bit binary equivalent and added after the Mode Indicator and before the binary data sequence.

For any number of data characters the length of the bit stream in 8-bit Byte Mode is given by the following formula:

$B = 4 + C + 8D$ where:

B = number of bits in bit stream

C = number of bits in Character Count Indicator (from Table 3)

D = number of input data characters

8.4.5 Kanji Mode

In the Shift JIS system, Kanji characters are represented by a two byte combination. These byte values are shifted from the JIS X 0208 values. Refer to JIS X 0208 Annex 1 Shift Coded Representation for detail. Input data characters in Kanji Mode are compacted to 13 bit binary codewords as defined below. The binary data is then concatenated and prefixed with the Mode Indicator and the Character Count Indicator. The Character Count Indicator in the Kanji Mode has 8, 10 or 12 bits as defined in Table 3. The number of input data characters is converted to its 8, 10 or 12 bit binary equivalent and added after the Mode Indicator and before the binary data sequence.

1. For characters with Shift JIS values from 8140_{HEX} to $9FFC_{HEX}$:

a) Subtract 8140_{HEX} from Shift JIS value;

b) Multiply most significant byte of result by $C0_{HEX}$;

c) Add least significant byte to product from b);

d) Convert result to a 13 bit binary string.

2. For characters with Shift JIS values from $E040_{HEX}$ to $EBBF_{HEX}$:

a) Subtract $C140_{HEX}$ from Shift JIS value;

b) Multiply most significant byte of result by $C0_{HEX}$;

c) Add least significant byte to product from b);

d) Convert result to a 13 bit binary string;

Examples

Input character	“点”	“茗”
(Shift JIS value):	**935F**	**E4AA**
1. Subtract 8140 or C140	**935F - 8140 = 121F**	**E4AA - C140 = 236A**
2. Multiply m.s.b. by C0	**12 × C0 = D80**	**23 × C0 = 1A40**
3. Add l.s.b.	**D80 + 1F = D9F**	**1A40 + 6A = 1AAA**
4. Convert to 13 bit binary	**0D9F → 0 1101 1001 1111**	**1AAA →1 1010 1010 1010**

3. For all characters:

e) Prefix binary sequence representing input data characters with Mode Indicator (**1000**) and Character Count Indicator binary equivalent (8, 10 or 12 bits);

For any number of data characters the length of the bit stream in Kanji Mode is given by the following formula:

$B = 4 + C + 13D$

where:

B = number of bits in bit stream

C = number of bits in Character Count Indicator (from Table 3)

D = number of input data characters

8.4.6 Mixing modes

There is the option for a symbol to contain sequences of data in one mode and then to change modes if the data content requires it, or in order to increase the density of encodation. Refer to Annex H for guidance. Each segment of data is encoded in the appropriate mode as indicated in 8.4.1 to 8.4.5, with the basic structure Mode Indicator/Character Count Indicator/Data and followed immediately by the Mode Indicator commencing the next segment. Figure 10 illustrates the structure of data containing n segments.

Segment 1			Segment 2			……	Segment n		
Mode Indicator 1	Character Count Indicator	Data	Mode Indicator 2	Character Count Indicator	Data	……	Mode Indicator n	Character Count Indicator	Data

Figure 10 — Format of mixed mode data

8.4.7 FNC1 Modes

There are two Mode Indicators which are used cumulatively with those defined in 8.3.1 to 8.3.8 and 8.4.1 to 8.4.6 to identify symbols encoding messages formatted according to specific predefined industry or application specifications. These (together with any associated parameter data) precede the Mode Indicator(s) used to encode the data efficiently. When these Mode Indicators are used, it is necessary for the decoder to transmit the Symbology Identifier as defined in 15.1 and Annex F.

8.4.7.1 FNC1 in first position

This Mode Indicator identifies symbols encoding data formatted according to the UCC/EAN Application Identifiers standard. For this purpose, it is only used once in a symbol and shall always be placed immediately before the first Mode Indicator used for efficient data encoding (Numeric, Alphanumeric, 8-bit byte or Kanji), and after any ECI or Structured Append header. Where the UCC/EAN specifications call for the FNC1 character (in other symbologies which use this special character) to be used as a data field separator (i.e. at the end of a variable-length data field), QR Code symbols shall use the **%** character in Alphanumeric Mode or character **GS** (ASCII/JIS8 value 29) in 8-bit Byte Mode to perform this function. If the **%** character occurs as part of the data it shall be encoded as **%%**. Decoders encountering **%** in these symbols shall transmit it as ASCII/JIS8 value 29, and if **%%** is encountered it shall be transmitted as a single **%** character.

Examples

Input data:

0104912345123459 (Application Identifier 01 = UCC/EAN article no., fixed length; data: 04912345123459)

15970331 (Application Identifier 15 = "Best before" date YYMMDD, fixed length; data: 31 March 1997)

30128 (Application Identifier 30 = quantity, variable length; data: 128) (requires separator character)

10ABC123 (Application Identifier 10 = batch number, variable length; data: ABC123)

Data to be encoded:

0104912345123459159703313012 8%10ABC123

Bit sequence in symbol:

0101 (Mode indicator, FNC1 implied in 1st position)

0001 (Mode Indicator, Numeric Mode)

0000011101 (Character Count Indicator, 29)

<data bits for **0104912345123459159703313012 8**>

0010 (Mode Indicator, Alphanumeric Mode)

000001001 (Character Count Indicator, 9)

<data bits for **%10ABC123**>

Transmitted data (see 15.1 and Annex F)

]Q30104912345123459159703313012 8**<ASCII 29>**10ABC123

Example of encoding/transmission of % character in data:

Input data: 123%

Encoded as: 123%%

Transmitted as: 123%

8.4.7.2 FNC1 in second position

This Mode Indicator identifies symbols formatted in accordance with specific industry or application specifications previously agreed with AIM International. It is immediately followed by a one-byte codeword the value of which is that of the Application Indicator assigned to identify the specification concerned by AIM International. For this purpose, it is only used once in a symbol and shall always be placed immediately before the first Mode Indicator used for efficient data encoding (Numeric, Alphanumeric, 8-bit byte or Kanji), and after any ECI or structured Append header. An Application Indicator may take the form of any single Latin alphabetic character from the set {a - z, A - Z} (represented by the ASCII value of the character plus 100) or a two-digit number (represented by its numeric value directly) and shall be transmitted by the decoder as the first one or two characters immediately preceding the data.

Example:

(Application Indicator 37 has not been assigned at the time of publication to any organisation and the data content of the example is purely arbitrary.)

Application Indicator: 37

Input data: AA1234BBB112text text text text<CR>

Bit sequence in symbol:

1001 (Mode Indicator, FNC1 implied in 2nd position)

00100101 (Application Indicator, 37)

0010 (Mode Indicator, Alphanumeric Mode)

000001100 (Character Count Indicator, 12)

<data bits for **AA1234BBB112**>

0100 (Mode Indicator, 8-bit Byte Mode)

00010100 (Character Count Indicator, 20)

<data bits for **text text text text<CR>** >

Transmitted data:

]Q537AA1234BBB112text text text text<CR>

8.4.8 Terminator

The end of data in the symbol is signalled by the Terminator sequence **0000**, appended to the data bit stream following the final mode segment. This may be omitted if the data bit stream completely fills the capacity of the symbol, or abbreviated if the remaining capacity of the symbol is less than 4 bits.

8.4.9 Bit stream to codeword conversion

The bit streams corresponding to each mode segment shall be connected in order. The Terminator shall be appended to the complete bit stream, unless the data bit stream completely fills the capacity of the symbol. The resulting message bit stream shall then be divided into codewords. All codewords are 8 bits in length. If the bit stream length is such that the final codeword is not exactly 8 bits in length, it shall be made 8 bits long by the addition of padding bits with binary value 0. Padding bits shall be added after the final bit (least significant bit) of the data stream. The message bit stream shall then be extended to fill the data capacity of the symbol corresponding to the Version and Error Correction Level, as defined in Tables 7 to 11, by the addition of the Pad Codewords **11101100** and **00010001** alternately. The resulting series of codewords, the data codeword sequence, is then processed as described in 8.5 to add error correction codewords to the message. In certain versions of symbol, it may be necessary to add 3, 4 or 7 Remainder Bits (all zeros) to the end of the message in order exactly to fill the symbol capacity (see Table 1).

Table 7 — Number of symbol characters and input data capacity for versions 1 to 8

Version	Error correction level	Number of data codewords[a]	Number of data bits[b]	Data capacity			
				Numeric	Alphanumeric	8-bit Byte	Kanji
1	L	19	152	41	25	17	10
	M	16	128	34	20	14	8
	Q	13	104	27	16	11	7
	H	9	72	17	10	7	4
2	L	34	272	77	47	32	20
	M	28	224	63	38	26	16
	Q	22	176	48	29	20	12
	H	16	128	34	20	14	8
3	L	55	440	127	77	53	32
	M	44	352	101	61	42	26
	Q	34	272	77	47	32	20
	H	26	208	58	35	24	15
4	L	80	640	187	114	78	48
	M	64	512	149	90	62	38
	Q	48	384	111	67	46	28
	H	36	288	82	50	34	21
5	L	108	864	255	154	106	65
	M	86	688	202	122	84	52
	Q	62	496	144	87	60	37
	H	46	368	106	64	44	27
6	L	136	1 088	322	195	134	82
	M	108	864	255	154	106	65
	Q	76	608	178	108	74	45
	H	60	480	139	84	58	36
7	L	156	1 248	370	224	154	95
	M	124	992	293	178	122	75
	Q	88	704	207	125	86	53
	H	66	528	154	93	64	39
8	L	194	1 552	461	279	192	118
	M	154	1 232	365	221	152	93
	Q	110	880	259	157	108	66
	H	86	688	202	122	84	52

[a] All codewords shall be 8 bits in length.

[b] The number of Data Bits includes bits for Mode Indicator and Character Count Indicator.

Table 8 — Number of symbol characters and input data capacity for versions 9 to 16

Version	Error correction level	Number of data codewords[a]	Number of data bits[b]	Data capacity			
				Numeric	Alphanumeric	8-bit Byte	Kanji
9	L	232	1 856	552	335	230	141
	M	182	1 456	432	262	180	111
	Q	132	1 056	312	189	130	80
	H	100	800	235	143	98	60
10	L	274	2 192	652	395	271	167
	M	216	1 728	513	311	213	131
	Q	154	1 232	364	221	151	93
	H	122	976	288	174	119	74
11	L	324	2 592	772	468	321	198
	M	254	2 032	604	366	251	155
	Q	180	1 440	427	259	177	109
	H	140	1 120	331	200	137	85
12	L	370	2 960	883	535	367	226
	M	290	2 320	691	419	287	177
	Q	206	1 648	489	296	203	125
	H	158	1 264	374	227	155	96
13	L	428	3 424	1 022	619	425	262
	M	334	2 672	796	483	331	204
	Q	244	1 952	580	352	241	149
	H	180	1 440	427	259	177	109
14	L	461	3 688	1 101	667	458	282
	M	365	2 920	871	528	362	223
	Q	261	2 088	621	376	258	159
	H	197	1 576	468	283	194	120
15	L	523	4 184	1 250	758	520	320
	M	415	3 320	991	600	412	254
	Q	295	2 360	703	426	292	180
	H	223	1 784	530	321	220	136
16	L	589	4 712	1 408	854	586	361
	M	453	3 624	1 082	656	450	277
	Q	325	2 600	775	470	322	198
	H	253	2 024	602	365	250	154

[a] All codewords shall be 8 bits in length.

[b] The number of Data Bits includes bits for Mode Indicator and Character Count Indicator.

Table 9 — Number of symbol characters and input data capacity for versions 17 to 24

Version	Error correction level	Number of data codewords[a]	Number of data bits[b]	Data capacity			
				Numeric	Alphanumeric	8-bit Byte	Kanji
17	L	647	5 176	1 548	938	644	397
	M	507	4 056	1 212	734	504	310
	Q	367	2 936	876	531	364	224
	H	283	2 264	674	408	280	173
18	L	721	5 768	1 725	1 046	718	442
	M	563	4 504	1 346	816	560	345
	Q	397	3 176	948	574	394	243
	H	313	2 504	746	452	310	191
19	L	795	6 360	1 903	1 153	792	488
	M	627	5 016	1 500	909	624	384
	Q	445	3 560	1 063	644	442	272
	H	341	2 728	813	493	338	208
20	L	861	6 888	2 061	1 249	858	528
	M	669	5 352	1 600	970	666	410
	Q	485	3 880	1 159	702	482	297
	H	385	3 080	919	557	382	235
21	L	932	7 456	2 232	1 352	929	572
	M	714	5 712	1 708	1 035	711	438
	Q	512	4 096	1 224	742	509	314
	H	406	3 248	969	587	403	248
22	L	1 006	8 048	2 409	1 460	1 003	618
	M	782	6 256	1 872	1 134	779	480
	Q	568	4 544	1 358	823	565	348
	H	442	3 536	1 056	640	439	270
23	L	1 094	8 752	2 620	1 588	1 091	672
	M	860	6 880	2 059	1 248	857	528
	Q	614	4 912	1 468	890	611	376
	H	464	3 712	1 108	672	461	284
24	L	1 174	9 392	2 812	1 704	1 171	721
	M	914	7 312	2 188	1 326	911	561
	Q	664	5 312	1 588	963	661	407
	H	514	4 112	1 228	744	511	315

[a] All codewords shall be 8 bits in length.

[b] The number of Data Bits includes bits for Mode Indicator and Character Count Indicator.

Table 10 — Number of symbol characters and input data capacity for versions 25 to 32

Version	Error correction level	Number of data codewords[a]	Number of data bits[b]	Data capacity			
				Numeric	Alphanumeric	8-bit Byte	Kanji
25	L	1 276	10 208	3 057	1 853	1 273	784
	M	1 000	8 000	2 395	1 451	997	614
	Q	718	5 744	1 718	1 041	715	440
	H	538	4 304	1 286	779	535	330
26	L	1 370	10 960	3 283	1 990	1 367	842
	M	1 062	8 496	2 544	1 542	1 059	652
	Q	754	6 032	1 804	1 094	751	462
	H	596	4 768	1 425	864	593	365
27	L	1 468	11 744	3 517	2 132	1 465	902
	M	1 128	9 024	2 701	1 637	1 125	692
	Q	808	6 464	1 933	1 172	805	496
	H	628	5 024	1 501	910	625	385
28	L	1 531	12 248	3 669	2 223	1 528	940
	M	1 193	9 544	2 857	1 732	1 190	732
	Q	871	6 968	2 085	1 263	868	534
	H	661	5 288	1 581	958	658	405
29	L	1 631	13 048	3 909	2 369	1 628	1 002
	M	1 267	10 136	3 035	1 839	1 264	778
	Q	911	7 288	2 181	1 322	908	559
	H	701	5 608	1 677	1 016	698	430
30	L	1 735	13 880	4 158	2 520	1 732	1 066
	M	1 373	10 984	3 289	1 994	1 370	843
	Q	985	7 880	2 358	1 429	982	604
	H	745	5 960	1 782	1 080	742	457
31	L	1 843	14 744	4 417	2 677	1 840	1 132
	M	1 455	11 640	3 486	2 113	1 452	894
	Q	1 033	8 264	2 473	1 499	1 030	634
	H	793	6 344	1 897	1 150	790	486
32	L	1 955	15 640	4 686	2 840	1 952	1 201
	M	1 541	12 328	3 693	2 238	1 538	947
	Q	1 115	8 920	2 670	1 618	1 112	684
	H	845	6 760	2 022	1 226	842	518

[a] All codewords shall be 8 bits in length.

[b] The number of Data Bits includes bits for Mode Indicator and Character Count Indicator.

Table 11 — Number of symbol characters and input data capacity for versions 33 to 40

Version	Error correction level	Number of data codewords[a]	Number of data bits[b]	Data capacity			
				Numeric	Alphanumeric	8-bit Byte	Kanji
33	L	2 071	16 568	4 965	3 009	2 068	1 273
	M	1 631	13 048	3 909	2 369	1 628	1 002
	Q	1 171	9 368	2 805	1 700	1 168	719
	H	901	7 208	2 157	1 307	898	553
34	L	2 191	17 528	5 253	3 183	2 188	1 347
	M	1 725	13 800	4 134	2 506	1 722	1 060
	Q	1 231	9 848	2 949	1 787	1 228	756
	H	961	7 688	2 301	1 394	958	590
35	L	2 306	18 448	5 529	3 351	2 303	1 417
	M	1 812	14 496	4 343	2 632	1 809	1 113
	Q	1 286	10 288	3 081	1 867	1 283	790
	H	986	7 888	2 361	1 431	983	605
36	L	2 434	19 472	5 836	3 537	2 431	1 496
	M	1 914	15 312	4 588	2 780	1 911	1 176
	Q	1 354	10 832	3 244	1 966	1 351	832
	H	1 054	8 432	2 524	1 530	1 051	647
37	L	2 566	20 528	6 153	3 729	2 563	1 577
	M	1 992	15 936	4 775	2 894	1 989	1 224
	Q	1 426	11 408	3 417	2 071	1 423	876
	H	1 096	8 768	2 625	1 591	1 093	673
38	L	2 702	21 616	6 479	3 927	2 699	1 661
	M	2 102	16 816	5 039	3 054	2 099	1 292
	Q	1 502	12 016	3 599	2 181	1 499	923
	H	1 142	9 136	2 735	1 658	1 139	701
39	L	2 812	22 496	6 743	4 087	2 809	1 729
	M	2 216	17 728	5 313	3 220	2 213	1 362
	Q	1 582	12 656	3 791	2 298	1 579	972
	H	1 222	9 776	2 927	1 774	1 219	750
40	L	2 956	23 648	7 089	4 296	2 953	1 817
	M	2 334	18 672	5 596	3 391	2 331	1 435
	Q	1 666	13 328	3 993	2 420	1 663	1 024
	H	1 276	10 208	3 057	1 852	1 273	784

[a] All codewords shall be 8 bits in length.

[b] The number of Data Bits includes bits for Mode Indicator and Character Count Indicator.

8.5 Error correction

8.5.1 Error correction capacity

QR Code employs error correction to generate a series of error correction codewords which are added to the data codeword sequence in order to enable the symbol to withstand damage without loss of data. There are four user-selectable levels of error correction, as shown in Table 12, offering the capability of recovery from the following amounts of damage:

Table 12 — Error correction levels

Error Correction Level	Recovery Capacity % (approx.)
L	7
M	15
Q	25
H	30

Clause I.3 gives guidance on the appropriate level of error correction to be applied to a symbol.

The error correction codewords can correct two types of erroneous codewords, erasures (erroneous codewords at known locations) and errors (erroneous codewords at unknown locations). An erasure is an unscanned or undecodable symbol character. An error is a misdecoded symbol character. Since QR Code is a matrix symbology, a defect converting a module from dark to light or vice versa will result in the affected symbol character misdecoding as an apparently valid but different codeword. Such an error causing a substitution error in the data requires two error correction codewords to correct it.

The number of erasures and errors correctable is given by the following formula:

$$e + 2t \leq d - p$$

where:

e = number of erasures

t = number of errors

d = number of error correction codewords

p = number of misdecode protection codewords

For example, in a version 6-H symbol there is a total of 172 codewords, of which 112 are error correction codewords (leaving 60 data codewords). The 112 error correction codewords can correct 56 misdecodes or substitution errors, i.e. 56/172 or 32.6% of the symbol capacity

In the formula above, p = 3 in version 1-L symbols, p = 2 in version 1-M and 2-L symbols, p = 1 in version 1-Q, 1-H and 3-L symbols, p = 0 in all other cases. Where $p > 0$ there are p (i.e. 1, 2 or 3) codewords which act as error detection codewords and prevent transmission of data from symbols where the number of errors exceeds the error correction capacity, e must be less than $d/2$. In a Version 2-L symbol, for example, the total number of codewords is 44; of these, 34 are data codewords and 10 error correction codewords. From Table 13 it can be seen that the error correction capacity is 4 errors (where e = 0). Substituting in the formula above,

$$0 + (2 \times 4) = 10 - 2$$

meaning that the correction of the 4 errors requires only 8 error correction codewords; the remaining 2 error correction codewords can therefore detect (but not correct) any additional errors and the symbol would, if there were more than 4 errors, fail to decode.

Depending on the Version and Error Correction Level, the data codeword sequence shall be subdivided into one or more blocks, to each of which the error correction algorithm shall be applied separately. Tables 13 to 22 list, for each version and Error Correction Level, the total number of codewords, the total number of error correction codewords, and the structure and number of error correction blocks.

If Remainder Bits are required to fill remaining modules in the symbol capacity for certain symbol versions they shall all be 0 bits.

Table 13 — Error correction characteristics for versions 1 to 6

Version	Total number of codewords	Error correction level	Number of error correction codewords	Number of error correction blocks	Error correction code per block [a]
1	26	L	7	1	(26,19,2) [b]
		M	10	1	(26,16,4) [b]
		Q	13	1	(26,13,6) [b]
		H	17	1	(26,9,8) [b]
2	44	L	10	1	(44,34,4) [b]
		M	16	1	(44,28,8)
		Q	22	1	(44,22,11)
		H	28	1	(44,16,14)
3	70	L	15	1	(70,55,7) [b]
		M	26	1	(70,44,13)
		Q	36	2	(35,17,9)
		H	44	2	(35,13,11)
4	100	L	20	1	(100,80,10)
		M	36	2	(50,32,9)
		Q	52	2	(50,24,13)
		H	64	4	(25,9,8)
5	134	L	26	1	(134,108,13)
		M	48	2	(67,43,12)
		Q	72	2 2	(33,15,9) (34,16,9)
		H	88	2 2	(33,11,11) (34,12,11)
6	172	L	36	2	(86,68,9)
		M	64	4	(43,27,8)
		Q	96	4	(43,19,12)
		H	112	4	(43,15,14)

[a] (c, k, r): c = total number of codewords
k = number of data codewords
r = number of error correction capacity

[b] Error correction capacity is less than half the number of error correction codewords to reduce the probability of misdecodes.

Table 14 — Error correction characteristics for versions 7 to 10

Version	Total number of codewords	Error correction level	Number of error correction codewords	Number of error correction blocks	Error correction code per block [a]
7	196	L	40	2	(98,78,10)
		M	72	4	(49,31,9)
		Q	108	2 4	(32,14,9) (33,15,9)
		H	130	4 1	(39,13,13) (40,14,13)
8	242	L	48	2	(121,97,12)
		M	88	2 2	(60,38,11) (61,39,11)
		Q	132	4 2	(40,18,11) (41,19,11)
		H	156	4 2	(40,14,13) (41,15,13)
9	292	L	60	2	(146,116,15)
		M	110	3 2	(58,36,11) (59,37,11)
		Q	160	4 4	(36,16,10) (37,17,10)
		H	192	4 4	(36,12,12) (37,13,12)
10	346	L	72	2 2	(86,68,9) (87,69,9)
		M	130	4 1	(69,43,13) (70,44,13)
		Q	192	6 2	(43,19,12) (44,20,12)
		H	224	6 2	(43,15,14) (44,16,14)

[a] (c, k, r): c = total number of codewords
k = number of data codewords
r = number of error correction capacity

Table 15 — Error correction characteristics for Model 2, versions 11 to 14

Version	Total number of codewords	Error correction level	Number of error correction codewords	Number of error correction blocks	Error correction code per block [a]
11	404	L	80	4	(101,81,10)
		M	150	1 4	(80,50,15) (81,51,15)
		Q	224	4 4	(50,22,14) (51,23,14)
		H	264	3 8	(36,12,12) (37,13,12)
12	466	L	96	2 2	(116,92,12) (117,93,12)
		M	176	6 2	(58,36,11) (59,37,11)
		Q	260	4 6	(46,20,13) (47,21,13)
		H	308	7 4	(42,14,14) (43,15,14)
13	532	L	104	4	(133,107,13)
		M	198	8 1	(59,37,11) (60,38,11)
		Q	288	8 4	(44,20,12) (45,21,12)
		H	352	12 4	(33,11,11) (34,12,11)
14	581	L	120	3 1	(145,115,15) (146,116,15)
		M	216	4 5	(64,40,12) (65,41,12)
		Q	320	11 5	(36,16,10) (37,17,10)
		H	384	11 5	(36,12,12) (37,13,12)

[a] (c, k, r): c = total number of codewords
k = number of data codewords
r = number of error correction capacity

Table 16 — Error correction characteristics for versions 15 to 18

Version	Total number of codewords	Error correction level	Number of error correction codewords	Number of error correction blocks	Error correction code per block [a]
15	655	L	132	5 1	(109,87,11) (110,88,11)
		M	240	5 5	(65,41,12) (66,42,12)
		Q	360	5 7	(54,24,15) (55,25,15)
		H	432	11 7	(36,12,12) (37,13,12)
16	733	L	144	5 1	(122,98,12) (123,99,12)
		M	280	7 3	(73,45,14) (74,46,14)
		Q	408	15 2	(43,19,12) (44,20,12)
		H	480	3 13	(45,15,15) (46,16,15)
17	815	L	168	1 5	(135,107,14) (136,108,14)
		M	308	10 1	(74,46,14) (75,47,14)
		Q	448	1 15	(50,22,14) (51,23,14)
		H	532	2 17	(42,14,14) (43,15,14)
18	901	L	180	5 1	(150,120,15) (151,121,15)
		M	338	9 4	(69,43,13) (70,44,13)
		Q	504	17 1	(50,22,14) (51,23,14)
		H	588	2 19	(42,14,14) (43,15,14)

[a] (c, k, r): c = total number of codewords
k = number of data codewords
r = number of error correction capacity

Table 17 — Error correction characteristics for versions 19 to 22

Version	Total number of codewords	Error correction level	Number of error correction codewords	Number of error correction blocks	Error correction code per block [a]
19	991	L	196	3 4	(141,113,14) (142,114,14)
		M	364	3 11	(70,44,13) (71,45,13)
		Q	546	17 4	(47,21,13) (48,22,13)
		H	650	9 16	(39,13,13) (40,14,13)
20	1 085	L	224	3 5	(135,107,14) (136,108,14)
		M	416	3 13	(67,41,13) (68,42,13)
		Q	600	15 5	(54,24,15) (55,25,15)
		H	700	15 10	(43,15,14) (44,16,14)
21	1 156	L	224	4 4	(144,116,14) (145,117,14)
		M	442	17	(68,42,13)
		Q	644	17 6	(50,22,14) (51,23,14)
		H	750	19 6	(46,16,15) (47,17,15)
22	1 258	L	252	2 7	(139,111,14) (140,112,14)
		M	476	17	(74,46,14)
		Q	690	7 16	(54,24,15) (55,25,15)
		H	816	34	(37,13,12)

[a] (c, k, r): c = total number of codewords
k = number of data codewords
r = number of error correction capacity

Table 18 — Error correction characteristics for versions 23 to 26

Version	Total number of codewords	Error correction level	Number of error correction codewords	Number of error correction blocks	Error correction code per block [a]
23	1 364	L	270	4 5	(151,121,15) (152,122,15)
		M	504	4 14	(75,47,14) (76,48,14)
		Q	750	11 14	(54,24,15) (55,25,15)
		H	900	16 14	(45,15,15) (46,16,15)
24	1 474	L	300	6 4	(147,117,15) (148,118,15)
		M	560	6 14	(73,45,14) (74,46,14)
		Q	810	11 16	(54,24,15) (55,25,15)
		H	960	30 2	(46,16,15) (47,17,15)
25	1 588	L	312	8 4	(132,106,13) (133,107,13)
		M	588	8 13	(75,47,14) (76,48,14)
		Q	870	7 22	(54,24,15) (55,25,15)
		H	1050	22 13	(45,15,15) (46,16,15)
26	1 706	L	336	10 2	(142,114,14) (143,115,14)
		M	644	19 4	(74,46,14) (75,47,14)
		Q	952	28 6	(50,22,14) (51,23,14)
		H	1110	33 4	(46,16,15) (47,17,15)

[a] (c, k, r): c = total number of codewords
k = number of data codewords
r = number of error correction capacity

Table 19 — Error correction characteristics for versions 27 to 30

Version	Total number of codewords	Error correction level	Number of error correction codewords	Number of error correction blocks	Error correction code per block [a]
27	1 828	L	360	8 4	(152,122,15) (153,123,15)
		M	700	22 3	(73,45,14) (74,46,14)
		Q	1 020	8 26	(53,23,15) (54,24,15)
		H	1 200	12 28	(45,15,15) (46,16,15)
28	1 921	L	390	3 10	(147,117,15) (148,118,15)
		M	728	3 23	(73,45,14) (74,46,14)
		Q	1 050	4 31	(54,24,15) (55,25,15)
		H	1 260	11 31	(45,15,15) (46,16,15)
29	2 051	L	420	7 7	(146,116,15) (147,117,15)
		M	784	21 7	(73,45,14) (74,46,14)
		Q	1 140	1 37	(53,23,15) (54,24,15)
		H	1 350	19 26	(45,15,15) (46,16,15)
30	2 185	L	450	5 10	(145,115,15) (146,116,15)
		M	812	19 10	(75,47,14) (76,48,14)
		Q	1 200	15 25	(54,24,15) (55,25,15)
		H	1 440	23 25	(45,15,15) (46,16,15)

[a] (c, k, r): c = total number of codewords
k = number of data codewords
r = number of error correction capacity

Table 20 — Error correction characteristics for versions 31 to 34

Version	Total number of codewords	Error correction level	Number of error correction codewords	Number of error correction blocks	Error correction code per block [a]
31	2 323	L	480	13 3	(145,115,15) (146,116,15)
		M	868	2 29	(74,46,14) (75,47,14)
		Q	1 290	42 1	(54,24,15) (55,25,15)
		H	1 530	23 28	(45,15,15) (46,16,15)
32	2 465	L	510	17	(145,115,15)
		M	924	10 23	(74,46,14) (75,47,14)
		Q	1 350	10 35	(54,24,15) (55,25,15)
		H	1 620	19 35	(45,15,15) (46,16,15)
33	2 611	L	540	17 1	(145,115,15) (146,116,15)
		M	980	14 21	(74,46,14) (75,47,14)
		Q	1 440	29 19	(54,24,15) (55,25,15)
		H	1 710	11 46	(45,15,15) (46,16,15)
34	2 761	L	570	13 6	(145,115,15) (146,116,15)
		M	1 036	14 23	(74,46,14) (75,47,14)
		Q	1 530	44 7	(54,24,15) (55,25,15)
		H	1 800	59 1	(46,16,15) (47,17,15)

[a] (c, k, r): c = total number of codewords
k = number of data codewords
r = number of error correction capacity

Table 21 — Error correction characteristics for versions 35 to 38

Version	Total number of codewords	Error correction level	Number of error correction codewords	Number of error correction blocks	Error correction code per block [a]
35	2 876	L	570	12 7	(151,121,15) (152,122,15)
		M	1 064	12 26	(75,47,14) (76,48,14)
		Q	1 590	39 14	(54,24,15) (55,25,15)
		H	1 890	22 41	(45,15,15) (46,16,15)
36	3 034	L	600	6 14	(151,121,15) (152,122,15)
		M	1 120	6 34	(75,47,14) (76,48,14)
		Q	1 680	46 10	(54,24,15) (55,25,15)
		H	1 980	2 64	(45,15,15) (46,16,15)
37	3 196	L	630	17 4	(152,122,15) (153,123,15)
		M	1 204	29 14	(74,46,14) (75,47,14)
		Q	1 770	49 10	(54,24,15) (55,25,15)
		H	2 100	24 46	(45,15,15) (46,16,15)
38	3 362	L	660	4 18	(152,122,15) (153,123,15)
		M	1 260	13 32	(74,46,14) (75,47,14)
		Q	1 860	48 14	(54,24,15) (55,25,15)
		H	2 220	42 32	(45,15,15) (46,16,15)

[a] (c, k, r): c = total number of codewords
k = number of data codewords
r = number of error correction capacity

Table 22 — Error correction characteristics for versions 39 to 40

Version	Total number of codewords	Error correction level	Number of error correction codewords	Number of error correction blocks	Error correction code per block [a]
39	3 532	L	720	20 4	(147,117,15) (148,118,15)
		M	1 316	40 7	(75,47,14) (76,48,14)
		Q	1 950	43 22	(54,24,15) (55,25,15)
		H	2 310	10 67	(45,15,15) (46,16,15)
40	3 706	L	750	19 6	(148,118,15) (149,119,15)
		M	1 372	18 31	(75,47,14) (76,48,14)
		Q	2 040	34 34	(54,24,15) (55,25,15)
		H	2 430	20 61	(45,15,15) (46,16,15)

[a] (c, k, r): c = total number of codewords
k = number of data codewords
r = number of error correction capacity

8.5.2 Generating the error correction codewords

The data codewords including Pad codewords as necessary shall be divided into the number of blocks shown in Tables 13 to 22. Error correction codewords shall be calculated for each block and appended to the data codewords.

The polynomial arithmetic for QR Code shall be calculated using bit-wise modulo 2 arithmetic and byte-wise modulo 100011101 arithmetic. This is a Galois field of 2^8 with 100011101 representing the field's prime modulus polynomial $x^8 + x^4 + x^3 + x^2 + 1$.

The data codewords are the coefficients of the terms of a polynomial with the coefficient of the highest term being the first data codeword and that of the lowest power term being the last data codeword before the first error correction codeword.

The error correction codewords are the remainder after dividing the data codewords by a polynomial *g(x)* used for error correction codes (see Annex A). The highest order coefficient of the remainder is the first error correction codeword and the zero power coefficient is the last error correction codeword and the last codeword in the block.

Thirty-one different generator polynomials are used for generating the error correction codewords. These are given in Annex A.1.

This can be implemented by using the division circuit as shown in Figure 11. The registers b_0 through b_{k-1} are initialized as zeros. There are two phases to generate the encoding. In the first phase, with the switch in the down position the data codewords are passed both to the output and the circuit. The first phase is complete after *n* clock pulses. In the second phase (*n*+1 ... *n*+*k* clock pulses), with the switch in the up position, the error correction codewords ε_{k-1} ... ε_0 are generated by flushing the registers in order while keeping the data input at 0.

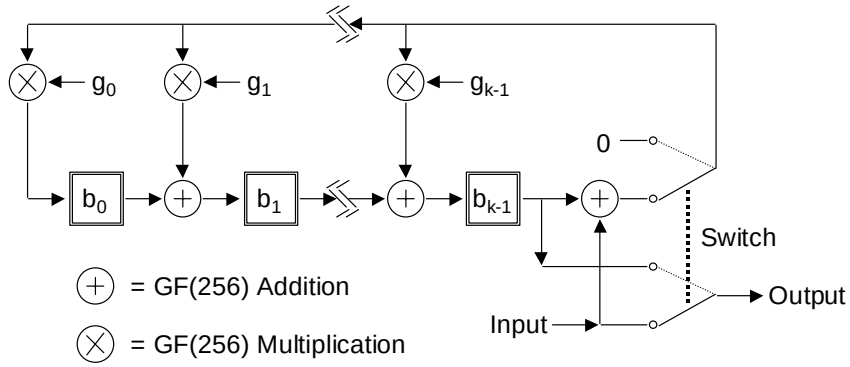


Figure 11 — Error correction codeword encoding circuit

8.6 Constructing the final message codeword sequence

The total number of codewords in the message shall always be equal to the total number of codewords capable of being represented in the symbol, as shown in Tables 7 to 12 and 13 to 22.

The following steps shall be followed to construct the final sequence of codewords (data plus error correction codewords plus Remainder Codewords if necessary):

1. Divide the data codeword sequence into *n* blocks as defined in Tables 13 to 22 according to the version and error correction level.

2. For each data block, calculate a corresponding block of error correction codewords as defined in 8.5.2 and Annex A.

3. Assemble the final sequence by taking data and error correction codewords from each block in turn: data block 1, codeword 1; data block 2, codeword 1; data block 3, codeword 1; and similarly to data block *n* - 1, final codeword; data block *n*, final codeword; then error correction block 1, codeword 1, error correction block 2, codeword 1, ... and similarly to error correction block *n* - 1, final codeword; error correction block *n*, final codeword. QR Code symbols contain data and error correction blocks which always exactly fill the symbol codeword capacity. In certain versions, however, there may be a need for 3, 4 or 7 Remainder Bits to be appended to the final message bit stream in order exactly to fill the number of modules in the encoding region.

The shortest data block (or blocks) shall be placed first in the sequence and all the data codewords shall be placed in the symbol before the first error correction codeword. For example, the Version 5-H symbol comprises four data and four error correction blocks, the first two of each of which contain 11 data and 22 error correction codewords respectively, while the third and fourth pairs of blocks contain 12 data and 22 error correction codewords respectively. In this symbol, the character arrangement can be depicted as follows. Each row of the table corresponds to one block of data codewords (shown as Dn) followed by the associated block of error correction codewords (shown as En); the sequence of character placement in the symbol is obtained by reading down each column of the table in turn.

	Data codewords					Error correction codewords			
Block 1	D1	D2	.....	D11		E1	E2	.....	E22
Block 2	D12	D13	.....	D22		E23	E24	.....	E44
Block 3	D23	D24	.....	D33	D34	E45	E46	.....	E66
Block 4	D35	D36	.....	D45	D46	E67	E68	.....	E88

The final message codeword sequence for the Version 5-H symbol is therefore:

D1, D12, D23, D35, D2, D13, D24, D36, ... D11, D22, D33, D45, D34, D46, E1, E23, E45, E67, E2, E24, E46, E68, ... E22, E44, E66, E88. The symbol module capacity is filled by adding Remainder (0) bits as needed after the final codeword.

8.7 Codeword placement in matrix

8.7.1 Symbol character representation

There are two types of symbol character, regular and irregular, in the QR Code symbol. Their use depends on their position in the symbol, relative to other symbol characters and function patterns.

Most codewords shall be represented in a regular 2 x 4 module block in the symbol. There are two ways of positioning these blocks, in a vertical arrangement (2 modules wide and 4 modules high) and, if necessary when placement changes direction, in a horizontal arrangement (4 modules wide and 2 modules high). Irregular symbol characters are used when changing direction or in the vicinity of Alignment or other function Patterns.

8.7.2 Function pattern placement

A square blank matrix shall be constructed with the number of modules horizontally and vertically corresponding to the Version in use. Positions corresponding to the Finder Pattern, Separator, Timing Pattern, and Alignment Patterns shall be filled with either dark modules or light modules as appropriate. Module positions for the Format Information and Version Information shall be left temporarily blank. These positions are shown in Figures 15 and 16 and are common to all Versions. Annex E defines the positioning of Alignment Patterns.

8.7.3 Symbol character placement

In the encoding region of the QR Code symbol, symbol characters are positioned in two-module wide columns commencing at the lower right corner of the symbol and running alternately upwards and downwards from the right

to the left. The principles governing the placement of characters and of bits within the characters are given below. Figures 15 and 16 illustrate Version 2 and Version 7 symbols applying these principles.

a) The sequence of bit placement in the column shall be from right to left and either upwards or downwards in accordance with the direction of symbol character placement.

b) The most significant bit (shown as bit 7) of each codeword shall be placed in the first available module position. Subsequent bits shall be placed in the next module positions. The most significant bit therefore occupies the lower right module of a regular symbol character when the direction of placement is upwards, and the upper right module when the direction of placement is downwards. It may however occupy the lower left module of an irregular symbol character if the previous character has ended in the right-hand module column (see Figure 14 under e) below).

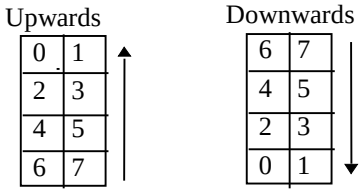


Figure 12 — Bit placement in regular symbol character in upwards and downwards directions

c) When a symbol character encounters the horizontal boundary of an Alignment Pattern or of the Timing Pattern in both module columns, it shall continue above or below the pattern as though the encoding region were continuous.

d) When the upper or lower boundary of the symbol character region is reached (i.e. the edge of the symbol, Format Information, Version Information, or Separator) any remaining bits in the codeword shall be placed in the next column to the left. The direction of placement reverses.

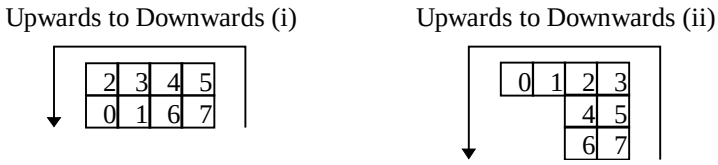


Figure 13 — Example of bit placement in symbol characters when direction of placement changes

e) When the right-hand module column of the symbol character column encounters an Alignment Pattern or an area occupied by Version Information, bits are placed to form an irregular symbol character, extending along the single module column adjacent to the Alignment Pattern or Version Information. If the character ends before two columns are available for the next symbol character, the most significant bit of the next character shall be placed in the single column.

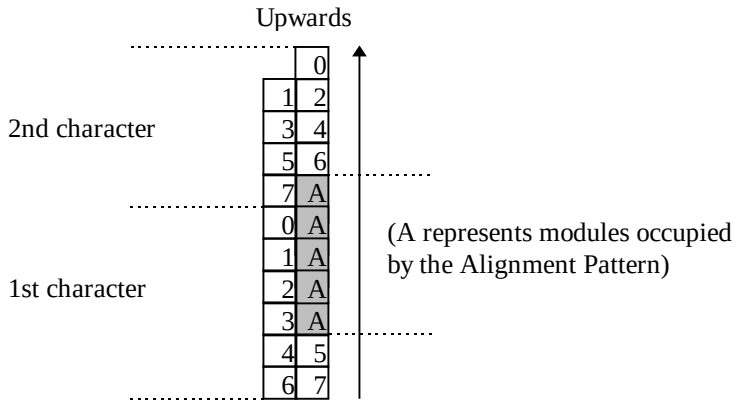


Figure 14 — Example of bit placement adjacent to Alignment Pattern

An alternative method for placement in the symbol, which yields the same result, is to regard the interleaved codeword sequence as a single bit stream, which is placed (starting with the most significant bit) in the two-module wide columns alternately upwards and downwards from the right to left of the symbol. In each column the bits are placed alternately in the right and left modules, moving upwards or downwards according to the direction of placement and skipping areas occupied by function patterns, changing direction at the top or bottom of the column. Each bit shall always be placed in the first available module position.

When the data capacity of the symbol is such that it does not divide exactly into a number of eight-bit symbol characters, the appropriate number of Remainder Bits (3, 4 or 7 as shown in Table 1) shall be used to fill the symbol capacity. These Remainder Bits shall always have the value 0 before masking according to 8.8.

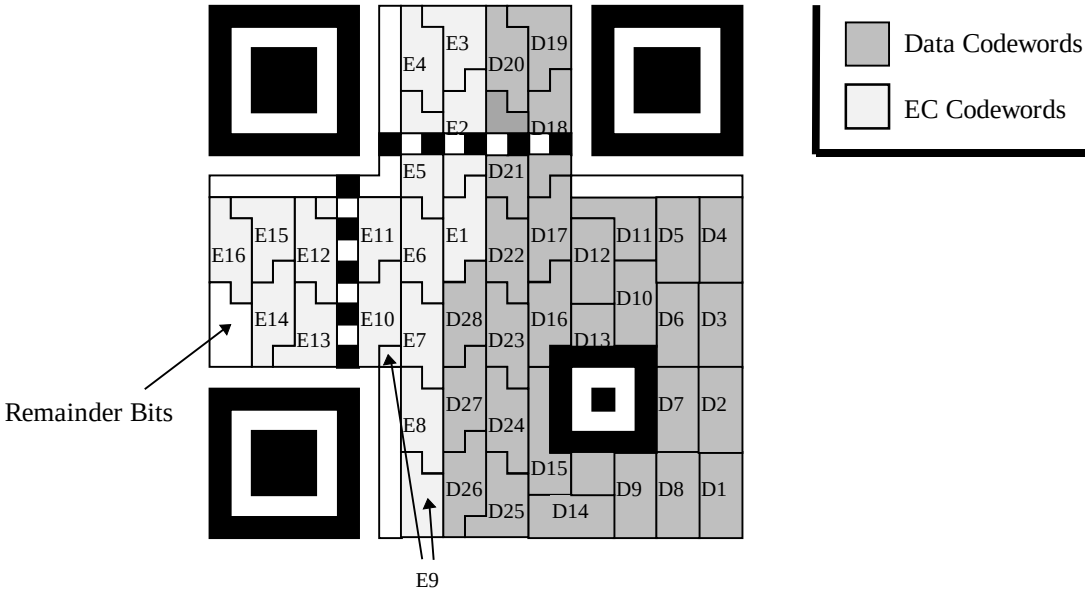
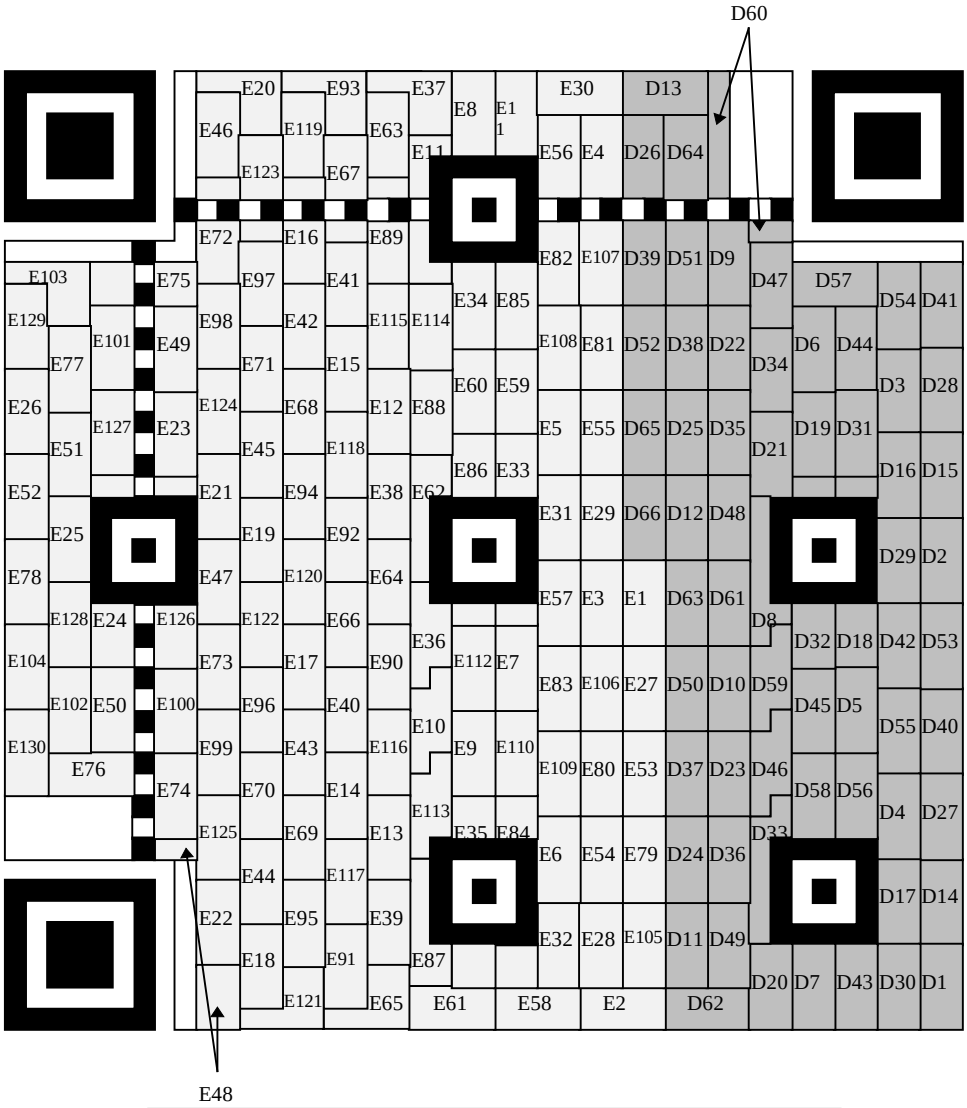


Figure 15 — Symbol character arrangement in version 2-M symbol



D1 - D13	Data Block 1
D14 - D26	Data Block 2
D27 - D39	Data Block 3
D40 - D52	Data Block 4
D53 - D66	Data Block 5
E1 - E26	EC Block 1
E27 - E52	EC Block 2
E53 - E78	EC Block 3
E79 - E104	EC Block 4
E105 - E130	EC Block 5

Figure 16 — Symbol character arrangement in version 7-H symbol

8.8 Masking

For reliable QR Code reading, it is preferable for dark and light modules to be arranged in a well-balanced manner in the symbol. The bit pattern **1011101** particularly found in the Position Detection Pattern should be avoided in other areas of the symbol as much as possible. To meet the above conditions, masking should be applied following the steps described below:

1. Masking is not applied to function patterns.

2. Convert the given module pattern in the encoding region (excluding the Format Information and the Version Information) with multiple matrix patterns successively through the XOR operation. For the XOR operation, lay the module pattern over each of the masking matrix patterns in turn and reverse the modules (from light to dark or vice versa) which correspond to dark modules of the masking pattern.

3. Then evaluate all the resulting converted patterns by charging penalties for undesirable features on each conversion result.

4. Select the pattern with the lowest penalty points score.

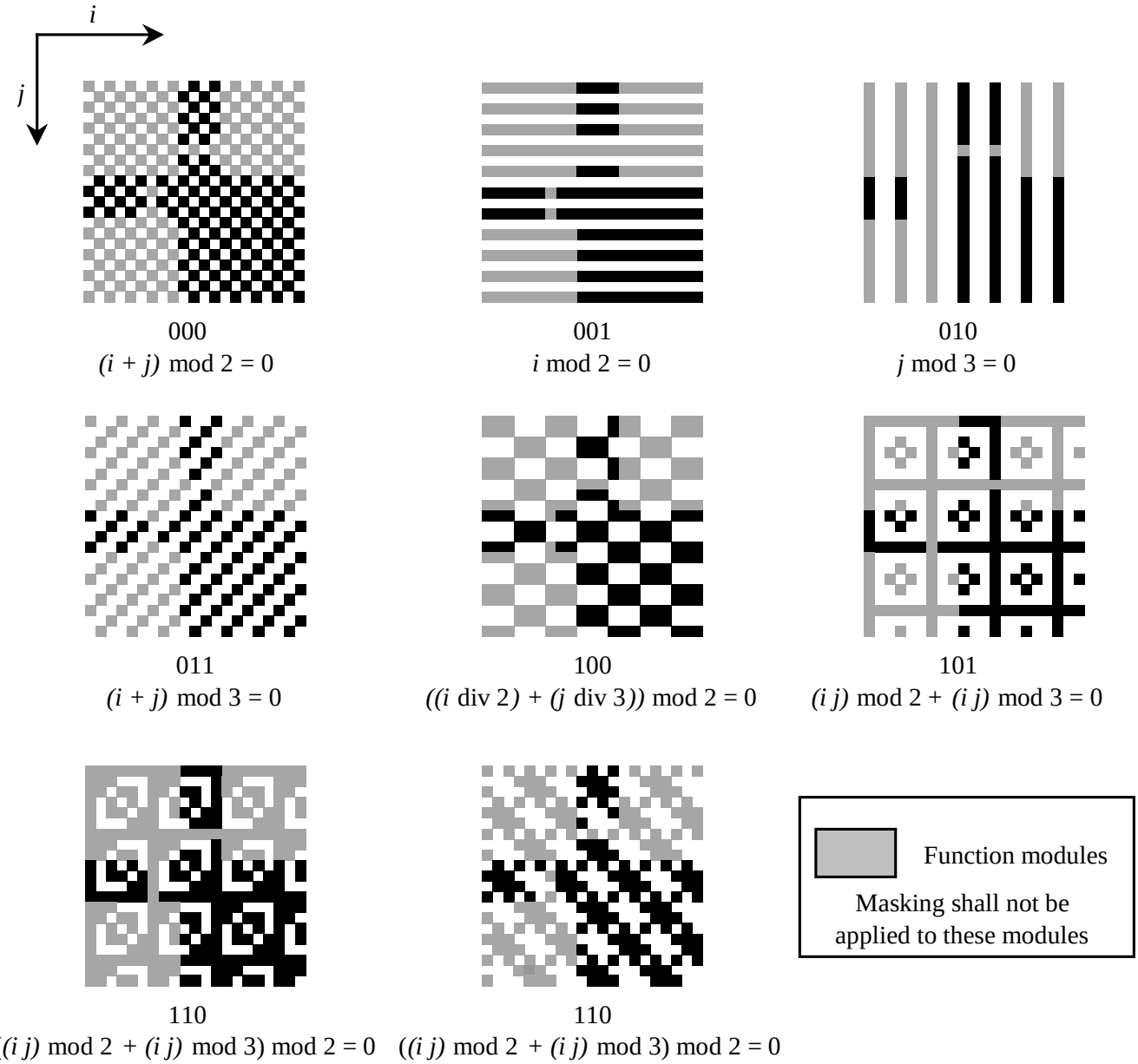
8.8.1 Mask Patterns

Table 23 shows the Mask Pattern Reference (binary reference for use in the Format Information) and the mask pattern generation condition. The mask pattern is generated by defining as dark any module in the encoding region (excluding the area reserved for Format Information and the Version Information) for which the condition is true; in the condition, *i* refers to the row position of the module in question and *j* to its column position, with (*i, j*) = (0, 0) for the top left module in the symbol.

Table 23 — Mask pattern generation conditions

Mask Pattern Reference	Condition
000	$(i + j) \bmod 2 = 0$
001	$i \bmod 2 = 0$
010	$j \bmod 3 = 0$
011	$(i + j) \bmod 3 = 0$
100	$((i \operatorname{div} 2) + (j \operatorname{div} 3)) \bmod 2 = 0$
101	$(i\,j) \bmod 2 + (i\,j) \bmod 3 = 0$
110	$((i\,j) \bmod 2 + (i\,j) \bmod 3) \bmod 2 = 0$
111	$((i\,j) \bmod 3 + (i+j) \bmod 2) \bmod 2 = 0$

Figure 17 shows all Mask Patterns, illustrated in a Version 1 symbol; Figure 18 simulates the effects of masking using Mask Pattern References 000 to 111.



NOTE 1 The three bits below each pattern represent the Mask Pattern Reference.

NOTE 2 The equation below the Mask Pattern Reference shows the mask pattern generation condition; modules which meet the condition are shown dark.

Figure 17 — Mask patterns for version 1 symbol

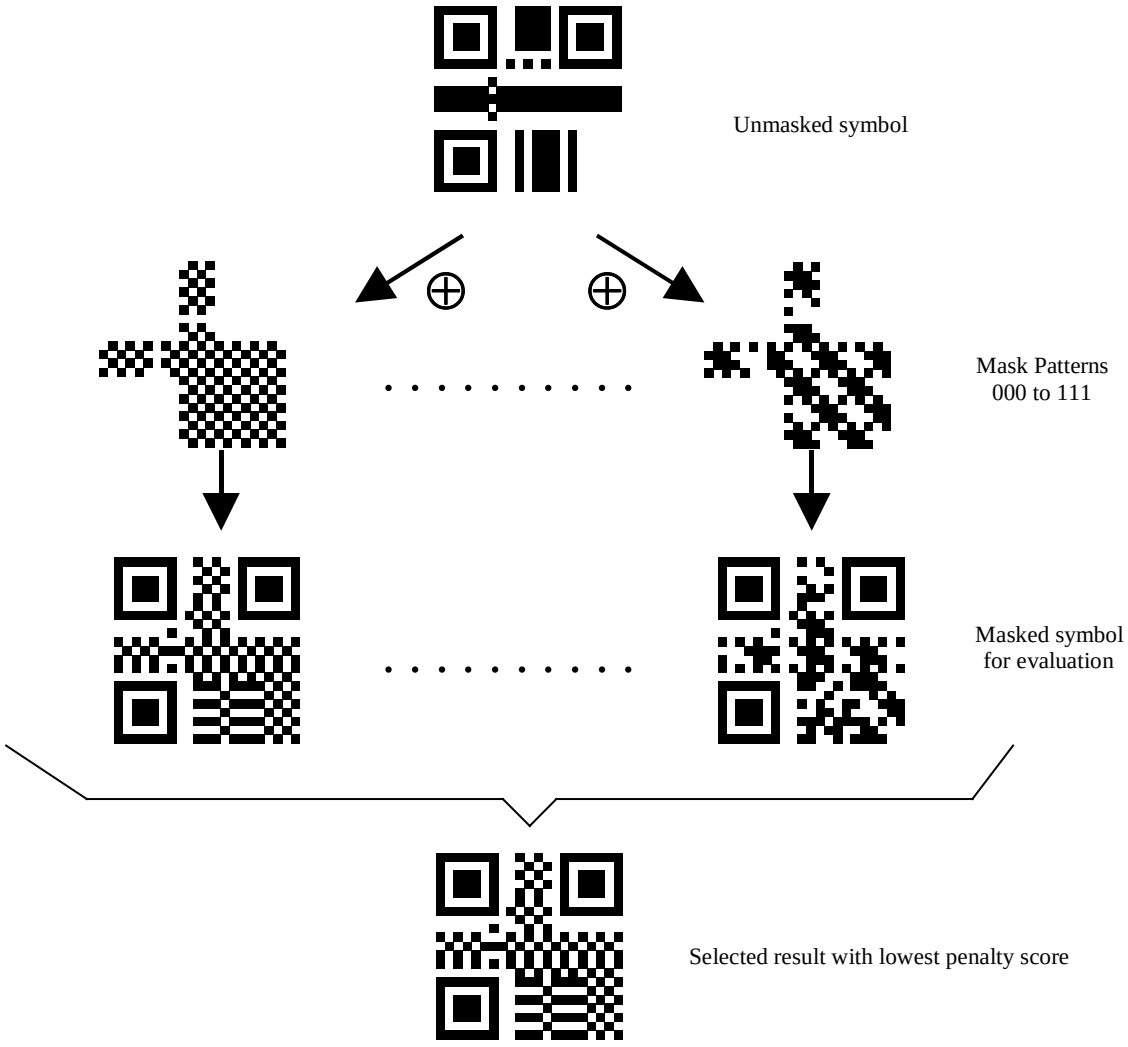


Figure 18 — Masking simulation in Model 2

8.8.2 Evaluation of masking results

After performing the masking operation with each Mask Pattern in turn, the results shall be evaluated by scoring penalty points for each occurrence of the following features. The higher the number of points, the less acceptable the result. In Table 24 below, the variables N_1 to N_4 represent weighted penalty scores for the undesirable features (N_1=3, N_2=3, N_3=40, N_4=10), i is the amount by which the number of adjacent modules of the same color exceeds 5 and k is the rating of the deviation of the proportion of dark modules in the symbol from 50% in steps of 5%. Although the masking operation is only performed on the encoding region of the symbol excluding the Format Information, the area to be evaluated is the complete symbol.

Table 24 — Scoring of masking results

Feature	Evaluation condition	Points
Adjacent modules in row/column in same color	No. of modules = (5 + i)	N_1 + i
Block of modules in same color	Block size = m × n	$N_2 \times (m - 1) \times (n - 1)$
1:1:3:1:1 ratio (dark:light:dark:light:dark) pattern in row/column		N_3
Proportion of dark modules in entire symbol	50 ± (5 × k)% to 50 ± (5 × (k + 1))%	$N_4 \times k$

The Mask Pattern which results in the lowest score shall be selected for the symbol.

8.9 Format Information

The Format Information is a 15 bit sequence containing 5 data bits, with 10 error correction bits calculated using the (15, 5) BCH code. For details of the error correction calculation for the Format Information, refer to Annex C. The first two data bits contain the Error Correction Level of the symbol, indicated as follows:

Table 25 — Error correction level indicators

Error Correction Level	Binary indicator
L	01
M	00
Q	11
H	10

The third to fifth data bits of the Format Information contain the Mask Pattern Reference from Table 23 above for the pattern selected according to 8.8.2.

The 10 error correction bits shall be calculated as described in Annex C and appended to the 5 data bits.

The 15 bit error corrected Format Information shall then be XORed with the Mask Pattern **101010000010010**, in order to ensure that no combination of Error Correction Level and Mask Pattern will result in an all-zero data string.

The resulting masked Format Information shall be mapped into the areas reserved for it in the symbol as shown in Figure 19. Note that the Format Information appears twice in the symbol in order to provide redundancy since its correct decoding is essential to the decoding of the complete symbol. The least significant bit of the Format Information is located in the modules numbered 0, and the most significant bit in the modules numbered 14 in Figure 19. The module in position (4V+ 9, 8) where V is the version number, shall always be dark and does not form part of the Format Information.

Example:

Assume Error Correction Level M:	**00**
and Mask Pattern Reference:	**101**
Data:	**00101**
BCH bits:	**0011011100**
Unmasked bit sequence:	**001010011011100**
Mask pattern for XOR operation:	**101010000010010**
Format Information module pattern:	**100000011001110**

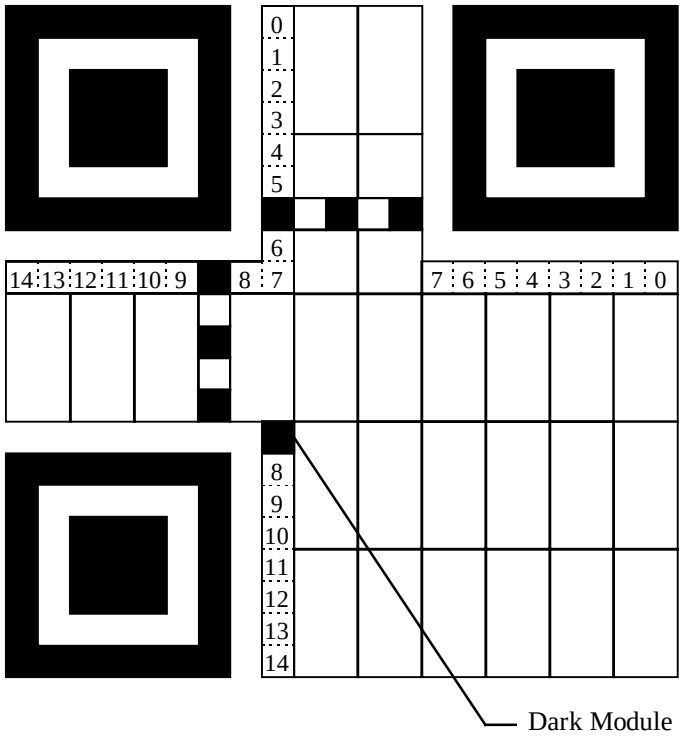


Figure 19 — Format Information positioning

8.10 Version Information

The Version Information is an 18 bit sequence containing 6 data bits, with 12 error correction bits calculated using the (18, 6) BCH code. For details of the error correction calculation for the Version Information, refer to Annex D. The six data bits contain the Version of the symbol, most significant bit first.

The 12 error correction bits shall be calculated as described in Annex D and appended to the 6 data bits.

No Version Information will result in an all-zero data string since only Versions 7 to 40 symbols contain the Version Information. Masking is not therefore applied to the Version Information.

The resulting Version Information shall be mapped into the areas reserved for it in the symbol as shown in Figure 20. Note that the Version Information appears twice in the symbol in order to provide redundancy since its correct decoding is essential to the decoding of the complete symbol. The least significant bit of the Version Information is located in the modules numbered 0, and the most significant bit in the modules numbered 17, in Figure 21.

Example:	
Version number:	**7**
Data:	**000111**
BCH bits:	**110010010100**
Format Information module pattern:	**000111110010010100**

The Version Information areas are the 6 × 3 module block above the Timing Pattern and immediately to the left of the top right Position Detection Pattern Separator, and the 3 × 6 module block to the left of the Timing Pattern and immediately above the lower left Position Detection Pattern separator.

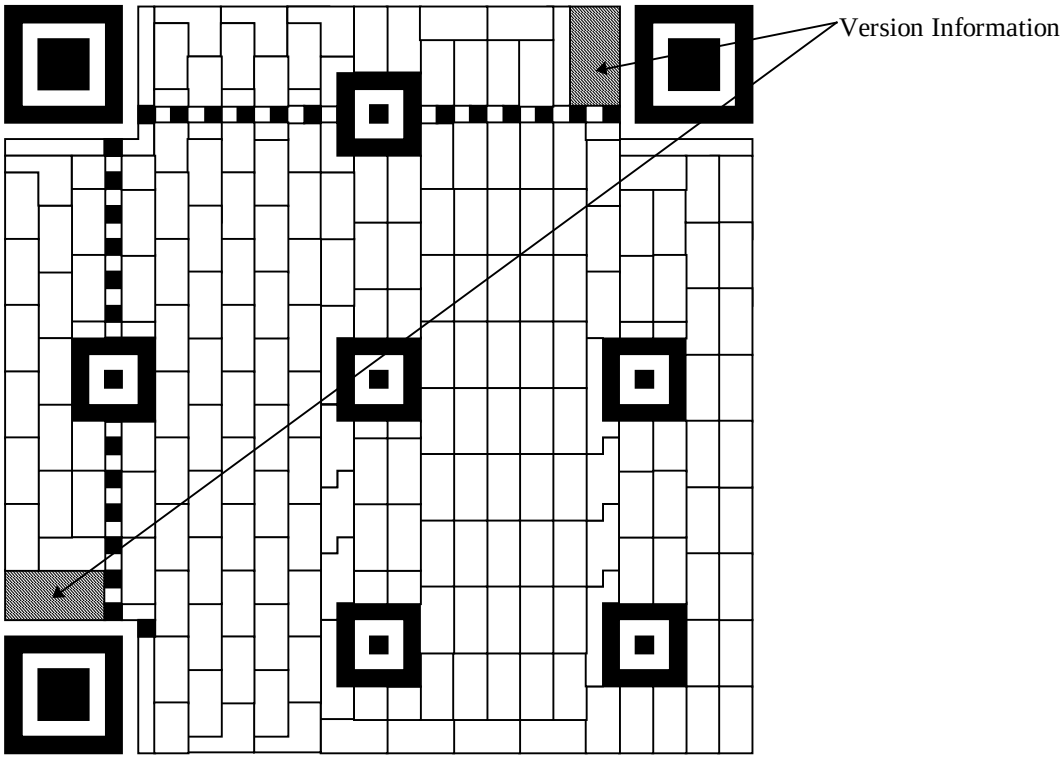


Figure 20 — Version Information positioning

0	3	6	9	12	15
1	4	7	10	13	16
2	5	8	11	14	17

Version Information in lower left

0	1	2
3	4	5
6	7	8
9	10	11
12	13	14
15	16	17

Version Information in upper right

Figure 21 — Module arrangement in Version Information

9 Structured Append

9.1 Basic principles

Up to 16 QR Code symbols may be appended in a structured format. If a symbol is part of a Structured Append message, it is indicated by a header block in the first three symbol character positions.

The Structured Append Mode Indicator **0011** is placed in the four most significant bit positions in the first symbol character.

This is immediately followed by two Structured Append codewords, spread over the four least significant bits of the first symbol character, the second symbol character and the four most significant bits of the third symbol character. The first codeword is the symbol sequence indicator. The second codeword is the parity data and is identical in all symbols in the message, enabling it to be verified that all symbols read form part of the same Structured Append message. This header is immediately followed by the data codewords for the symbol commencing with the first Mode Indicator. If one or more ECIs other than the default ECI is in force, an ECI header for each ECI, consisting of the ECI Mode Indicator and ECI Designator, shall follow the Structured Append header.

The lower part of Figure 22 shows an example of four Structured Append symbols, with the same data as that in the upper symbol.

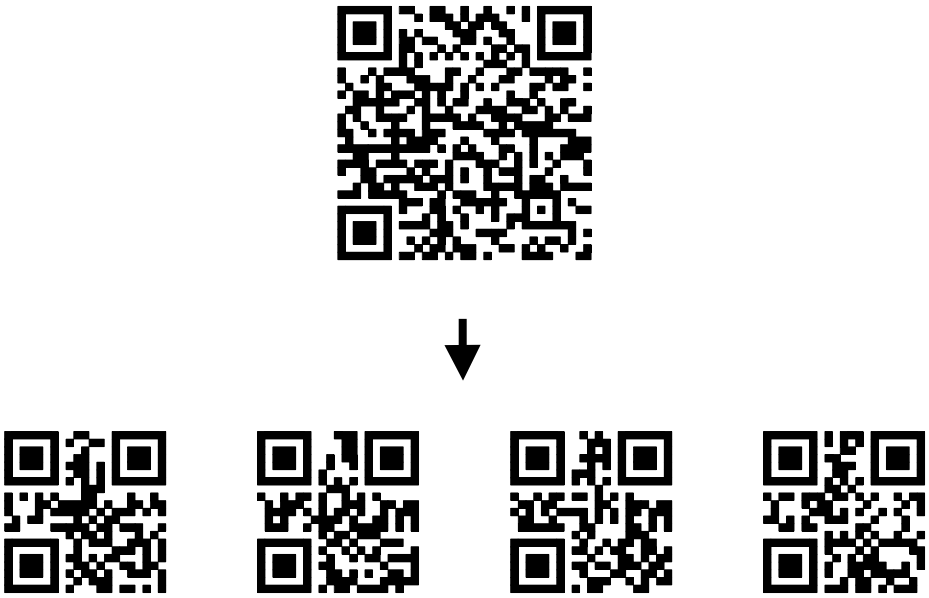


Figure 22 — Structured append

9.2 Symbol Sequence Indicator

This codeword indicates the position of the symbol within the set of (up to 16) QR Code symbols in the Structured Append format (in the form *m* of *n* symbols). The first 4 bits of this codeword identify the position of the particular symbol. The last 4 bits identify the total number of symbols to be concatenated in the Structured Append format. The 4-bit patterns shall be the binary equivalents of (*m* - 1) and (*n* - 1) respectively.

Example:

To indicate the 3rd symbol of a set of 7, this shall be encoded thus:

3rd position: **0010**

Total 7 symbols: **0110**

Bit pattern: **00100110**

9.3 Parity Data

The Parity Data shall be an 8 bit byte following the Symbol Sequence Indicator. The parity data is a value obtained by XORing byte by byte the ASCII/JIS values of all the original input data before division into symbol blocks. Mode Indicators, Character Count Identifiers, padding bits, Terminator and Pad Characters shall be excluded from the calculation. Input data is represented for this calculation by 2-byte Shift JIS values for Kanji (each byte being treated separately in the XOR calculation) and 8-bit JIS values as shown in Table 6 for other characters. In ECI Mode the 8-bit byte values obtained after any encryption or compression of the data shall be used for the calculation.

For example, "0123456789日本" is divided into "0123", "4567" and "89日本" as follows:

1st symbol block ("0123") - hex. values 30, 31, 32, 33

2nd symbol block("4567") - hex. values 34, 35, 36, 37

3rd symbol block ("89日本") - hex. values 38, 39, 93FA, 967B

The parity data is calculated from "0123456789日本" by XORing the data successively, byte by byte.

30 ⊕ 31 ⊕ 32 ⊕ 33 ⊕ 34 ⊕ 35 ⊕ 36 ⊕ 37 ⊕ 38 ⊕ 39 ⊕ 93 ⊕ FA ⊕ 96 ⊕ 7B = 85

Note that the calculation of the parity data may be performed either before the data is sent to the printer or in the printer, based on the capabilities of the printer.

10 Symbol printing and marking

10.1 Dimensions

QR Code symbols shall conform to the following dimensions:

X dimension: the width of a module shall be specified by the application, taking into account the scanning technology to be used, and the technology to produce the symbol;

Y dimension: the height of a module shall be equal to the X dimension;

minimum quiet zone: equal to 4X on all four sides.

10.2 Human-readable interpretation

Because QR Code symbols are capable of encoding thousands of characters, a human readable interpretation of the data characters may not be practical. As an alternative, descriptive text rather than literal text may accompany the symbol.

The character size and font are not specified, and the message may be printed anywhere in the area surrounding the symbol. The human readable interpretation should not interfere with the symbol itself nor the quiet zones.

10.3 Marking guidelines

QR Code symbols can be printed or marked using a number of different techniques. Annex I provides user guidelines.

11 Symbol quality

QR Code symbols shall be assessed for quality using the 2D matrix bar code symbol print quality guidelines presented in Annex K, as augmented and modified below.

11.1 Obtaining the test image

A grey-scale image of the symbol being tested shall be obtained with a precision video camera-based setup as described in Annex K.1, but with an illumination color and direction specified by the application.

11.2 Symbol quality parameters

11.2.1 Decode

The reference decode algorithm set out in 13 shall be applied to the test image. If it results in a successful decode of the entire data message, then Decode passes with a grade of 4 ("A"), otherwise it fails with a grade of 0 ("F").

11.2.2 Symbol Contrast

Symbol Contrast shall be graded using all of the grey-scale pixel values in the test image which fall within the symbol's boundary, including the 4X wide quiet zones, as determined by the reference decode. The procedure is defined in Annex K.2.2.

11.2.3 "Print" growth

The reference decode begins by creating a high resolution binary digitized image of the test symbol, and at a later point establishes the position of the "alternating module centerlines" which bisect the Timing Patterns of the symbol. "Print" Growth shall be assessed by checking that the "duty cycle" of the lines through the alternating patterns is around 50%.

Taking the two Timing Patterns independently (since horizontal and vertical growth can differ substantially), proceed from the inner side of the outer square of the Position Detection Pattern adjacent to each Timing Pattern along each Timing Pattern to the outer edge of the Separator adjacent to the other Position Detection Pattern and sum the number of light (N_L) and dark (N_D) pixels encountered. The resulting measure of print growth in each direction is $D = N_D / (N_L + N_D)$, which shall be graded against $D_{NOM} = 0.50$ with $D_{MIN} = 0.35$ and $D_{MAX} = 0.65$ as prescribed in Annex K.2.3. The Print Growth grade shall be the lower of those achieved along the vertical and horizontal Timing Patterns.

11.2.4 Axial Nonuniformity

The reference decode algorithm ultimately creates a grid of data module sampling points throughout the entire area of the test image. The precise horizontal and vertical spacings of those sampling points are the basis for assessing axial nonuniformity.

Working independently with the horizontal and vertical spacings between adjacent data modules, compute their average values X_{AVG} and Y_{AVG} over the whole symbol. The Axial Nonuniformity grade is then based on how closely these two average spacings match each other, as computed by the procedure defined in Annex K.2.4.

11.2.5 Unused Error Correction

QR Code employs Reed-Solomon error control encoding. The smaller symbols contain a single error correction field while the larger symbols are subdivided into two or more error correction fields. In all cases, each error correction field shall be graded independently as specified in Annex K.2.5, then the Unused Error Correction grade shall be the lowest achieved by any of the fields. This calculation shall not apply, however, to the Format Information nor to the Version Information.

11.3 Overall symbol grade

The overall print quality grade for a QR Code symbol is the lowest of the five grades achieved above. Table 26 summarizes the test criteria.

Table 26 — Symbol grading criteria

Grade	Reference decode	Symbol Contrast	“Print” growth	Axial Nonuniformity	Unused Error Correction
4,0 (A)	Passes	SC ≥ 0,70	-0,50 ≤ D’ ≤ 0,50	AN ≤ 0,06	UEC ≥ 0,62
3,0 (B)		SC ≥ 0,55	-0,70 ≤ D’ ≤ 0,70	AN ≤ 0,08	UEC ≥ 0,50
2,0 (C)		SC ≥ 0,40	-0,85 ≤ D’ ≤ 0,85	AN ≤ 0,10	UEC ≥ 0,37
1,0 (D)		SC ≥ 0,20	-1,00 ≤ D’ ≤ 1,00	AN ≤ 0,12	UEC ≥ 0,25
0,0 (F)	Fails	SC < 0,20	D’ < -1,00 or D’ > 1,00	AN > 0,12	UEC < 0,25

11.4 Process control measurements

A variety of tools and methods is available which can perform useful measurements for monitoring and controlling the process of creating QR Code symbols. These include:

1. Symbol contrast readings using a linear bar code verifier.

2. Horizontal (and vertical) print growth by measurement of the Position Detection Patterns in both axes using a linear bar code verifier.

3. Determination of axial nonuniformity by physical measurement.

4. Visual inspection of the position detection and timing patterns for grid nonuniformities and defects.

Each of these tools and methods is described in Annex L.

12 Decoding procedure overview

The decoding steps from reading a QR Code symbol to outputting data characters are the reverse of the encoding procedure. Figure 23 shows an outline of the process flow.

1. Locate and obtain an image of the Symbol. Recognize Dark and Light Modules as an array of “0” and “1” bits.

2. Read the Format Information. (Release the masking pattern and perform error correction on the Format Information modules as necessary; identify Error Correction Level and Mask Pattern Reference).

3. Read the Version Information (where applicable), then determine the Version of the symbol.

4. Release the Masking by XORing the encoding region bit pattern with the Mask Pattern the reference of which has been extracted from the Format Information.

5. Read the symbol characters according to the placement rules for the model and restore the data and error correction codewords of the message.

6. Detect errors using the error correction codewords corresponding to the Level Information. If any error is detected, correct it.

7. Divide the data codewords into segments according to the Mode Indicators and Character Count Indicators.

8. Finally, decode the Data Characters in accordance with the Mode(s) in use and output the result.

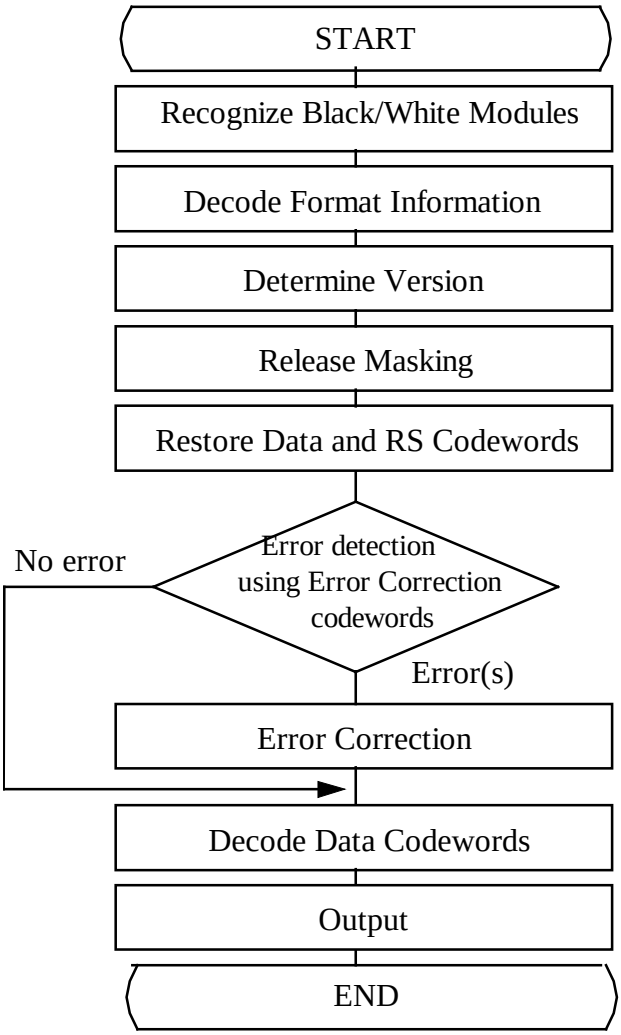


Figure 23 — QR Code decoding steps

13 Reference decode algorithm for QR Code

This reference decode algorithm finds the symbol in an image and decodes them. The decode algorithm refers to dark and light states in the image.

1. Determine a Global Threshold by taking a reflectance value midway between the maximum reflectance and minimum reflectance in the image. Convert the image to a set of dark and light pixels using the Global Threshold.

2. Locate the Finder Pattern. The finder pattern in QR Code consists of three identical Position Detection Patterns located at three of the four corners of the symbol. As described in 7.3.2, module width in each Position Detection Pattern is constructed of a dark-light-dark-light-dark sequence the relative widths of each element of which are in the ratios 1:1:3:1:1. For the purposes of this algorithm the tolerance for each of these widths is 0,5 (i.e. a range of 0,5 to 1,5 for the single module box and 2,5 to 3,5 for the three module square box).

   a) When a candidate area is detected note the position of the first and last points A and B respectively at which a line of pixels in the image encounters the outer edges of the position detection pattern (see Figure 29). Repeat this for adjacent pixel lines in the image until all lines crossing the central box of the position detection pattern in the x axis of the image have been identified.

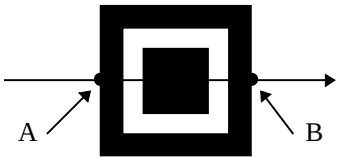


Figure 24 — Scan line in Position Detection Pattern

b) Repeat step a) for pixel columns crossing the central box of the position detection pattern in the y axis of the image.

c) Locate the center of the pattern. Construct a line through the midpoints between the points A and B on the outermost pixel lines crossing the central box of the position detection pattern in the x axis. Construct a similar line through points A and B on the outermost pixel columns crossing the central box in the y axis. The center of the pattern is located at the intersection of these two lines.

d) Repeat steps a) to c) to locate the centers of the two other position detection patterns.

3. Determine the orientation of the symbol by analysing the position detection pattern center coordinates to identify which pattern is the upper left pattern in the symbol and the angle of rotation of the symbol.

4. Determine a) the distance D crossing the full width of the symbol between the centers of the upper left position detector pattern and of the upper right position detection pattern and b) the width of the two patterns, W_{UL} and W_{UR}.

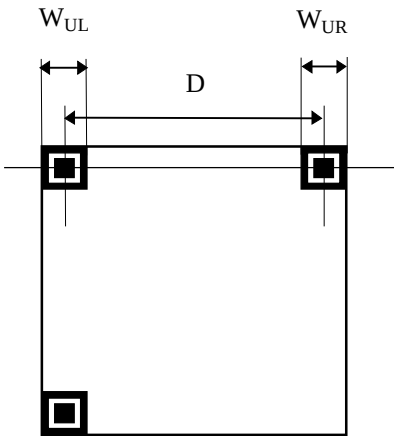


Figure 25 — Upper Position Detection Patterns

5. Calculate the nominal X dimension of the symbol.

   $X = (W_{UL} + W_{UR}) / 14$

6. Provisionally determine the version V of the symbol.

   $V = [(D/X) - 10] / 4$

7. If the provisional symbol version is 6 or less, this is specified as the defined version. If the provisional symbol version is 7 or more, the version information is decoded as follows.

   a) Divide the width W_{UR} of the upper right Position Detection Pattern by 7 to calculate the module size CP_{UR}.

      $CP_{UR} = W_{UR} / 7$

b) Find the guide lines AC and AB from A, B and C, which pass through the centers of the three Position Detection Patterns, as shown in Figure 26 below. The sampling grid for each module center in the Version Information 1 area is determined based on lines parallel to the guide lines, the central coordinates of the Position Detection Patterns, and the module size CP_{UR}. Binary values 0 and 1 are determined from the light or dark pattern on the sampling grid.

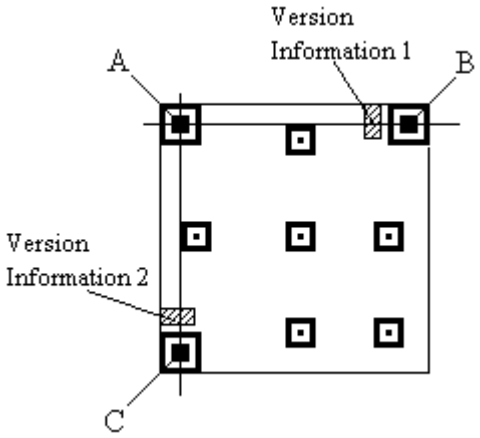


Figure 26 — Position Detection Patterns and Version Information

c) Determine the version by detecting and correcting errors, if any, based on the BCH error correction applied to the Version Information modules, defined in Annex D.

d) If errors exceeding the error correction capacity are detected, then calculate the pattern width W_{DL} of the lower left Position Detection Pattern and follow a similar procedure to steps a), b) and c) above to decode Version Information 2.

8. Decoding of a Version 1 symbol with no Alignment Pattern continues in accordance with steps 7 and 8 of the algorithm in Annex M.14, reverting to this algorithm at step 9. Decoding of version 2 and larger symbols requires the central coordinate of each Alignment Pattern to be determined from the coordinates defined in 7.3.5 and Annex E to determine the sampling grids.

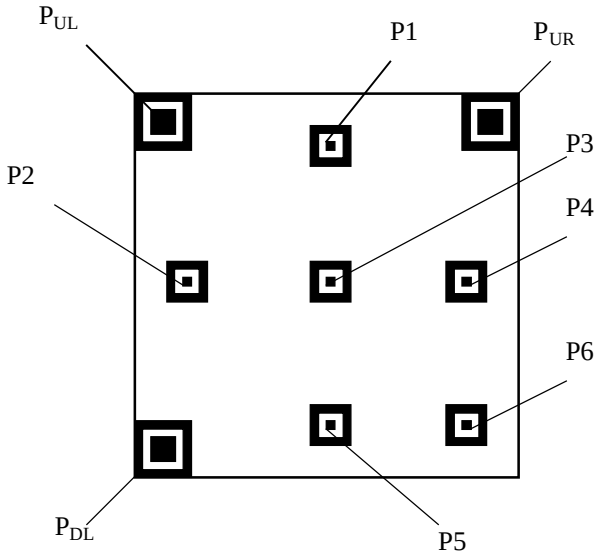


Figure 27 — Position Detection Patterns and Alignment Patterns

a) Divide the pattern width W_{UL} of the upper left Position Detection Pattern P_{UL} by 7 to calculate the module size CP_{UL}.

$CP_{UL} = W_{UL} / 7$

b) Determine the provisional central coordinates of the Alignment Patterns P1 and P2, based on the coordinate of the center A of the upper left Position Detection Pattern P_{UL}, lines parallel to the guide lines AB and AC obtained in 7c), and the module size CP_{UL}.

c) Scan the outline of the white square in Alignment pattern P1 and P2 starting from the pixel of the provisional central coordinate to find the actual central coordinates Xi and Yj (see Figure 28).

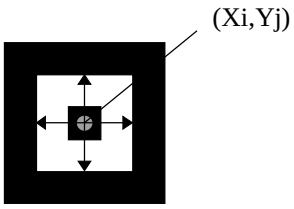


Figure 28 — Central coordinates of Alignment Pattern

d) Estimate the provisional central coordinate of the Alignment Pattern P3, based on the central coordinate of the upper left Position Detection Pattern P_{UL} and the actual central coordinates of the Alignment Patterns P1 and P2 obtained in c).

e) Find the actual central coordinate of the Alignment Pattern P3 by following the same procedure in c).

f) Find Lx, which is the center-to-center distance of the Alignment patterns P2 and P3, and Ly, which is the center-to-center distance of the Alignment Patterns P1 and P3. Divide Lx and Ly by the defined spacing of the Alignment Patterns to obtain the module pitches CPx in the lower side and CPy in the right side in the upper left area of the symbol (see Figure 29).

CPx = Lx / AP

CPy = Ly / AP

where AP is the spacing in modules of the Alignment Pattern centers (see Table E.1).

In the same fashion, find Lx', which is the horizontal distance between the central coordinates of the upper left Position Detection Pattern P_{UL} and the central coordinates of the Alignment Pattern P1, and Ly', which is the vertical distance between the central coordinates of the upper left Position Detection Pattern P_{UL} and the central coordinates of the Alignment Pattern P2. Divide Lx' and Ly' by the formula below to obtain the module pitches CPx' in the upper side and CPy' in the left side in the upper left area of the symbol.

CPx' = Lx' / (Column coordinate of the central module of the Alignment Pattern P1

- Column coordinate of the central module of the upper left Position Detection Pattern P_{UL})

CPy' = Ly' / (Row coordinate of the central module of the Alignment Pattern P2

- Row coordinate of the central module of the upper left Position Detection Pattern P_{UL})

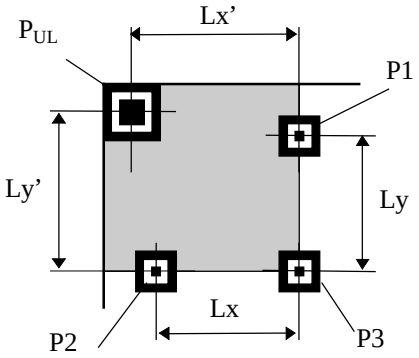


Figure 29 — Upper left area of symbol

g) Determine the sampling grid covering the upper left area of the symbol based on the module pitches CPx, CPx', CPy and CPy' representing each side in the upper left area of the symbol.

h) In the same fashion detemine the sampling grids for the upper right area (covered by the upper right Position Detection Pattern P_{UR}, Alignment Patterns P1, P3 and P4) and lower left area (covered by the upper right Position Detection Pattern P_{UR}, Alignment Patterns P2, P3 and P5) of the symbol.

i) For the Alignment Pattern P6 (see Figure 30), estimate its provisional central coordinate from the module pitches CPx' and CPy', the values of which are obtained from the spacings of Alignment Patterns P3, P4 and P5, guide lines passing through the centers of the Alignment Patterns P3 and P4, and P3 and P5 respectively, and the central coordinates of these Patterns.

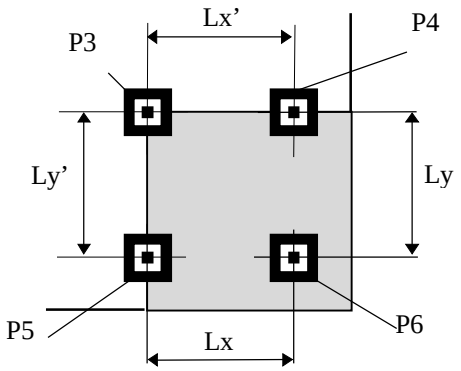


Figure 30 — Lower right area of symbol

j) Repeat steps e) to h) to determine the sampling grid for the lower right area of the symbol.

k) The same principles shall be applied to determine the sampling grids for any areas of the symbol not already covered.

9. Sample the image pixel on each intersection of the grid lines and determine whether it is dark or light based on the Global Threshold. Construct a bit matrix mapping the dark pixels as binary 1 and light pixels as binary 0.

10. Decode the Format Information adjacent to the upper left Position Detection Pattern to yield the Error Correction Level and the Mask Pattern applied to the symbol. If errors exceeding the error correction capacity of the Format Information are detected, then follow the same procedure to decode the Format Information adjacent to the upper right and lower left Position Detection Patterns.

11. XOR the Mask Pattern with the encoding region of the symbol to release the masking and restore the symbol characters representing data and error correction codewords. This reverses the effect of the masking process applied during the encoding procedure.

12. Determine the symbol codewords in accordance with the placement rules in 8.7.3.

13. Rearrange the codeword sequence into blocks as required for the symbol Version and Error Correction Level, by reversing the interleaving process defined in 8.6. step 3).

14. Follow the error detection and correction decoding procedure in Annex B to correct errors and erasures up to the maximum correction capacity for the symbol version and Error Correction Level.

15. Restore the original message bit stream by assembling the data blocks in sequence.

16. Subdivide the data bit stream into segments each commencing with a Mode Indicator and the length of which is determined by the Character Count Indicator following the Mode Indicator.

17. Decode each segment according to the rules for the Mode in force.

14 Autodiscrimination capability

QR Code can be used in an autodiscrimination environment with a number of other symbologies. (See Annex J). In addition, Model 1 and Model 2 symbols can be autodiscriminated by analysis of the Format Information mask pattern. Equipment suppliers may offer the option of decoding of only Model 1 or only Model 2 or of both Models.

15 Transmitted data

All encoded data characters shall be included in the data transmission. The function patterns, format and version information, error correction characters, Pad and Remainder characters shall not transmitted. The default transmission mode for all data shall be as their 8-bit JIS8 values or 16-bit Shift JIS values. Because of the character value assignments this gives unambiguous transmission of any sequence of numeric, Latin, Kana and Kanji data. The Structured Append header block shall not be transmitted by decoders operating in buffered mode which reconstructed the complete message before transmission. If the decoder is operating in unbuffered mode the Structured Append header shall be transmitted as the first 2 bytes of every symbol. More complex interpretations including the transmission of data in an Extended Channel Interpretation, are addressed below.

15.1 Symbology Identifier

ISO/IEC 15424 provides a standard procedure for reporting the symbology which has been read, together with options set in the decoder and any special features encountered in the symbol.

Once the structure of the data (including the use of any ECI) has been identified, the appropriate Symbology Identifier should be added by the decoder as a preamble to the transmitted data; if ECIs are used the Symbology Identifier is required. See Annex F for the Symbology Identifier and option values which apply to QR Code.

15.2 Extended Channel Interpretations

In systems where the ECI protocol is supported the transmission of the Symbology Identifier is required with every transmission. Whenever the ECI Mode Indicator is encountered, it shall be transmitted as the escape character $5C_{HEX}$, which represents the character "¥" in the default encodation for QR code in accordance with JIS X 0201 (in the AIM ECI specification and the ASCII character set this "¥" character value maps to the backslash character "\"). The codeword(s) representing the ECI Designator are converted into a 6 digit number by inverting the rules defined in Table 4. These 6 digits shall be transmitted as the corresponding 8-bit values in the range 30_{HEX} to 39_{HEX}, immediately following the escape character.

Application software recognizing **\nnnnnn** should interpret all subsequent characters as being from the ECI defined by the 6 digit designator. This interpretation remains in effect until:

a) the end of the encoded data

b) a change to a new ECI signaled by Mode Indicator **0111**, subject to rules defined by the AIM ECI specification.

When reverting to the default interpretation the decoder shall output the appropriate escape sequence as prefix to the data.

If the character "¥" needs to be used as encoded data, transmission shall be as follows: whenever character $5C_{HEX}$ occurs as data, two bytes of the value shall be transmitted, thus a single occurrence is always an escape character and a double occurrence indicates true data.

Example 1

Encoded data: ABC¥1234

Transmitted data: ABC¥¥1234

Encoded data: ABC followed by <further data> encoded according to rules for ECI 123456.

Transmitted data: ABC¥123456<further data>

Example 2 (using data in 8.4.1.1)

The message contains ECI Mode Indicator/ECI Designator/Mode Indicator/Character count indicator/Data in the form of

0111 00001001 0100 00000101 10100001 10100010 10100011 10100100 10100101

Symbology Identifier **]Q2** (see Annex F) must be added to the data transmission.

Transmission (hex. values): **5D 51 32 5C 30 30 30 30 30 39 A1 A2 A3 A4 A5**

Encoded data in ECI 000009: **ΑΒΓΔΕ**

In Structured Append Mode, when the ECI Mode Indicator is encountered at the beginning of the symbol, subsequent data characters shall be interpreted as being from the ECI(s) in force at the end of the preceding symbol.

Notes: The backslash character "\", ASCII value $5C_{HEX}$, is equivalent to "¥" in JIS X 0201.

15.3 FNC1

In the modes with implied FNC1 in either first or second position, this implied character cannot be transmitted directly as there is no ASCII or JIS8 value corresponding to it. It is therefore necessary to indicate its presence in the first or second position by the transmission of the relevant Symbology Identifier (**]Q3,]Q4,]Q5** or **]Q6**). Elsewhere in these symbols it may occur in accordance with the relevant application specification as a data field separator, represented in Alphanumeric Mode by the character **%** and in 8-bit Byte Mode by the character **GS** (ASCII/JIS8 value $1D_{HEX}$). In both cases the decoder shall transmit ASCII/JIS8 value $1D_{HEX}$.

If the character **%** is used as part of the encoded data while in Alphanumeric Mode, it is represented in the symbol by **%%**. If this is encountered the decoder shall transmit a single **%** character.

Annex A
(normative)

Error detection and correction generator polynomials

The check character generation polynomial is used to divide the data codeword polynomial, where each codeword is the coefficient of the dividend polynomial in descending power order. The coefficients of the remainder of this division are the error correction codeword values.

Tables A.1 to A.7 show the generator polynomials for the error correction codes which are used for each Version and Level. In the table, α is the primitive element 2 under GF(2^8). Each generator polynomial is the product of the first degree polynomials: $x-2^0$, $x-2^1$, ..., $x-2^{n-1}$; where n is the degree of the generator polynomial.

Table A.1 — Generator polynomials for 7 - 22 error correction codewords

Number of error correction codewords	Generator polynomials
7	$x^7+\alpha^{87}x^6+\alpha^{229}x^5+\alpha^{146}x^4+\alpha^{149}x^3+\alpha^{238}x^2+\alpha^{102}x+\alpha^{21}$
10	$x^{10}+\alpha^{251}x^9+\alpha^{67}x^8+\alpha^{46}x^7+\alpha^{61}x^6+\alpha^{118}x^5+\alpha^{70}x^4+\alpha^{64}x^3$ $+\alpha^{94}x^2+\alpha^{32}x+\alpha^{45}$
13	$x^{13}+\alpha^{74}x^{12}+\alpha^{152}x^{11}+\alpha^{176}x^{10}+\alpha^{100}x^9+\alpha^{86}x^8+\alpha^{100}x^7+\alpha^{106}x^6$ $+\alpha^{104}x^5+\alpha^{130}x^4+\alpha^{218}x^3+\alpha^{206}x^2+\alpha^{140}x+\alpha^{78}$
15	$x^{15}+\alpha^{8}x^{14}+\alpha^{183}x^{13}+\alpha^{61}x^{12}+\alpha^{91}x^{11}+\alpha^{202}x^{10}+\alpha^{37}x^9+\alpha^{51}x^8$ $+\alpha^{58}x^7+\alpha^{58}x^6+\alpha^{237}x^5+\alpha^{140}x^4+\alpha^{124}x^3+\alpha^{5}x^2+\alpha^{99}x+\alpha^{105}$
16	$x^{16}+\alpha^{120}x^{15}+\alpha^{104}x^{14}+\alpha^{107}x^{13}+\alpha^{109}x^{12}+\alpha^{102}x^{11}+\alpha^{161}x^{10}$ $+\alpha^{76}x^9+\alpha^{3}x^8+\alpha^{91}x^7+\alpha^{191}x^6+\alpha^{147}x^5+\alpha^{169}x^4\alpha^{182}x^3$ $+\alpha^{194}x^2+\alpha^{225}x+\alpha^{120}$
17	$x^{17}+\alpha^{43}x^{16}+\alpha^{139}x^{15}+\alpha^{206}x^{14}+\alpha^{78}x^{13}+\alpha^{43}x^{12}+\alpha^{239}x^{11}$ $+\alpha^{123}x^{10}+\alpha^{206}x^9+\alpha^{214}x^8+\alpha^{147}x^7+\alpha^{24}x^6+\alpha^{99}x^5+\alpha^{150}x^4$ $+\alpha^{39}x^3+\alpha^{243}x^2+\alpha^{163}x+\alpha^{136}$
18	$x^{18}+\alpha^{215}x^{17}+\alpha^{234}x^{16}+\alpha^{158}x^{15}+\alpha^{94}x^{14}+\alpha^{184}x^{13}+\alpha^{97}x^{12}$ $+\alpha^{118}x^{11}+\alpha^{170}x^{10}+\alpha^{79}x^9+\alpha^{187}x^8+\alpha^{152}x^7+\alpha^{148}x^6$ $+\alpha^{252}x^5+\alpha^{179}x^4+\alpha^{5}x^3+\alpha^{98}x^2+\alpha^{96}x+\alpha^{153}$
20	$x^{20}+\alpha^{17}x^{19}+\alpha^{60}x^{18}+\alpha^{79}x^{17}+\alpha^{50}x^{16}+\alpha^{61}x^{15}+\alpha^{163}x^{14}+\alpha^{26}x^{13}$ $+\alpha^{187}x^{12}+\alpha^{202}x^{11}+\alpha^{180}x^{10}+\alpha^{221}x^9+\alpha^{225}x^8+\alpha^{83}x^7$ $+\alpha^{239}x^6+\alpha^{156}x^5+\alpha^{164}x^4+\alpha^{212}x^3+\alpha^{212}x^2+\alpha^{188}x+\alpha^{190}$
22	$x^{22}+\alpha^{210}x^{21}+\alpha^{171}x^{20}+\alpha^{247}x^{19}+\alpha^{242}x^{18}+\alpha^{93}x^{17}+\alpha^{230}x^{16}$ $+\alpha^{14}x^{15}+\alpha^{109}x^{14}+\alpha^{221}x^{13}+\alpha^{53}x^{12}+\alpha^{200}x^{11}+\alpha^{74}x^{10}+\alpha^{8}x^9$ $+\alpha^{172}x^8+\alpha^{98}x^7+\alpha^{80}x^6+\alpha^{219}x^5+\alpha^{134}x^4+\alpha^{160}x^3+\alpha^{105}x^2$ $+\alpha^{165}x+\alpha^{231}$

Table A.2 — Generator polynomials for 24 - 36 error correction codewords

Number of error correction codewords	Generator polynomials
24	$x^{24}+\alpha^{229}x^{23}+\alpha^{121}x^{22}+\alpha^{135}x^{21}+\alpha^{48}x^{20}+\alpha^{211}x^{19}+\alpha^{117}x^{18}$ $\alpha^{251}x^{17}+\alpha^{126}x^{16}+\alpha^{159}x^{15}+\alpha^{180}x^{14}+\alpha^{169}x^{13}+\alpha^{152}x^{12}$ $+\alpha^{192}x^{11}+\alpha^{226}x^{10}+\alpha^{228}x^{9}+\alpha^{218}x^{8}+\alpha^{111}x^{7}+x^{6}+\alpha^{117}x^{5}$ $+\alpha^{232}x^{4}+\alpha^{87}x^{3}+\alpha^{96}x^{2}+\alpha^{227}x+\alpha^{21}$
26	$x^{26}+\alpha^{173}x^{25}+\alpha^{125}x^{24}+\alpha^{158}x^{23}+\alpha^{2}x^{22}+\alpha^{103}x^{21}+\alpha^{182}x^{20}$ $+\alpha^{118}x^{19}+\alpha^{17}x^{18}+\alpha^{145}x^{17}+\alpha^{201}x^{16}+\alpha^{111}x^{15}+\alpha^{28}x^{14}$ $+\alpha^{165}x^{13}+\alpha^{53}x^{12}+\alpha^{161}x^{11}+\alpha^{21}x^{10}+\alpha^{245}x^{9}+\alpha^{142}x^{8}$ $+\alpha^{13}x^{7}+\alpha^{102}x^{6}+\alpha^{48}x^{5}+\alpha^{227}x^{4}+\alpha^{153}x^{3}+\alpha^{145}x^{2}+\alpha^{218}x$ $+\alpha^{70}$
28	$x^{28}+\alpha^{168}x^{27}+\alpha^{223}x^{26}+\alpha^{200}x^{25}+\alpha^{104}x^{24}+\alpha^{224}x^{23}+\alpha^{234}x^{22}$ $+\alpha^{108}x^{21}+\alpha^{180}x^{20}+\alpha^{110}x^{19}+\alpha^{190}x^{18}+\alpha^{195}x^{17}+\alpha^{147}x^{16}$ $+\alpha^{205}x^{15}+\alpha^{27}x^{14}+\alpha^{232}x^{13}+\alpha^{201}x^{12}+\alpha^{21}x^{11}+\alpha^{43}x^{10}$ $+\alpha^{245}x^{9}+\alpha^{87}x^{8}+\alpha^{42}x^{7}+\alpha^{195}x^{6}+\alpha^{212}x^{5}+\alpha^{119}x^{4}+\alpha^{242}x^{3}$ $+\alpha^{37}x^{2}+\alpha^{9}x+\alpha^{123}$
30	$x^{30}+\alpha^{41}x^{29}+\alpha^{173}x^{28}+\alpha^{145}x^{27}+\alpha^{152}x^{26}+\alpha^{216}x^{25}+\alpha^{31}x^{24}$ $+\alpha^{179}x^{23}+\alpha^{182}x^{22}+\alpha^{50}x^{21}+\alpha^{48}x^{20}+\alpha^{110}x^{19}+\alpha^{86}x^{18}$ $+\alpha^{239}x^{17}+\alpha^{96}x^{16}+\alpha^{222}x^{15}+\alpha^{125}x^{14}+\alpha^{42}x^{13}+\alpha^{173}x^{12}$ $+\alpha^{226}x^{11}+\alpha^{193}x^{10}+\alpha^{224}x^{9}+\alpha^{130}x^{8}+\alpha^{156}x^{7}+\alpha^{37}x^{6}$ $+\alpha^{251}x^{5}+\alpha^{216}x^{4}+\alpha^{238}x^{3}+\alpha^{40}x^{2}+\alpha^{192}x+\alpha^{180}$
32	$x^{32}\alpha^{10}x^{31}+\alpha^{6}x^{30}+\alpha^{106}x^{29}+\alpha^{190}x^{28}+\alpha^{249}x^{27}+\alpha^{167}x^{26}$ $+\alpha^{4}x^{25}+\alpha^{67}x^{24}+\alpha^{209}x^{23}+\alpha^{138}x^{22}+\alpha^{138}x^{21}+\alpha^{32}x^{20}$ $+\alpha^{242}x^{19}+\alpha^{123}x^{18}+\alpha^{89}x^{17}+\alpha^{27}x^{16}+\alpha^{120}x^{15}+\alpha^{185}x^{14}$ $+\alpha^{80}x^{13}+\alpha^{156}x^{12}+\alpha^{38}x^{11}+\alpha^{69}x^{10}+\alpha^{171}x^{9}+\alpha^{60}x^{8}+\alpha^{28}x^{7}$ $+\alpha^{222}x^{6}+\alpha^{80}x^{5}+\alpha^{52}x^{4}+\alpha^{254}x^{3}+\alpha^{185}x^{2}+\alpha^{220}x+\alpha^{241}$
34	$x^{34}+\alpha^{111}x^{33}+\alpha^{77}x^{32}+\alpha^{146}x^{31}+\alpha^{94}x^{30}+\alpha^{26}x^{29}+\alpha^{21}x^{28}$ $+\alpha^{108}x^{27}+\alpha^{19}x^{26}+\alpha^{105}x^{25}+\alpha^{94}x^{24}+\alpha^{113}x^{23}+\alpha^{193}x^{22}$ $+\alpha^{86}x^{21}+\alpha^{140}x^{20}+\alpha^{163}x^{19}+\alpha^{125}x^{18}+\alpha^{58}x^{17}+\alpha^{158}x^{16}$ $+\alpha^{229}x^{15}+\alpha^{239}x^{14}+\alpha^{218}x^{13}+\alpha^{103}x^{12}+\alpha^{56}x^{11}+\alpha^{70}x^{10}$ $+\alpha^{114}x^{9}+\alpha^{61}x^{8}+\alpha^{183}x^{7}+\alpha^{129}x^{6}+\alpha^{167}x^{5}+\alpha^{13}x^{4}+\alpha^{98}x^{3}$ $+\alpha^{62}x^{2}+\alpha^{129}x+\alpha^{51}$
36	$x^{36}+\alpha^{200}x^{35}+\alpha^{183}x^{34}+\alpha^{98}x^{33}+\alpha^{16}x^{32}+\alpha^{172}x^{31}+\alpha^{31}x^{30}$ $+\alpha^{246}x^{29}+\alpha^{234}x^{28}+\alpha^{60}x^{27}+\alpha^{152}x^{26}+\alpha^{115}x^{25}+x^{24}$ $+\alpha^{167}x^{23}+\alpha^{152}x^{22}+\alpha^{113}x^{21}+\alpha^{248}x^{20}+\alpha^{238}x^{19}+\alpha^{107}x^{18}$ $+\alpha^{18}x^{17}+\alpha^{63}x^{16}+\alpha^{218}x^{15}+\alpha^{37}x^{14}+\alpha^{87}x^{13}+\alpha^{210}x^{12}$ $+\alpha^{105}x^{11}+\alpha^{177}x^{10}+\alpha^{120}x^{9}+\alpha^{74}x^{8}+\alpha^{121}x^{7}+\alpha^{196}x^{6}$ $+\alpha^{117}x^{5}+\alpha^{251}x^{4}+\alpha^{113}x^{3}+\alpha^{233}x^{2}+\alpha^{30}x+\alpha^{120}$

Table A.3 — Generator polynomials for 40 - 46 error correction codewords

Number of error correction codewords	Generator polynomials
40	$x^{40}+\alpha^{59}x^{39}+\alpha^{116}x^{38}+\alpha^{79}x^{37}+\alpha^{161}x^{36}+\alpha^{252}x^{35}+\alpha^{98}x^{34}$ $+\alpha^{128}x^{33}+\alpha^{205}x^{32}+\alpha^{128}x^{31}+\alpha^{161}x^{30}+\alpha^{247}x^{29}+\alpha^{57}x^{28}$ $+\alpha^{163}x^{27}+\alpha^{56}x^{26}+\alpha^{235}x^{25}+\alpha^{106}x^{24}+\alpha^{53}x^{23}+\alpha^{26}x^{22}$ $+\alpha^{187}x^{21}+\alpha^{174}x^{20}+\alpha^{226}x^{19}+\alpha^{104}x^{18}+\alpha^{170}x^{17}+\alpha^{7}x^{16}$ $+\alpha^{175}x^{15}+\alpha^{35}x^{14}+\alpha^{181}x^{13}+\alpha^{114}x^{12}+\alpha^{88}x^{11}+\alpha^{41}x^{10}$ $+\alpha^{47}x^{9}+\alpha^{163}x^{8}+\alpha^{125}x^{7}+\alpha^{134}x^{6}+\alpha^{72}x^{5}+\alpha^{20}x^{4}+\alpha^{232}x^{3}$ $+\alpha^{53}x^{2}+\alpha^{35}x+\alpha^{15}$
42	$x^{42}+\alpha^{250}x^{41}+\alpha^{103}x^{40}+\alpha^{221}x^{39}+\alpha^{230}x^{38}+\alpha^{25}x^{37}+\alpha^{18}x^{36}$ $+\alpha^{137}x^{35}+\alpha^{231}x^{34}+x^{33}+\alpha^{3}x^{32}+\alpha^{58}x^{31}+\alpha^{242}x^{30}+\alpha^{221}x^{29}$ $+\alpha^{191}x^{28}+\alpha^{110}x^{27}+\alpha^{84}x^{26}+\alpha^{230}x^{25}+\alpha^{8}x^{24}+\alpha^{188}x^{23}$ $+\alpha^{106}x^{22}+\alpha^{96}x^{21}+\alpha^{147}x^{20}+\alpha^{15}x^{19}+\alpha^{131}x^{18}+\alpha^{139}x^{17}$ $+\alpha^{34}x^{16}+\alpha^{101}x^{15}+\alpha^{223}x^{14}+\alpha^{39}x^{13}+\alpha^{101}x^{12}+\alpha^{213}x^{11}$ $+\alpha^{199}x^{10}+\alpha^{237}x^{9}+\alpha^{254}x^{8}+\alpha^{201}x^{7}+\alpha^{123}x^{6}+\alpha^{171}x^{5}$ $+\alpha^{162}x^{4}+\alpha^{194}x^{3}+\alpha^{117}x^{2}+\alpha^{50}x+\alpha^{96}$
44	$x^{44}+\alpha^{190}x^{43}+\alpha^{7}x^{42}+\alpha^{61}x^{41}+\alpha^{121}x^{40}+\alpha^{71}x^{39}+\alpha^{246}x^{38}$ $+\alpha^{69}x^{37}+\alpha^{55}x^{36}+\alpha^{168}x^{35}+\alpha^{188}x^{34}+\alpha^{89}x^{33}+\alpha^{243}x^{32}$ $+\alpha^{191}x^{31}+\alpha^{25}x^{30}+\alpha^{72}x^{29}+\alpha^{123}x^{28}+\alpha^{9}x^{27}+\alpha^{145}x^{26}$ $+\alpha^{14}x^{25}+\alpha^{247}x^{24}+\alpha x^{23}+\alpha^{238}x^{22}+\alpha^{44}x^{21}+\alpha^{78}x^{20}$ $+\alpha^{143}x^{19}+\alpha^{62}x^{18}+\alpha^{224}x^{17}+\alpha^{126}x^{16}+\alpha^{118}x^{15}+\alpha^{114}x^{14}$ $+\alpha^{68}x^{13}+\alpha^{163}x^{12}+\alpha^{52}x^{11}+\alpha^{194}x^{10}+\alpha^{217}x^{9}+\alpha^{147}x^{8}$ $+\alpha^{204}x^{7}+\alpha^{169}x^{6}+\alpha^{37}x^{5}+\alpha^{130}x^{4}+\alpha^{113}x^{3}+\alpha^{102}x^{2}+\alpha^{73}x+\alpha^{181}$
46	$x^{46}+\alpha^{112}x^{45}+\alpha^{94}x^{44}+\alpha^{88}x^{43}+\alpha^{112}x^{42}+\alpha^{253}x^{41}+\alpha^{224}x^{40}$ $+\alpha^{202}x^{39}+\alpha^{115}x^{38}+\alpha^{187}x^{37}+\alpha^{99}x^{36}+\alpha^{89}x^{35}+\alpha^{5}x^{34}$ $+\alpha^{54}x^{33}+\alpha^{113}x^{32}+\alpha^{129}x^{31}+\alpha^{44}x^{30}+\alpha^{58}x^{29}+\alpha^{16}x^{28}$ $+\alpha^{135}x^{27}+\alpha^{216}x^{26}+\alpha^{169}x^{25}+\alpha^{211}x^{24}+\alpha^{36}x^{23}+\alpha x^{22}$ $+\alpha^{4}x^{21}+\alpha^{96}x^{20}+\alpha^{60}x^{19}+\alpha^{241}x^{18}+\alpha^{73}x^{17}+\alpha^{104}x^{16}$ $+\alpha^{234}x^{15}+\alpha^{8}x^{14}+\alpha^{249}x^{13}+\alpha^{245}x^{12}+\alpha^{119}x^{11}+\alpha^{174}x^{10}$ $+\alpha^{52}x^{9}+\alpha^{25}x^{8}+\alpha^{157}x^{7}+\alpha^{224}x^{6}+\alpha^{43}x^{5}+\alpha^{202}x^{4}+\alpha^{223}x^{3}$ $+\alpha^{19}x^{2}+\alpha^{82}x+\alpha^{15}$

Table A.4 — Generator polynomials for 48 - 54 error correction codewords

Number of error correction codewords	Generator polynomials
48	$x^{48}+\alpha^{228}x^{47}+\alpha^{25}x^{46}+\alpha^{196}x^{45}+\alpha^{130}x^{44}+\alpha^{211}x^{43}+\alpha^{146}x^{42}$ $+\alpha^{60}x^{41}+\alpha^{24}x^{40}+\alpha^{251}x^{39}+\alpha^{90}x^{38}+\alpha^{39}x^{37}+\alpha^{102}x^{36}$ $+\alpha^{240}x^{35}+\alpha^{61}x^{34}+\alpha^{178}x^{33}+\alpha^{63}x^{32}+\alpha^{46}x^{31}+\alpha^{123}x^{30}$ $+\alpha^{115}x^{29}+\alpha^{18}x^{28}+\alpha^{221}x^{27}+\alpha^{111}x^{26}+\alpha^{135}x^{25}+\alpha^{160}x^{24}$ $+\alpha^{182}x^{23}+\alpha^{205}x^{22}+\alpha^{107}x^{21}+\alpha^{206}x^{20}+\alpha^{95}x^{19}+\alpha^{150}x^{18}$ $+\alpha^{120}x^{17}+\alpha^{184}x^{16}+\alpha^{91}x^{15}+\alpha^{21}x^{14}+\alpha^{247}x^{13}+\alpha^{156}x^{12}$ $+\alpha^{140}x^{11}+\alpha^{238}x^{10}+\alpha^{191}x^{9}+\alpha^{11}x^{8}+\alpha^{94}x^{7}+\alpha^{227}x^{6}$ $+\alpha^{84}x^{5}+\alpha^{50}x^{4}+\alpha^{163}x^{3}+\alpha^{39}x^{2}+\alpha^{34}x+\alpha^{108}$
50	$x^{50}+\alpha^{232}x^{49}+\alpha^{125}x^{48}+\alpha^{157}x^{47}+\alpha^{161}x^{46}+\alpha^{164}x^{45}+\alpha^{9}x^{44}$ $+\alpha^{118}x^{43}+\alpha^{46}x^{42}+\alpha^{209}x^{41}+\alpha^{99}x^{40}+\alpha^{203}x^{39}+\alpha^{193}x^{38}$ $+\alpha^{35}x^{37}+\alpha^{3}x^{36}+\alpha^{209}x^{35}+\alpha^{111}x^{34}+\alpha^{195}x^{33}+\alpha^{242}x^{32}$ $+\alpha^{203}x^{31}+\alpha^{225}x^{30}+\alpha^{46}x^{29}+\alpha^{13}x^{28}+\alpha^{32}x^{27}+\alpha^{160}x^{26}$ $+\alpha^{126}x^{25}+\alpha^{209}x^{24}+\alpha^{130}x^{23}+\alpha^{160}x^{22}+\alpha^{242}x^{21}+\alpha^{215}x^{20}$ $+\alpha^{242}x^{19}+\alpha^{75}x^{18}+\alpha^{77}x^{17}+\alpha^{42}x^{16}+\alpha^{189}x^{15}+\alpha^{32}x^{14}$ $+\alpha^{113}x^{13}+\alpha^{65}x^{12}+\alpha^{124}x^{11}+\alpha^{69}x^{10}+\alpha^{228}x^{9}+\alpha^{114}x^{8}$ $+\alpha^{235}x^{7}+\alpha^{175}x^{6}+\alpha^{124}x^{5}+\alpha^{170}x^{4}+\alpha^{215}x^{3}+\alpha^{232}x^{2}$ $+\alpha^{133}x+\alpha^{205}$
52	$x^{52}+\alpha^{116}x^{51}+\alpha^{50}x^{50}+\alpha^{86}x^{49}+\alpha^{186}x^{48}+\alpha^{50}x^{47}+\alpha^{220}x^{46}$ $+\alpha^{251}x^{45}+\alpha^{89}x^{44}+\alpha^{192}x^{43}+\alpha^{46}x^{42}+\alpha^{86}x^{41}+\alpha^{127}x^{40}$ $+\alpha^{124}x^{39}+\alpha^{19}x^{38}+\alpha^{184}x^{37}+\alpha^{233}x^{36}+\alpha^{151}x^{35}+\alpha^{215}x^{34}$ $+\alpha^{22}x^{33}+\alpha^{14}x^{32}+\alpha^{59}x^{31}+\alpha^{145}x^{30}+\alpha^{37}x^{29}+\alpha^{242}x^{28}$ $+\alpha^{203}x^{27}+\alpha^{134}x^{26}+\alpha^{254}x^{25}+\alpha^{89}x^{24}+\alpha^{190}x^{23}+\alpha^{94}x^{22}$ $+\alpha^{59}x^{21}+\alpha^{65}x^{20}+\alpha^{124}x^{19}+\alpha^{113}x^{18}+\alpha^{100}x^{17}+\alpha^{233}x^{16}$ $+\alpha^{235}x^{15}+\alpha^{121}x^{14}+\alpha^{22}x^{13}+\alpha^{76}x^{12}+\alpha^{86}x^{11}+\alpha^{97}x^{10}$ $+\alpha^{39}x^{9}+\alpha^{242}x^{8}+\alpha^{200}x^{7}+\alpha^{220}x^{6}+\alpha^{101}x^{5}+\alpha^{33}x^{4}+\alpha^{239}x^{3}$ $+\alpha^{254}x^{2}+\alpha^{116}x+\alpha^{51}$
54	$x^{54}+\alpha^{183}x^{53}+\alpha^{26}x^{52}+\alpha^{201}x^{51}+\alpha^{87}x^{50}+\alpha^{210}x^{49}+\alpha^{221}x^{48}$ $+\alpha^{113}x^{47}+\alpha^{21}x^{46}+\alpha^{46}x^{45}+\alpha^{65}x^{44}+\alpha^{45}x^{43}+\alpha^{50}x^{42}$ $+\alpha^{238}x^{41}+\alpha^{184}x^{40}+\alpha^{249}x^{39}+\alpha^{225}x^{38}+\alpha^{102}x^{37}+\alpha^{58}x^{36}$ $+\alpha^{209}x^{35}+\alpha^{218}x^{34}+\alpha^{109}x^{33}+\alpha^{165}x^{32}+\alpha^{26}x^{31}+\alpha^{95}x^{30}$ $+\alpha^{184}x^{29}+\alpha^{192}x^{28}+\alpha^{52}x^{27}+\alpha^{245}x^{26}+\alpha^{35}x^{25}+\alpha^{254}x^{24}$ $+\alpha^{238}x^{23}+\alpha^{175}x^{22}+\alpha^{172}x^{21}+\alpha^{79}x^{20}+\alpha^{123}x^{19}+\alpha^{25}x^{18}$ $+\alpha^{122}x^{17}+\alpha^{43}x^{16}+\alpha^{120}x^{15}+\alpha^{108}x^{14}+\alpha^{215}x^{13}+\alpha^{80}x^{12}$ $+\alpha^{128}x^{11}+\alpha^{201}x^{10}+\alpha^{235}x^{9}+\alpha^{8}x^{8}+\alpha^{153}x^{7}+\alpha^{59}x^{6}+\alpha^{101}x^{5}$ $+\alpha^{31}x^{4}+\alpha^{198}x^{3}+\alpha^{76}x^{2}+\alpha^{31}x+\alpha^{156}$

Table A.5 — Generator polynomials for 56 - 60 error correction codewords

Number of error correction codewords	Generator polynomials
56	$x^{56}+\alpha^{106}x^{55}+\alpha^{120}x^{54}+\alpha^{107}x^{53}+\alpha^{157}x^{52}+\alpha^{164}x^{51}+\alpha^{216}x^{50}$ $+\alpha^{112}x^{49}+\alpha^{116}x^{48}+\alpha^{2}x^{47}+\alpha^{91}x^{46}+\alpha^{248}x^{45}+\alpha^{163}x^{44}$ $+\alpha^{36}x^{43}+\alpha^{201}x^{42}+\alpha^{202}x^{41}+\alpha^{229}x^{40}+\alpha^{6}x^{39}+\alpha^{144}x^{38}$ $+\alpha^{254}x^{37}+\alpha^{155}x^{36}+\alpha^{135}x^{35}+\alpha^{208}x^{34}+\alpha^{170}x^{33}+\alpha^{209}x^{32}$ $+\alpha^{12}x^{31}+\alpha^{139}x^{30}+\alpha^{127}x^{29}+\alpha^{142}x^{28}+\alpha^{182}x^{27}+\alpha^{249}x^{26}$ $+\alpha^{177}x^{25}+\alpha^{174}x^{24}+\alpha^{190}x^{23}+\alpha^{28}x^{22}+\alpha^{10}x^{21}+\alpha^{85}x^{20}$ $+\alpha^{239}x^{19}+\alpha^{184}x^{18}+\alpha^{101}x^{17}+\alpha^{124}x^{16}+\alpha^{152}x^{15}+\alpha^{206}x^{14}$ $+\alpha^{96}x^{13}+\alpha^{23}x^{12}+\alpha^{163}x^{11}+\alpha^{61}x^{10}+\alpha^{27}x^{9}+\alpha^{196}x^{8}$ $+\alpha^{247}x^{7}+\alpha^{151}x^{6}+\alpha^{154}x^{5}+\alpha^{202}x^{4}+\alpha^{207}x^{3}+\alpha^{20}x^{2}+\alpha^{61}x+\alpha^{10}$
58	$x^{58}+\alpha^{82}x^{57}+\alpha^{116}x^{56}+\alpha^{26}x^{55}+\alpha^{247}x^{54}+\alpha^{66}x^{53}+\alpha^{27}x^{52}$ $+\alpha^{62}x^{51}+\alpha^{107}x^{50}+\alpha^{252}x^{49}+\alpha^{182}x^{48}+\alpha^{200}x^{47}+\alpha^{185}x^{46}$ $+\alpha^{235}x^{45}+\alpha^{55}x^{44}+\alpha^{251}x^{43}+\alpha^{242}x^{42}+\alpha^{210}x^{41}+\alpha^{144}x^{40}$ $+\alpha^{154}x^{39}+\alpha^{237}x^{38}+\alpha^{176}x^{37}+\alpha^{141}x^{36}+\alpha^{192}x^{35}+\alpha^{248}x^{34}$ $+\alpha^{152}x^{33}+\alpha^{249}x^{32}+\alpha^{206}x^{31}+\alpha^{85}x^{30}+\alpha^{253}x^{29}+\alpha^{142}x^{28}$ $+\alpha^{65}x^{27}+\alpha^{165}x^{26}+\alpha^{125}x^{25}+\alpha^{23}x^{24}+\alpha^{24}x^{23}+\alpha^{30}x^{22}$ $+\alpha^{122}x^{21}+\alpha^{240}x^{20}+\alpha^{214}x^{19}+\alpha^{6}x^{18}+\alpha^{129}x^{17}+\alpha^{218}x^{16}$ $+\alpha^{29}x^{15}+\alpha^{145}x^{14}+\alpha^{127}x^{13}+\alpha^{134}x^{12}+\alpha^{206}x^{11}+\alpha^{245}x^{10}$ $+\alpha^{117}x^{9}+\alpha^{29}x^{8}+\alpha^{41}x^{7}+\alpha^{63}x^{6}+\alpha^{159}x^{5}+\alpha^{142}x^{4}+\alpha^{233}x^{3}$ $+\alpha^{125}x^{2}+\alpha^{148}x+\alpha^{123}$
60	$x^{60}+\alpha^{107}x^{59}+\alpha^{140}x^{58}+\alpha^{26}x^{57}+\alpha^{12}x^{56}+\alpha^{9}x^{55}+\alpha^{141}x^{54}$ $+\alpha^{243}x^{53}+\alpha^{197}x^{52}+\alpha^{226}x^{51}+\alpha^{197}x^{50}+\alpha^{219}x^{49}+\alpha^{45}x^{48}$ $+\alpha^{211}x^{47}+\alpha^{101}x^{46}+\alpha^{219}x^{45}+\alpha^{120}x^{44}+\alpha^{28}x^{43}+\alpha^{181}x^{42}$ $+\alpha^{127}x^{41}+\alpha^{6}x^{40}+\alpha^{100}x^{39}+\alpha^{247}x^{38}+\alpha^{2}x^{37}+\alpha^{205}x^{36}$ $+\alpha^{198}x^{35}+\alpha^{57}x^{34}+\alpha^{115}x^{33}+\alpha^{219}x^{32}+\alpha^{101}x^{31}+\alpha^{109}x^{30}$ $+\alpha^{160}x^{29}+\alpha^{82}x^{28}+\alpha^{37}x^{27}+\alpha^{38}x^{26}+\alpha^{238}x^{25}+\alpha^{49}x^{24}$ $+\alpha^{160}x^{23}+\alpha^{209}x^{22}+\alpha^{121}x^{21}+\alpha^{86}x^{20}+\alpha^{11}x^{19}+\alpha^{124}x^{18}$ $+\alpha^{30}x^{17}+\alpha^{181}x^{16}+\alpha^{84}x^{15}+\alpha^{25}x^{14}+\alpha^{194}x^{13}+\alpha^{87}x^{12}$ $+\alpha^{65}x^{11}+\alpha^{102}x^{10}+\alpha^{190}x^{9}+\alpha^{220}x^{8}+\alpha^{70}x^{7}+\alpha^{27}x^{6}+\alpha^{209}x^{5}$ $+\alpha^{16}x^{4}+\alpha^{89}x^{3}+\alpha^{7}x^{2}+\alpha^{33}x+\alpha^{240}$

Table A.6 — Generator polynomials for 62 - 66 error correction codewords

Number of error correction codewords	Generator polynomials
62	$x^{62}+\alpha^{65}x^{61}+\alpha^{202}x^{60}+\alpha^{113}x^{59}+\alpha^{98}x^{58}+\alpha^{71}x^{57}+\alpha^{223}x^{56}$ $+\alpha^{248}x^{55}+\alpha^{118}x^{54}+\alpha^{214}x^{53}+\alpha^{94}x^{52}+x^{51}+\alpha^{122}x^{50}$ $+\alpha^{37}x^{49}+\alpha^{23}x^{48}+\alpha^{2}x^{47}+\alpha^{228}x^{46}+\alpha^{58}x^{45}+\alpha^{121}x^{44}$ $+\alpha^{7}x^{43}+\alpha^{105}x^{42}+\alpha^{135}x^{41}+\alpha^{78}x^{40}+\alpha^{243}x^{39}+\alpha^{118}x^{38}$ $+\alpha^{70}x^{37}+\alpha^{76}x^{36}+\alpha^{223}x^{35}+\alpha^{89}x^{34}+\alpha^{72}x^{33}+\alpha^{50}x^{32}$ $+\alpha^{70}x^{31}+\alpha^{111}x^{30}+\alpha^{194}x^{29}+\alpha^{17}x^{28}+\alpha^{212}x^{27}+\alpha^{126}x^{26}$ $+\alpha^{181}x^{25}+\alpha^{35}x^{24}+\alpha^{221}x^{23}+\alpha^{117}x^{22}+\alpha^{235}x^{21}+\alpha^{11}x^{20}$ $+\alpha^{229}x^{19}+\alpha^{149}x^{18}+\alpha^{147}x^{17}+\alpha^{123}x^{16}+\alpha^{213}x^{15}+\alpha^{40}x^{14}$ $+\alpha^{115}x^{13}+\alpha^{6}x^{12}+\alpha^{200}x^{11}+\alpha^{100}x^{10}+\alpha^{26}x^{9}+\alpha^{246}x^{8}$ $+\alpha^{182}x^{7}+\alpha^{218}x^{6}+\alpha^{127}x^{5}+\alpha^{215}x^{4}+\alpha^{36}x^{3}+\alpha^{186}x^{2}+\alpha^{110}x+\alpha^{106}$
64	$x^{64}+\alpha^{45}x^{63}+\alpha^{51}x^{62}+\alpha^{175}x^{61}+\alpha^{9}x^{60}+\alpha^{7}x^{59}+\alpha^{158}x^{58}$ $+\alpha^{159}x^{57}+\alpha^{49}x^{56}+\alpha^{68}x^{55}+\alpha^{119}x^{54}+\alpha^{92}x^{53}+\alpha^{123}x^{52}$ $+\alpha^{177}x^{51}+\alpha^{204}x^{50}+\alpha^{187}x^{49}+\alpha^{254}x^{48}+\alpha^{200}x^{47}+\alpha^{78}x^{46}$ $+\alpha^{141}x^{45}+\alpha^{149}x^{44}+\alpha^{119}x^{43}+\alpha^{26}x^{42}+\alpha^{127}x^{41}+\alpha^{53}x^{40}$ $+\alpha^{160}x^{39}+\alpha^{93}x^{38}+\alpha^{199}x^{37}+\alpha^{212}x^{36}+\alpha^{29}x^{35}+\alpha^{24}x^{34}$ $+\alpha^{145}x^{33}+\alpha^{156}x^{32}+\alpha^{208}x^{31}+\alpha^{150}x^{30}+\alpha^{218}x^{29}+\alpha^{209}x^{28}$ $+\alpha^{4}x^{27}+\alpha^{216}x^{26}+\alpha^{91}x^{25}+\alpha^{47}x^{24}+\alpha^{184}x^{23}+\alpha^{146}x^{22}$ $+\alpha^{47}x^{21}+\alpha^{140}x^{20}+\alpha^{195}x^{19}+\alpha^{195}x^{18}+\alpha^{125}x^{17}+\alpha^{242}x^{16}$ $+\alpha^{238}x^{15}+\alpha^{63}x^{14}+\alpha^{99}x^{13}+\alpha^{108}x^{12}+\alpha^{140}x^{11}+\alpha^{230}x^{10}$ $+\alpha^{242}x^{9}+\alpha^{31}x^{8}+\alpha^{204}x^{7}+\alpha^{11}x^{6}+\alpha^{178}x^{5}+\alpha^{243}x^{4}+\alpha^{217}x^{3}$ $+\alpha^{156}x^{2}+\alpha^{213}x+\alpha^{231}$
66	$x^{66}+\alpha^{5}x^{65}+\alpha^{118}x^{64}+\alpha^{222}x^{63}+\alpha^{180}x^{62}+\alpha^{136}x^{61}+\alpha^{136}x^{60}$ $+\alpha^{162}x^{59}+\alpha^{51}x^{58}+\alpha^{46}x^{57}+\alpha^{117}x^{56}+\alpha^{13}x^{55}+\alpha^{215}x^{54}$ $+\alpha^{81}x^{53}+\alpha^{17}x^{52}+\alpha^{139}x^{51}+\alpha^{247}x^{50}+\alpha^{197}x^{49}+\alpha^{171}x^{48}$ $+\alpha^{95}x^{47}+\alpha^{173}x^{46}+\alpha^{65}x^{45}+\alpha^{137}x^{44}+\alpha^{178}x^{43}+\alpha^{68}x^{42}$ $+\alpha^{111}x^{41}+\alpha^{95}x^{40}+\alpha^{101}x^{39}+\alpha^{41}x^{38}+\alpha^{72}x^{37}+\alpha^{214}x^{36}$ $+\alpha^{169}x^{35}+\alpha^{197}x^{34}+\alpha^{95}x^{33}+\alpha^{7}x^{32}+\alpha^{44}x^{31}+\alpha^{154}x^{30}$ $+\alpha^{77}x^{29}+\alpha^{111}x^{28}+\alpha^{236}x^{27}+\alpha^{40}x^{26}+\alpha^{121}x^{25}+\alpha^{143}x^{24}$ $+\alpha^{63}x^{23}+\alpha^{87}x^{22}+\alpha^{80}x^{21}+\alpha^{253}x^{20}+\alpha^{240}x^{19}+\alpha^{126}x^{18}$ $+\alpha^{217}x^{17}+\alpha^{77}x^{16}+\alpha^{34}x^{15}+\alpha^{232}x^{14}+\alpha^{106}x^{13}+\alpha^{50}x^{12}$ $+\alpha^{168}x^{11}+\alpha^{82}x^{10}+\alpha^{76}x^{9}+\alpha^{146}x^{8}+\alpha^{67}x^{7}+\alpha^{106}x^{6}$ $+\alpha^{171}x^{5}+\alpha^{25}x^{4}+\alpha^{132}x^{3}+\alpha^{93}x^{2}+\alpha^{45}x+\alpha^{105}$

Table A.7 — Generator polynomial for 68 error correction codewords

Number of error correction codewords	Generator polynomial
68	$x^{68}+\alpha^{247}x^{67}+\alpha^{159}x^{66}+\alpha^{223}x^{65}+\alpha^{33}x^{64}+\alpha^{224}x^{63}+\alpha^{93}x^{62}$ $+\alpha^{77}x^{61}+\alpha^{70}x^{60}+\alpha^{90}x^{59}+\alpha^{160}x^{58}+\alpha^{32}x^{57}+\alpha^{254}x^{56}$ $+\alpha^{43}x^{55}+\alpha^{150}x^{54}+\alpha^{84}x^{53}+\alpha^{101}x^{52}+\alpha^{190}x^{51}+\alpha^{205}x^{50}$ $+\alpha^{133}x^{49}+\alpha^{52}x^{48}+\alpha^{60}x^{47}+\alpha^{202}x^{46}+\alpha^{165}x^{45}+\alpha^{220}x^{44}$ $+\alpha^{203}x^{43}+\alpha^{151}x^{42}+\alpha^{93}x^{41}+\alpha^{84}x^{40}+\alpha^{15}x^{39}+\alpha^{84}x^{38}$ $+\alpha^{253}x^{37}+\alpha^{173}x^{36}+\alpha^{160}x^{35}+\alpha^{89}x^{34}+\alpha^{227}x^{33}+\alpha^{52}x^{32}$ $+\alpha^{199}x^{31}+\alpha^{97}x^{30}+\alpha^{95}x^{29}+\alpha^{231}x^{28}+\alpha^{52}x^{27}+\alpha^{177}x^{26}$ $+\alpha^{41}x^{25}+\alpha^{125}x^{24}+\alpha^{137}x^{23}+\alpha^{241}x^{22}+\alpha^{166}x^{21}+\alpha^{225}x^{20}$ $+\alpha^{118}x^{19}+\alpha^{2}x^{18}+\alpha^{54}x^{17}+\alpha^{32}x^{16}+\alpha^{82}x^{15}+\alpha^{215}x^{14}$ $+\alpha^{175}x^{13}+\alpha^{198}x^{12}+\alpha^{43}x^{11}+\alpha^{238}x^{10}+\alpha^{235}x^{9}+\alpha^{27}x^{8}$ $+\alpha^{101}x^{7}+\alpha^{184}x^{6}+\alpha^{127}x^{5}+\alpha^{3}x^{4}+\alpha^{5}x^{3}+\alpha^{8}x^{2}+\alpha^{163}x+\alpha^{238}$

Annex B
(normative)

Error correction decoding steps

Take the Version 1-M symbol as an example. For the symbol, the (26, 16, 4) Reed-Solomon code under GF(2^8) is used for error correction. Provided that the code after releasing Masking from the symbol is:

$$R=(r_0 , r_1 , r_2 , \dots , r_{25})$$

That is,

$$R(x)=r_0 + r_1x + r_2x^2 + \dots + r_{25}x^{25}$$

r_i(i=0-25) is an element of GF(2^8)

(i) Calculate the syndrome.

Find the syndrome S_i(i=0-7).

$$S_0 = R(1) = r_0 + r_1 + r_2 + \dots + r_{25}$$

$$S_1 = R(\alpha) = r_0+r_1\alpha+r_2\alpha^2 + \dots + r_{25}\alpha^{25}$$

...

...

$$S_7 = R(\alpha^7) = r_0 + r_1\alpha^7 + r_2\alpha^{14} + \dots + r_{25}\alpha^{175}$$

where α is a primitive element of GF(2^8)

(ii) Find the error position.

$$S_0\sigma_4 - S_1\sigma_3 + S_2\sigma_2 - S_3\sigma_1 + S_4 = 0$$

$$S_1\sigma_4 - S_2\sigma_3 + S_3\sigma_2 - S_4\sigma_1 + S_5 = 0$$

$$S_2\sigma_4 - S_3\sigma_3 + S_4\sigma_2 - S_5\sigma_1 + S_6 = 0$$

$$S_3\sigma_4 - S_4\sigma_3 + S_5\sigma_2 - S_6\sigma_1 + S_7 = 0$$

Find the variable σ_i(i=1-4) for each error position using the above formulas.
Then, substitute the variable for the following polynomial and substitute elements of GF(2^8) one by one.

$$\sigma(x) = \sigma_4 + \sigma_3x + \sigma_2x^2 + \sigma_1x^3 + x^4$$

Now, it is found that an error is on the jth digit (counting from the 0-th digit) for the element αj which makes σ (α)=0.

(iii) Find the error size.

Supposing that an error is on the j1, j2, j4 digits in (ii) above, then find the size of the error.

$$Y_1\alpha j^1 + Y_2\alpha j^2 + Y_3\alpha j^3 + Y_4\alpha j^4 = S_0$$

$$Y_1\alpha^{2j_1} + Y_2\alpha^{2j_2} + Y_3\alpha^{2j_3} + Y_4\alpha^{2j_4} = S_1$$

$$Y_1\alpha^{3j_1} + Y_2\alpha^{3j_2} + Y_3\alpha^{3j_3} + Y_4\alpha^{3j_4} = S_2$$

$$Y_1\alpha^{4j_1} + Y_2\alpha^{4j_2} + Y_3\alpha^{4j_3} + Y_4\alpha^{4j_4} = S_3$$

Solve the above equations to find the size of each error Y_i(i=1-4).

(iv) Correct the error.

Correct the error by adding the complement of the error size value to each error position.

Annex C
(normative)

Format Information

The Format Information consists of a 15 bit sequence comprising 5 data bits and 10 BCH error correction bits. This Annex describes the calculation of the error correction bits and the error correction decoding process.

C.1 Error correction bit calculation

The Bose-Chaudhuri-Hocquenghem (15,5) code shall be used for error correction. The polynomial whose coefficient is the data bit string shall be divided by the generator polynomial $G(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$. The coefficient string of the remainder polynomial shall be appended to the data bit string to form the (15,5) BCH code string. Finally, masking shall be applied by XORing the bit string with **101010000010010** to ensure that the format information bit pattern is not all zeroes for any combination of Mask Pattern and Error Correction Level and to enable Model 2 symbols to be autodiscriminated from Model 1 symbols.

.

Example:

Error Correction level M; Mask Pattern 101

Binary string: **00101**

Polynomial: $\mathbf{x^2 + 1}$

Raise power to the (15 - 5) th: $\mathbf{x^{12} + x^{10}}$

Divide by G(x): $\mathbf{= (x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1)x^2 + (x^7 + x^6 + x^4 + x^3 + x^2)}$

Add coefficient string of above remainder polynomial to Format Information data string:

00101 + 0011011100 ➔ 001010011011100

XOR with mask **101010000010010**

Result: **100000011001110**

Place these bits in the Format Information areas as described in 8.9.

C.2 Error correction decoding steps

Release the masking of the Format Information modules by XORing the bit sequence with the mask pattern **101010000010010**.

This will yield the following code:

$R = (r_0, r_1, r_2, \ldots, r_{14})$

That is,

$R(x) = r_0 + r_1 x + r_2 x^2 + \ldots + r_{14} x^{14}$

where r_i (i = 0-14) is 0 or 1.

Calculate the syndrome.

Find the syndrome S_i (I = 1, 3, 5).

$$S_1 = R(\alpha) = r_0 + r_1\alpha + r_2\alpha^2 + \ldots r_{14}\alpha^{14}$$

$$S_3 = R(\alpha^3) = r_0 + r_1\alpha^3 + r_2\alpha^6 + \ldots r_{14}\alpha^{42}$$

$$S_5 = R(\alpha^5) = r_0 + r_1\alpha^5 + r_2\alpha^{10} + \ldots r_{14}\alpha^{70}$$

where α is a primitive element of GF(2^4).

Find the error position.

$$S_1 + \sigma_1 = 0$$

$$S_3 + S_2\sigma_1 + S_1\sigma_2 + \sigma_3 = 0$$

$$S_5 + S_4\sigma_1 + S_3\sigma_2 + S_2\sigma_3 = 0$$

where: $S_2 = S_1$ squared, $S_4 = S_2$ squared

Find the variable σ_i(i = 1-3) for each error position using the above formulas. Then substitute the variable for the following polynomial and substitute elements of GF(2^4) one by one.

$$\sigma(x) = x^3 + \sigma_1 x^2 + \sigma_2 x + \sigma_3$$

Now, it is found that an error is on the jth digit (counting from the 0-th digit) for the element αj which makes $\sigma(\alpha_j) = 0$.

Correct the error by reversing the bit value of each error position.

Annex D
(normative)

Version Information

The Version Information consists of an 18 bit sequence comprising 6 data bits and 12 BCH error correction bits. This Annex describes the calculation of the error correction bits and the error correction decoding process.

D.1 Error correction bit calculation

The Bose-Chaudhuri-Hocquenghem (18,6) code shall be used for error correction. The polynomial whose coefficient is the data bit string shall be divided by the generator polynomial $G(x) = x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^5 + x^2 + 1$. The coefficient string of the remainder polynomial shall be appended to the data bit string to form the (18,6) BCH code string.

Example:

Version: **7**

Binary string: **000111**

Polynomial: $\boldsymbol{x^2 + x + 1}$

Raise power to the (18 - 6) th: $\boldsymbol{x^{14} + x^{13} + x^{12}}$

Divide by G(x): $\boldsymbol{= (x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^5 + x^2 + 1)x^2 + (x^{11} + x^{10} + x^7 + x^4 + x^2)}$

Add coefficient string of above remainder polynomial to Version Information data string:

000111 + 110010010100 ➔ 000111110010010100

Place these bits in the Version Information areas as described in 8.10.

Table D.1 below shows the full Version Information bit stream for each version.

Table D.1 — Version information bit stream for each version

Version	Version Information bit stream	Hexadecimal equivalent
7	00 0111 1100 1001 0100	07C94
8	00 1000 0101 1011 1100	085BC
9	00 1001 1010 1001 1001	09A99
10	00 1010 0100 1101 0011	0A4D3
11	00 1011 1011 1111 0110	0BBF6
12	00 1100 0111 0110 0010	0C762
13	00 1101 1000 0100 0111	0D847
14	00 1110 0110 0000 1101	0E60D
15	00 1111 1001 0010 1000	0F928
16	01 0000 1011 0111 1000	10B78
17	01 0001 0100 0101 1101	1145D
18	01 0010 1010 0001 0111	12A17
19	01 0011 0101 0011 0010	13532
20	01 0100 1001 1010 0110	149A6
21	01 0101 0110 1000 0011	15683
22	01 0110 1000 1100 1001	168C9
23	01 0111 0111 1110 1100	177EC
24	01 1000 1110 1100 0100	18EC4
25	01 1001 0001 1110 0001	191E1
26	01 1010 1111 1010 1011	1AFAB
27	01 1011 0000 1000 1110	1B08E
28	01 1100 1100 0001 1010	1CC1A
29	01 1101 0011 0011 1111	1D33F
30	01 1110 1101 0111 0101	1ED75
31	01 1111 0010 0101 0000	1F250
32	10 0000 1001 1101 0101	209D5
33	10 0001 0110 1111 0000	216F0
34	10 0010 1000 1011 1010	228BA
35	10 0011 0111 1001 1111	2379F
36	10 0100 1011 0000 1011	24B0B
37	10 0101 0100 0010 1110	2542E
38	10 0110 1010 0110 0100	26A64
39	10 0111 0101 0100 0001	27541
40	10 1000 1100 0110 1001	28C69

D.2 Error correction decoding steps

Provided that the following code has been read from the Version Information area of the symbol:

$R = (r_0, r_1, r_2, \ldots, r_{17})$

That is,

$R(x) = r_0 + r_1 x + r_2 x^2 + \ldots + r_{17} x^{17}$

where r_i (i = 0-17) is 0 or 1.

Calculate the syndrome.

Find the syndrome S_i (I = 1, 3, 5).

$S_1 = R(\alpha) = r_0 + r_1\alpha + r_2\alpha^2 + \ldots r_{17}\alpha^{17}$

$S_3 = R(\alpha^3) = r_0 + r_1\alpha^3 + r_2\alpha^6 + \ldots r_{17}\alpha^{51}$

$S_5 = R(\alpha^5) = r_0 + r_1\alpha^5 + r_2\alpha^{10} + \ldots r_{14}\alpha^{85}$

where α is a primitive element of GF(2^5).

Find the error position.

$S_1 + \sigma_1 = 0$

$S_3 + S_2\sigma_1 + S_1\sigma_2 + \sigma_3 = 0$

$S_5 + S_4\sigma_1 + S_3\sigma_2 + S_2\sigma_3 = 0$

where: $S_2 = (S_1)^2$ and $S_4 = (S_2)^2$

Find the variable σ_i(i = 1-3) for each error position using the above formulas. Then substitute the variable for the following polynomial and substitute elements of GF(2^5) one by one.

$\sigma(x) = x^3 + \sigma_1 x^2 + \sigma_2 x + \sigma_3$

Now, it is found that an error is on the jth digit (counting from the 0-th digit) for the element αj which makes $\sigma(\alpha_j) = 0$.

Correct the error by reversing the bit value of each error position.

Annex E
(normative)

Position of Alignment Patterns

The Alignment Patterns are positioned symmetrically on either side of the diagonal running from the top left corner of the symbol to the bottom right corner. They are spaced as evenly as possible between the Timing Pattern and the opposite side of the symbol, any uneven spacing being accommodated between the Timing Pattern and the first Alignment Pattern in the symbol interior.

Table E.1 below shows, for each version, the number of Alignment Patterns and the row or column coordinates of the center module of each Alignment Pattern.

Table E.1 — Row/column coordinates of center module of Alignment Patterns

Version	Number of Alignment Patterns	Row/Column coordinates of center module						
1	0	-						
2	1	6	18					
3	1	6	22					
4	1	6	26					
5	1	6	30					
6	1	6	34					
7	6	6	22	38				
8	6	6	24	42				
9	6	6	26	46				
10	6	6	28	50				
11	6	6	30	54				
12	6	6	32	58				
13	6	6	34	62				
14	13	6	26	46	66			
15	13	6	26	48	70			
16	13	6	26	50	74			
17	13	6	30	54	78			
18	13	6	30	56	82			
19	13	6	30	58	86			
20	13	6	34	62	90			
21	22	6	28	50	72	94		
22	22	6	26	50	74	98		
23	22	6	30	54	78	102		
24	22	6	28	54	80	106		
25	22	6	32	58	84	110		
26	22	6	30	58	86	114		
27	22	6	34	62	90	118		
28	33	6	26	50	74	98	122	
29	33	6	30	54	78	102	126	
30	33	6	26	52	78	104	130	
31	33	6	30	56	82	108	134	
32	33	6	34	60	86	112	138	
33	33	6	30	58	86	114	142	
34	33	6	34	62	90	118	146	
35	46	6	30	54	78	102	126	150
36	46	6	24	50	76	102	128	154
37	46	6	28	54	80	106	132	158
38	46	6	32	58	84	110	136	162
39	46	6	26	54	82	110	138	166
40	46	6	30	58	86	114	142	170

For example, in a Version 7 symbol the table indicates values 6, 22 and 38. The Alignment Patterns, therefore, are to be centered on (row, column) positions (6,22), (22,6), (22,22), (22,38), (38,22), (38,38). Note that the coordinates (6,6), (6,38), (38,6) are occupied by Position Detection Patterns and are not therefore used for Alignment Patterns.

Annex F
(normative)

Symbology Identifier

The Symbology Identifier assigned to QR Code in ISO/IEC 15424, which should be added as a preamble to the decoded data by a suitably programmed decoder is:

]Qm

where:] is the Symbology Identifier flag (ASCII value 93)

Q is the code character for the QR Code symbology

m is the modifier character with one of the values defined in Table F.1

Table F.1 — Symbology Identifier options and modifier values

Modifier value	Option
0	Model 1 symbol
1	Model 2 symbol, ECI protocol not implemented
2	Model 2 symbol, ECI protocol implemented
3	Model 2 symbol, ECI protocol not implemented, FNC1 implied in first position
4	Model 2 symbol, ECI protocol implemented, FNC1 implied in first position
5	Model 2 symbol, ECI protocol not implemented, FNC1 implied in second position
6	Model 2 symbol, ECI protocol implemented, FNC1 implied in second position

The permissible values of m are: 0, 1, 2, 3, 4, 5, 6

Annex G
(informative)

Symbol encoding example

This Annex describes the encoding of the data string **01234567** into a version 1-M symbol, using the Numeric Mode in accordance with 8.4.2.

Step 1: Data Encodation

- Divide into groups of three digits and convert each group to its 10 or 7 bit binary equivalent:

  012 → 0000001100

  345 → 0101011001

  67 → 1000011

- Convert Character Count Indicator to binary (10 bits for version 1-M)

Character count indicator (8) = **0000001000**

- Connect Mode Indicator for Numeric Mode (**0001**), Character Count Indicator, binary data, and Terminator (**0000**)

0001 0000001000 0000001100 0101011001 1000011 0000

- Divide into 8-bit codewords, adding padding bits (shown underlined for illustration) as needed

00010000 00100000 00001100 01010110 01100001 10000000

- Add Pad codewords to fill data codeword capacity of symbol (for version 1-M, 16 data codewords, therefore 10 Pad codewords required (shown underlined for illustration)), giving the result:

00010000 00100000 00001100 01010110 01100001 10000000 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100

Step 2: Error Correction Codeword generation

Using the Reed-Solomon algorithm to generate the required number of error correction codewords (for a Version 1-M symbol, 10 are needed), these (shown underlined for illustration) should be added to the bit stream, resulting in:

00010000 00100000 00001100 01010110 01100001 10000000 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100 10100101 00100100 11010100 11000001 11101101 00110110 11000111 10000111 00101100 01010101

Step 3: Module placement in matrix.

As there is only a single error correction block in a version 1-M symbol, no interleaving is required in this instance. The Position Detection Patterns and Timing Patterns are placed in a blank 21 × 21 matrix and the module positions for the Format Information are left temporarily blank. The codewords from Step 2 are placed in the matrix in accordance with 8.7.3.

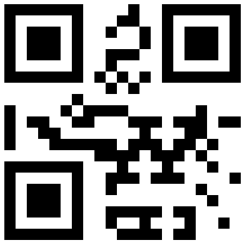


Figure G.1 — Data modules placed in symbol prior to masking

Step 4: Masking Pattern selection

Apply the masking patterns defined in 8.8.1 in turn and evaluate the results in accordance with 8.8.2. The Masking Pattern selected is referenced **011**.

Step 5: Format Information

The error correction level is M and the masking pattern is 011. Therefore, the data bits of the Format Information are **00 011**.

The BCH error correction calculation gives **1101011001** as the bit sequence to be added to the data, giving:

000111101011001 as the unmasked Format Information.

XOR this bit stream with the mask **101010000010010**:

000111101011001 (raw bit stream)

101010000010010 (mask)

101101101001011 (Format Information to be placed in symbol)

Step 6: Final symbol construction

Apply the selected masking pattern to the encoding region of the symbol as described in 8.8, and add Format Information modules in positions reserved in step 3.

Figure G.2 — Final version 1-M symbol encoding 01234567

Annex H
(informative)

Optimisation of bit stream length

As described in this standard, QR Code offers various modes of encodation each of which differs in the number of bits it requires to represent a given data string. Since there is an overlap between the character sets of each mode - for example, numeric data may be encoded in Numeric, Alphanumeric and 8-bit Byte modes, and Latin alphanumeric data may be encoded in Alphanumeric and 8-bit Byte modes - the symbol generation software may need to choose the most appropriate mode in which to encode data characters which appear in more than one mode.

This choice must be made initially and may also be possible part way through a data stream.

A number of alternative approaches may be adapted to minimize the bit stream length. The algorithm will need not only to consider the immediate sequence of characters but also look ahead to the next sequence of data in view of the overhead required for switching modes.

The compaction efficiencies given in 8.4.2 to 8.4.5 need to be interpreted carefully. The best scheme for a given set of data may not be the one with the fewest bits per data character. If the highest degree of compaction is required, account has to be taken of the additional bits required to change modes (additional Mode Indicator and Character Count Indicator). It should also be noted that even if the number of codewords is minimized, the codeword stream may need to be expanded to fill a symbol. This fill process is done using pad characters.

The following guidelines may form the basis of one possible algorithm to determine the shortest bit stream for any given input data. Numbers of characters shown in square brackets e.g. [5,7,9] are applicable to versions 1 - 9, 10 - 26, and 27 - 40 respectively. In the guidelines, the term “exclusive subset of” refers to the set of characters within the character set of a mode which are not shared with the more restricted character set of another mode, e.g. the exclusive subset of the 8-bit byte character set comprises JIS8 values 00_{HEX} - FF_{HEX}, but excludes hexadecimal values 20, 24, 25, 2A, 2B, 2D - 3A, and 41 - 5A; that of the Alphanumeric character set is the set {A - Z, space, $ % * + - . / :}.

1. Select initial mode:

   a) If initial input data is in Kanji character set but in no other, select Kanji mode;

   b) If initial input data is in the exclusive subset of the 8-bit byte character, select 8-bit byte mode;

   c) If initial input data is in the exclusive subset of the Alphanumeric character set AND if there are less than [6,7,8] characters followed by data from the remainder of the 8-bit byte character set, THEN select the 8-bit byte mode ELSE select Alphanumeric mode;

   d) If initial data is numeric, AND if there are less than [4,4,5] characters followed by data from the exclusive subset of the 8-bit byte character set, THEN select 8-bit byte mode ELSE IF there are less than [6,7,8] characters followed by data from the exclusive subset of the Alphanumeric character set THEN select Alphanumeric mode ELSE select Numeric mode;

2. While in 8-bit byte mode:

   a) If one or more Kanji character occurs, switch to Kanji mode;

   b) If a sequence of at least [6,8,9] Numeric characters occurs before more data from the exclusive subset of the 8-bit byte character set, switch to Numeric mode;

c) If a sequence of at least [11,15,16] character from the exclusive subset of the Alphanumeric character set occurs before more data from the exclusive subset of the 8-bit byte character set, switch to Alphanumeric mode;

3. While in Alphanumeric mode:

a) If one or more Kanji character occurs, switch to Kanji mode;

b) If one or more characters from the exclusive subset of the 8-bit byte character set occurs, switch to 8-bit byte mode;

c) If a sequence of at least [13,15,17] Numeric characters occurs before more data from the exclusive subset of the Alphanumeric character set, switch to Numeric mode;

4. While in Numeric mode:

a) If one or more Kanji character occurs, switch to Kanji mode;

b) If one or more characters from the exclusive subset of the 8-bit byte character set occurs, switch to 8-bit byte mode;

c) If one or more characters from the exclusive subset of the Alphanumeric character set occurs, switch to Alphanumeric mode;

Annex I
(informative)

User guidelines for printing and scanning of QR Code symbols

I.1 General

Any QR Code application must be viewed as a total system solution. All the symbology encoding/decoding components (surface markers or printers, labels, readers) making up an installation need to operate together as a system. A failure in any link of the chain, or a mismatch between them, could compromise the performance of the overall system.

While compliance with the specifications is one key to assuring overall system success, other considerations come into play which may influence performance as well. The following guidelines suggest some factors to keep in mind when specifying or implementing bar or matrix code systems:

1. Select a print density which will yield tolerance values that can be achieved by the marking or printing technology being used. Ensure that the module dimension is an integer multiple of the print head pixel dimension (both parallel to and perpendicular to the print direction). Ensure also that any adjustment for print gain (or loss) is performed by changing an equal integer number of pixels from dark to light (or light to dark) on all dark-to-light boundaries of individual or groups of adjoining dark modules in order to ensure that the module center spacing remains constant, although the apparent bit-map representation of the individual dark (or light) modules is adjusted in size to suit the direction of compensation.

2. Choose a reader with a resolution suitable for the symbol density and quality produced by the marking or printing technology.

3. Ensure that the optical properties of the printed symbol are compatible with the wavelength of the scanner light source or sensor.

4. Verify symbol compliance in the final label or package configuration. Overlays, show-through and curved or irregular surfaces can all affect symbol readability.

The effects of specular reflection from glossy symbol surfaces must be considered. Scanning systems must take into account the variations in diffuse reflection between dark and light features. At some scanning angles, the specular component of the reflected light can greatly exceed the desired diffuse component, changing the scanning performance. In cases where the surface of the material or part can be altered, matt, non-glossy surfaces may help minimize specular effects. Where this option is not available, particular must be taken to ensure the illumination of the symbol to be read optimizes the desired contrast components.

I.2 User selection of Model

Model 2 symbols are recommended for all new and open systems applications, because the incorporation of Alignment Patterns greatly assists the reading process in determining the module grid and maintaining its accuracy, and because the availability of symbols up to version 40 provides large data capacities. Model 2 symbols should always be specified because their design is less susceptible to the effects of distortion. Model 1 symbols are limited to use in existing applications.

I.3 User selection of error correction level

The users should define the appropriate level of error correction to suit the application requirements. As shown in Table 12, the four levels from L to H offer increasing capabilities of detecting and correcting errors, at the cost of

some increase in symbol size for a given message length. For example, a Version 20-Q symbol can contain a total of 485 data codewords, but if a lower level of error correction was acceptable, the same data could also be represented in a Version 15-L symbol (exact capacity 523 data codewords).

The error correction level should be determined in relation to:

- the expected level of symbol quality: the lower the expected quality grade, the higher the level to be applied;

- the importance of a high first read rate;

- the opportunity for re-scanning in the event of a read failure;

- the space constraints which might reduce the opportunity to use a higher error correction level.

Error correction level L is appropriate for high symbol quality and/or the need for the smallest possible symbol for given data. Level M is described as “Standard” level and offers a good compromise between small size and increased reliability. Level Q is a “High reliability” level and suitable for more critical or poor print quality applications while level H offers the maximum achievable reliability.

Annex J
(informative)

Autodiscrimination

QR Code may be read by suitably programmed decoders which have been designed to autodiscriminate it from other symbologies. A properly programmed QR Code reader will not decode a symbol in another symbology as a valid QR Code symbol; however, representations of short linear symbols may be found in any matrix symbol including QR Code.

QR Code Model 1 and Model 2 symbols can also be autodiscriminated from each other by a suitable decoder.

The decoder's valid set of symbologies should be limited to those needed by a given application in order to maximize reading security.

Annex K
(informative)

Matrix code print quality guideline

This Annex presents a framework for assessing the print quality of a 2D matrix bar code symbol, which can be adapted to any matrix symbology. Section 11 describes its application to QR Code. The method described here parallels in many ways that of ISO/IEC15416 for assessing print quality of linear bar code symbols. It starts by obtaining a high resolution grey-scale image of the symbol under controlled illumination and viewing conditions. The stored image is then analyzed for the parameters of Decode, Symbol Contrast, "Print" Growth, Axial Nonuniformity, and Unused Error Correction. The final symbol assessment is the lowest grade achieved for these five parameters and any others specified for a given symbology or application.

The procedures presented here must necessarily be augmented by the reference decode algorithm and other measurement details within any companion symbology specification, and they may also be altered or overridden as appropriate by governing symbology or application specifications.

K.1 Obtaining the test image

A test image of the symbol shall be obtained in a configuration that mimics the typical scanning situation for that symbol, but with substantially higher resolution, uniform illumination, and at best focus. Specialized applications clearly must dictate the color and angle of symbol illumination as well as the required imaging resolution, but the following general test setup should work suitably for many open applications.

A standard monochrome video camera shall image the test symbol directly on axis with its center and normal to its plane. The lens used shall be appropriate to frame the entire symbol (including any required quiet zones) in good focus, and with a sufficiently small field of view to minimize optical distortions. Light illumination shall uniformly flood the symbol area from at least two sides with a 45 degree angle of incidence. Test images can be captured with 8-bit grey-scale digitization using standard frame capture equipment, and the grey-scale shall be calibrated using targets of known diffuse reflectance.

Regardless of the exact optical setup, two principles should govern its selection. First, the test image's grey-scale shall be nominally linear and not be adjusted in any way to either enhance contrast or improve appearance. Second, the image resolution shall be adequate to produce consistent readings, which generally requires that the module widths and heights span at least five image pixels.

K.2 Assessing symbol parameters

K.2.1 Decode

A symbology's reference decoding algorithm shall be applied to the test image. If it achieves a valid decode, the Decode grade is "4", otherwise it is "0".

The Decode parameter tests on a Pass/Fail basis whether the symbol, when optimally imaged, has all its features sufficiently correct to be readable. Beyond this; the initial reference decode performs three additional tasks needed for subsequent measurement of the other symbol quality parameters. First, it locates and defines the area covered by the test symbol in the image. Second, it adaptively creates a grid mapping of the data module centers so as to sample them. Third, it performs error correction, detecting if symbol damage has consumed any of the error budget. These images, image coordinates, and error decoding each facilitate one or more of the following measurements.

K.2.2 Symbol Contrast

Within the grey-scale image, all of the image pixels which fall within the area of the test symbol, extending outward to the limits of any required quiet zones, shall be sorted by their reflectance values to select the darkest 10% of the pixels and the lightest 10% of the pixels. Calculate the arithmetic mean of the reflectance of the darkest 10% and the arithmetic mean of the reflectance of the lightest 10%. The difference of the two means is the Symbol Contrast, SC.

The Symbol Contrast grade is determined by

4,0 (A) if SC $\geq$ 70%

3,0 (B) if SC $\geq$ 55%

2,0 (C) if SC $\geq$ 40%

1,0 (D) if SC $\geq$ 20%

0,0 (F) if SC $<$ 20%

Symbol Contrast tests that the two reflective states in the symbol, namely light and dark, are sufficiently and consistently distinct throughout the symbol.

K.2.3 "Print" growth

Calculate a reflectance threshold halfway between the dark and light means from Annex K.2.2. Create a secondary binary image distinguishing dark and light regions using the threshold.

The print growth parameter, the extent to which dark or light markings appropriately fill their module boundaries, is an important indication of process quality which affects reading performance. The particular graphical structures most indicative of element growth or shrinkage from nominal dimensions will vary widely between symbologies, and shall be defined within their specifications, but will generally be either fixed structures or isolated elements whose dimension(s) D is/are determined by counting pixels in the binary digitized image. More than one dimension, for example both horizontal and vertical growth, may be specified and checked independently. Each checked dimension shall have specified both a nominal value D_{NOM} and maximum D_{MAX} and minimum D_{MIN} allowed values. Each measured D shall be normalized to its corresponding nominal and limit values:

$D' = (D - D_{NOM}) / (D_{MAX} - D_{NOM})$ if $D > D_{NOM}$

$D' = (D - D_{NOM}) / (D_{NOM} - D_{MIN})$ otherwise.

Print growth is then graded according to:

4,0 (A) if $-0{,}50 \leq D' \leq 0{,}50$

3,0 (B) if $-0{,}70 \leq D' \leq 0{,}70$

2,0 (C) if $-0{,}85 \leq D' \leq 0{,}85$

1,0 (D) if $-1{,}00 \leq D' \leq 1{,}00$

0,0 (F) if $D' < -1{,}00$ or $D' > 1{,}00$

Print Growth tests that the graphical features comprising the symbol have not grown or shrunk from nominal so much as to hinder readability with less optimum imaging conditions than the test condition.

K.2.4 Axial Nonuniformity

2D matrix symbols include data fields of modules nominally lying in a regular polygonal grid, and any reference decode algorithm must adaptively map the center positions of those modules to extract the data. Axial Nonuniformity measures and grades the spacing of the mapping centers, i.e. the sampling points, in the direction of each of the grid's major axes.

The spacings between adjacent sampling points are independently sorted for each polygonal axis, then the average spacing X_{AVG} along each axis is computed. Axial Nonuniformity is a measure of how much the sampling point spacing differs from one axis to another, namely:

$$AN = abs(X_{AVG} - Y_{AVG}) / ((X_{AVG} + Y_{AVG}) / 2)$$

where abs() yields the absolute value. If a symbology has more than two major axes, then AN is computed for those two average spacings which differ the most. Axial Nonuniformity is then graded as:

4,0 (A) if $AN \leq 0,06$

3,0 (B) if $AN \leq 0,08$

2,0 (C) if $AN \leq 0,10$

1,0 (D) if $AN \leq 0,12$

0,0 (F) if $AN > 0,12$

Axial Nonuniformity tests for uneven scaling of the symbol which would hinder readability at some non-normal viewing angles more than others.

K.2.5 Unused Error Correction

The correction capacity of Reed-Solomon decoding is expressed by the equation:

$$e + 2t \leq d - p$$

where e is the number of erasures

t is the number of errors

d is the number of error correction codewords

p is the number of codewords reserved for error detection

Values for d and p are defined by the symbology specification (often depending on symbol size), while e and t are determined during the successful reference decode. The amount of Unused Error Correction is computed as $UEC = 1,0 - (e+2t)/(d-p)$.

In symbols with more than one (e.g., interleaved) error correction block, UEC shall be calculated for each block independently, then the lowest value graded as:

4,0 (A) if $UEC \geq 0,62$

3,0 (B) if $UEC \geq 0,50$

2,0 (C) if $UEC \geq 0,37$

1,0 (D) if $UEC \geq 0,25$

0,0 (F) if $UEC < 0,25$

The Unused Error Correction parameter tests the extent to which regional or spot damage in the symbol has eroded the reading safety margin that error correction provides.

K.3 Overall symbol grade

The overall symbol grade is the lowest of the parameter grades achieved above. Table K.1 summarizes the test parameters and grade levels.

Table K.1 — Test parameters and values

Grade	Reference decode	Symbol Contrast	"Print" growth	Axial Nonuniformity	Unused Error Correction
4,0 (A)	Passes	SC ≥ 0,70	-0,50 ≤ D' ≤ 0,50	AN ≤ 0,06	UEC ≥ 0,62
3,0(B)		SC ≥ 0,55	-0,70 ≤ D' ≤ 0,70	AN ≤ 0,08	UEC ≥ 0,50
2,0 (C)		SC ≥ 0,40	-0,85 ≤ D' ≤ 0,85	AN ≤ 0,10	UEC ≥ 0,37
1,0 (D)		SC ≥ 0,20	-1,00 ≤ D' ≤ 1,00	AN ≤ 0,12	UEC ≥ 0,25
0,0 (F)	Fails	SC < 0,20	D' < -1,00 or D' > 1,00	AN > 0,12	UEC < 0,25

Quality grades are expressed either on a numeric scale ranging in descending order of quality from 4,0 to 0,0, to one decimal place, in accordance with ISO/IEC15416, or on an equivalent alphabetic scale from A to D, with a failing grade of F, as referred to in ANSI X3.182 (Bar Code Print Quality Guidelines)

Table K.2 maps the numeric and alphabetic grades to each other.

Table K.2 — Equivalence of numeric and alphabetic quality grades

4,0 ≥A ≥ 3,5
3,5 >B ≥2,5
2,5 >C ≥1,5
1,5 >D ≥0,5
0,5 > F

Annex L
(informative)

Process control techniques

This Annex describes tools and procedures useful for monitoring and controlling the process of creating scannable QR Code symbols. These techniques do not constitute a print quality check of the produced symbols - the method defined in 11 and Annex K is the required method for assessing symbol quality - but they individually and collectively yield good indications of whether the symbol production process is creating workable symbols.

L.1 Symbol Contrast

Most verifiers for linear bar code symbols have either a reflectometer mode or a mode for plotting scan reflectance profiles and/or reporting Symbol Contrast, as defined in EN1635, from undecodable scans. Except with symbols requiring special illumination configurations, the symbol contrast readings that can be obtained using a 0,150 mm or 0,250 mm aperture at 660 nm wavelength - either the reported symbol contrast value, the maximum to minimum scan reflectance profile excursions, or the difference between maximum and minimum reflectometer readings - are found to correlate well with an image-derived symbol contrast value. In particular these reading can be used to check that symbol contrast stays well above the minimum allowed for the intended symbol quality grade.

L.2 Assessing Axial Nonuniformity

For any symbol, measure the distance from the left edge of the upper left position detection pattern to the right edge of the upper right position detection pattern, and the distance from the top edge of the upper left position detection pattern to the bottom edge of the lower left position detection pattern. Divide each of these by the number of modules in that dimension. E.g. a version 2 symbol would have 25 as a divisor. Substitute the results for X_{AVG} and Y_{AVG} in the formula in L.2.4 and grade the result for an assessment of Axial Nonuniformity.

L.3 Visual inspection for symbol distortion and defects

Ongoing visual inspection of the Position Detection and Timing Patterns in sample symbols can monitor an important aspect of the production process.

Matrix code symbols are susceptible to errors caused by local distortions of the matrix grid. Any such distortions may show up visually as either crooked edges on the Position Detection Patterns or uneven spacings within the alternating Timing Patterns running between the Position Detection Patterns and aligned with the inner boundaries of these.

The Position Detection Patterns and the adjacent quiet zone areas should always be solidly dark and light. Failures in the print mechanism which may produce defects in the form of light or dark streaks through the symbol should be visibly evident where they traverse the finder pattern or the quiet zone. Such systematic failures in the print process should be corrected.

L.4 Assessing print growth

A linear bar code verifier capable of outputting direct measurements of bar and space patterns may be used for the assessment of print gain or loss in both horizontal and vertical axes, by measuring along two scan paths at right angles, one passing through both upper Position Detection Patterns and crossing the center 3 x 3 block of modules in each, and the other similarly passing through both left-hand Position Detection Patterns. Analysis of the output

should reveal an apparent bar/space/bar/space/bar pattern at each end of the scan path; the print gain (or loss) can be assessed by comparing the five measured element widths with the ideal 1:1:3:1:1 ratio of the widths.

Annex M
(informative)

Characteristics of Model 1 QR Code symbols

Model 1 of QR Code is the form of the symbology used for a number of early or closed systems applications but is not recommended for use in new or open systems applications, or those where data volumes are likely to be high. In most respects it follows the same specification as Model 2 but differs in a number of significant aspects which are detailed in this Annex.

M.1 Model 1 overall characteristics

Model 1 symbols differ from Model 2 in the following ways:

1. Symbol size (not including quiet zone):

   21 × 21 modules to 73 × 73 modules (Versions 1 to 14, increasing in steps of 4 modules per side)

2. Maximum data capacity (for maximum symbol size with lowest level of error correction, Version 14-L):

   - numeric data: 1 167 characters
   - alphanumeric data: 707 characters
   - 8-bit byte data: 486 characters
   - Kanji data: 299 characters

3. Symbol structure:

   - Alignment Patterns: Model 1 symbols have no Alignment Patterns
   - Extension patterns: Model 1 symbols have Extension Patterns on the right-hand and lower sides
   - Version Information : Model 1 symbols contain no Version Information
   - Symbol character placement : in consequence of the above, symbol character placement follows different rules.

4. Error correction: the error detection and correction codewords are calculated identically with Model 2, but the number and size of error correction blocks for any Version differs. Data and error correction codeword blocks are not subject to interleaving.

Figure M.1 below illustrates the structure of a Version 7 Model 1 QR Code symbol.

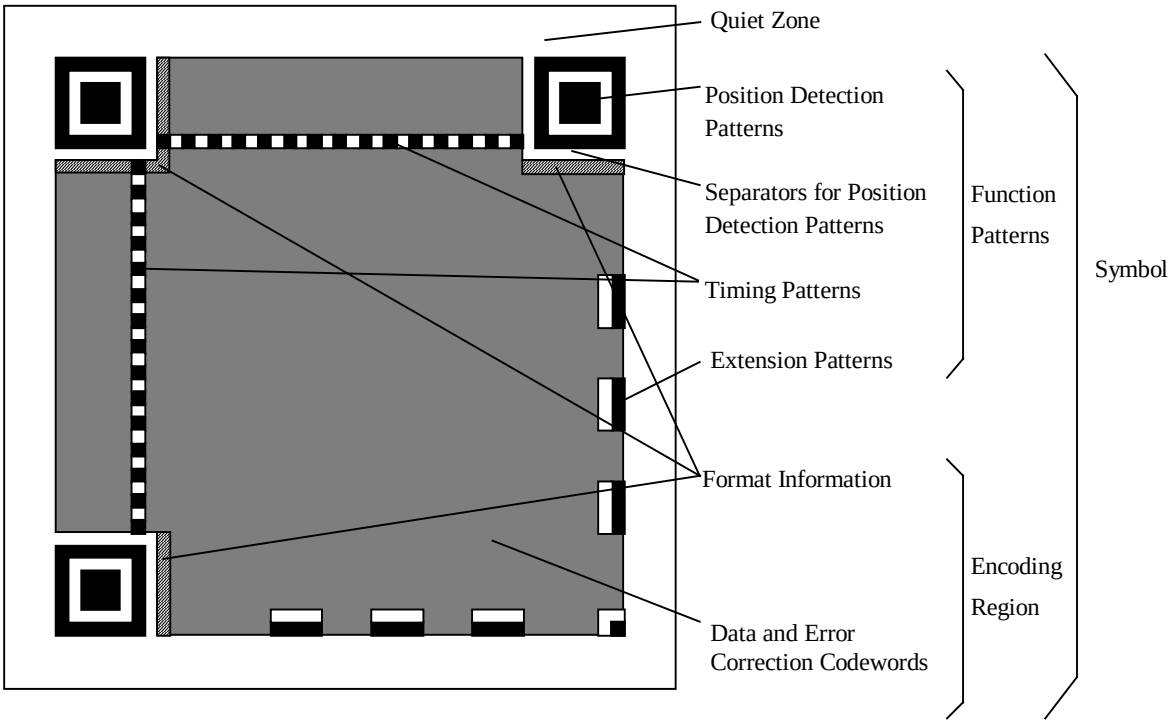


Figure M.1 — Structure of a QR Code Model 1 symbol

M.2 Symbol versions and sizes

There are only fourteen sizes of Model 1 symbol, from Version 1 to Version 14, the sizes of which are identical with those of Model 2 symbols with the same Version numbers, as defined in 7.3.1. Version 1 symbols therefore measure 21 x 21 modules, and Version 14 symbols 73 x 73 modules. Figures M.2 and M.3 illustrate Model 1 symbols in Version 1 and 2 and 13 and 14. Table M.1 shows the data capacity of all Model 1 symbols at the different error correction levels.

Table M.1 — Data capacity of all versions of Model 1 QR Code

Version	No. of Modules/ side (A)	Function Patterns Modules (B)	Format Information Modules (C)	Data Modules except (C) ($D=A^2-B-C$)	Data Capacity [codewords] [a] (E)
1	21	206	31	204	26
2	25	230	31	364	46
3	29	238	31	572	72
4	33	262	31	796	100
5	37	270	31	1 068	134
6	41	294	31	1 356	170
7	45	302	31	1 692	212
8	49	326	31	2 044	256
9	53	334	31	2 444	306
10	57	358	31	2 860	358
11	61	366	31	3 324	416
12	65	390	31	3 804	476
13	69	398	31	4 332	542
14	73	422	31	4 876	610

[a] The first codeword shall be 4 bits in length. All subsequent codewords shall be 8 bits in length. The first, 4 bit, data codeword shall be prefixed with 0000 to make its length 8 bits for generating the error correction codewords.

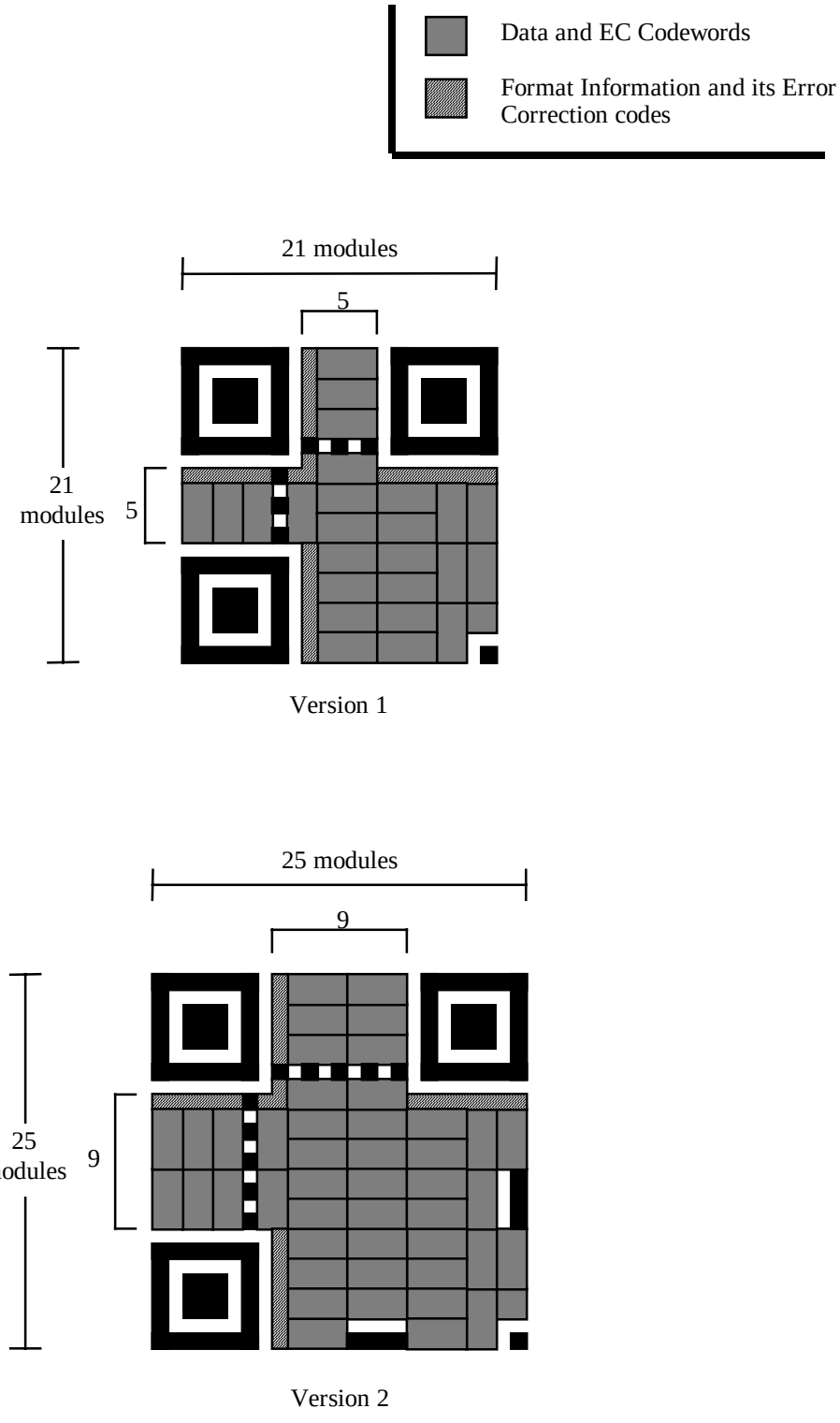
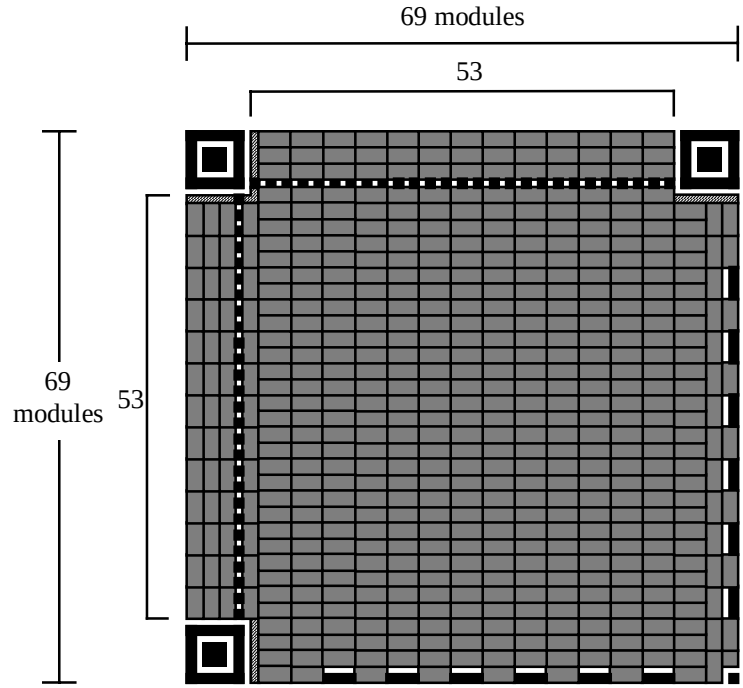
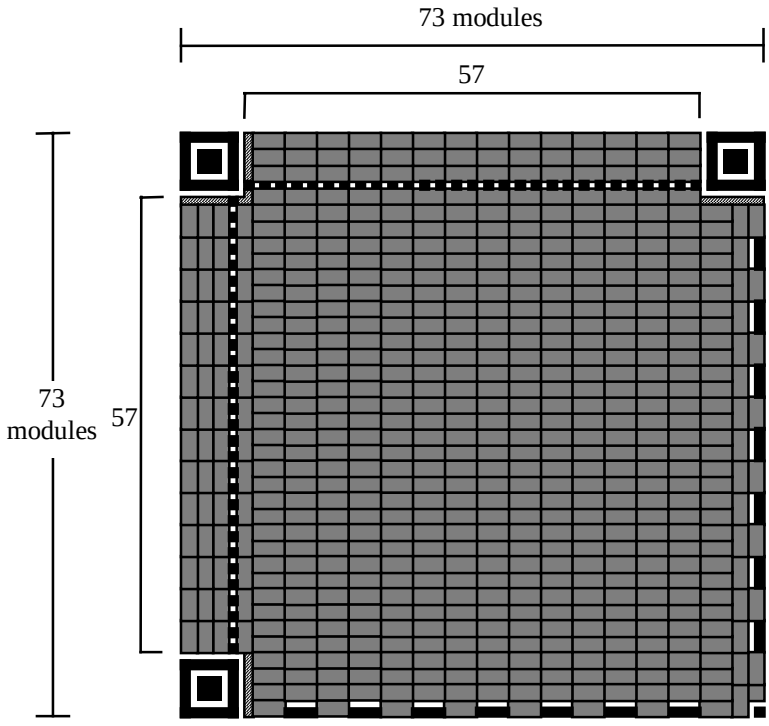


Figure M.2 — Model 1 symbols, versions 1 and 2



Version 13



Version 14

Figure M.3 — Model 1 symbols, versions 13 and 14

M.3 Symbol structure

M.3.1 Alignment patterns

Model 1 QR Code symbols have no alignment patterns. The encoding region covers the entire area shown shaded in Figure M.1 together with the Format Information.

M.3.2 Extension Patterns

These patterns were originally intended for future extension of QR Code functions and do not encode data. Extension Patterns shall consist of one four module square block located at the lower right corner of the symbol together with a number of eight module blocks located along the outer right and bottom edges of the symbol. The number of eight module blocks depends on the symbol version and may be calculated for version N from the formula

no. of eight module extension blocks = 2(N DIV 2).

This means that Version 1 symbols only have the four module Extension Pattern; Version 2 and 3 symbols have in addition 2 eight module blocks, Version 4 and 5 symbols have 4, and so on. Figures M.2 and M.3 illustrate the positioning of the Extension Patterns for Versions 1, 2, 13 and 14.

Figure M.4 below illustrates the dark and light module patterns for the Extension Patterns at the bottom right corner, right side and bottom of the symbol respectively.

For odd-numbered symbol versions, the first eight module blocks shall be positioned at the right hand end of rows 17 to 20 or at the bottom of columns 17 to 20. Subsequent blocks shall be positioned at the end (bottom) of rows (columns) 25 to 28, 33 to 36 and so on, leaving an eight module block as part of the encoding region between Extension Patterns.

The same principles shall apply to even-numbered versions, commencing in rows (columns) 13 to 16, then 21 to 24, 29 to 32 and so on.

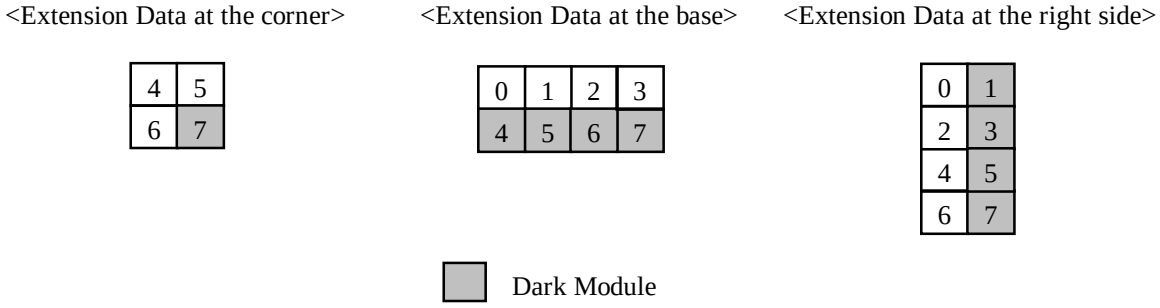


Figure M.4 — Extension patterns

NOTE Some early implementations of QR Code used Extension Patterns differing from those shown, with bits 0 and 3 (at the base) and bits 0 and 6 (at the right side) dark, in addition to the bits colored dark in Figure M.4. Both patterns are valid and the use of either pattern conveys no information in the symbol.

M.3.3 Version Information

There is no Version Information field in the encoding region of the symbol.

M.4 Encodation of Model 1 symbols

The encodation procedure is as defined in 8.1. The reference to Extended Channel Interpretations in Step 1 shall be ignored. Step 5 shall be replaced by the following:

Group the blocks of data codewords and error correction codewords in sequence and add any remainder codewords necessary to fill the symbol capacity.

In Step 6, replace ‘Alignment patterns’ by ‘Extension Patterns’.

M.4.1 Conversion of data to bit stream

Conversion of data to a bit stream in Model 1 follows the procedure defined for Model 2 as described in 8.4 (excluding sub-clause 8.4.1 and its sub-clauses). However, in the data analysis stage of the process it should be noted that Model 1 symbols do not support the ECI protocol and the data must therefore be verified to ensure that it contains only numeric, alphanumeric, 8-bit byte (JIS-8) data or Kanji data as specified in 8.3.2 to 8.3.5.

M.4.2 Bit stream to codeword conversion

In Model 1 symbols, the bit stream shall be divided into a sequence of codewords, commencing with one 4 bit codeword, and the remaining codewords shall all be 8 bits in length. The final codeword shall if necessary be padded to 8 bits in length by the adddition of pad bits with binary value 0 after the least significant data bit. Pad codewords **11101100** and **00010001** shall be added as described in 8.4.9 to fill the data codeword capacity of the symbol as shown in Tables M.2 and M.3.

Table M.2 — Number of symbol characters and input data capacity for Model 1, versions 1 to 8

Version	Error correction level	Number of data codewords[a]	Number of data bits[b]	Data capacity			
				Numeric	Alphanumeric	8-bit Byte	Kanji
1	L	19	148	40	24	17	10
	M	16	124	33	20	14	8
	Q	13	100	25	15	11	6
	H	9	68	16	10	7	4
2	L	36	284	81	49	34	20
	M	30	236	66	40	28	17
	Q	24	188	52	31	22	13
	H	16	124	33	20	14	8
3	L	57	452	131	79	55	33
	M	44	348	100	60	42	25
	Q	36	284	81	49	34	20
	H	24	188	52	31	22	13
4	L	80	636	186	113	78	48
	M	60	476	138	84	58	35
	Q	50	396	114	69	48	29
	H	34	268	76	46	32	19
5	L	108	860	253	154	106	65
	M	82	652	191	116	80	49
	Q	68	540	157	95	66	40
	H	46	364	105	63	44	27
6	L	136	1 084	321	194	134	82
	M	106	844	249	151	104	64
	Q	86	684	201	122	84	51
	H	58	460	133	81	56	34
7	L	170	1 356	402	244	168	103
	M	132	1 052	311	188	130	80
	Q	108	860	253	154	106	65
	H	72	572	167	101	70	43
8	L	208	1 660	493	299	206	126
	M	160	1 276	378	229	158	97
	Q	128	1 020	301	183	126	77
	H	87	692	203	123	85	52

[a] The first codeword shall be 4 bits in length. All subsequent codewords shall be 8 bits in length.

[b] The number of Data Bits includes bits for Mode Indicator and character count Indicator.

Table M.3 — Number of symbol characters and input data capacity for Model 1, versions 9 to 14

Version	Error correction Level	Number of data codewords[a]	Number of data bits[b]	Data capacity			
				Numeric	Alphanumeric	8-bit Byte	Kanji
9	L	246	1 964	585	354	244	150
	M	186	1 484	441	267	184	113
	Q	156	1 244	369	223	154	94
	H	102	812	239	145	100	61
10	L	290	2 316	690	418	287	177
	M	222	1 772	526	319	219	135
	Q	183	1 460	433	262	180	111
	H	124	988	291	176	121	74
11	L	336	2 684	800	485	333	205
	M	256	2 044	608	368	253	156
	Q	208	1 660	493	299	205	126
	H	145	1 156	342	207	142	87
12	L	384	3 068	915	555	381	234
	M	292	2 332	694	421	289	178
	Q	244	1 948	579	351	241	148
	H	165	1 316	390	236	162	100
13	L	432	3 452	1 030	624	429	264
	M	332	2 652	790	479	329	202
	Q	276	2 204	656	398	273	168
	H	192	1 532	454	275	189	116
14	L	489	3 908	1 167	707	486	299
	M	368	2 940	877	531	365	225
	Q	310	2 476	738	447	307	189
	H	210	1 676	498	302	207	127

[a] The first codeword shall be 4 bits in length. All subsequent codewords shall be 8 bits in length.

[b] The number of Data Bits includes bits for Mode Indicator and character count Indicator.

M.5 Error correction coding

The error correction coding procedure and error correction levels are as defined in 8.5.

In Model 1, since the first data codeword consists of only 4 bits, it shall be prefixed with four zero bits and treated as an 8 bit codeword for the error correction calculations. Tables M.4 and M.5 list, for each version and Error Correction Level, the total number of codewords including the number of Remainder Codewords, the total number of error correction codewords, and the structure and number of error correction blocks for Model 1 symbols.

A Remainder Codeword is a Pad codeword added after the end of the final block of error correction codewords to fill the capacity of the symbol. The Remainder Codewords serve no other purpose. For example, in a Version 14-H symbol, there are 6 blocks of 101 data and error correction codewords, totalling 606 codewords; since the symbol contains 610 codewords, 4 Remainder Codewords are added at the end. The Pad Codewords **11101100** and **00010001** shall be used alternately as Remainder Codewords.

Divide the codeword sequence into the required number of blocks (as defined in Tables M.4 and M.5 for Model 1) to enable the error correction algorithms to be processed. Generate the error correction codewords for each block.

Table M.4 — Error correction characteristics for Model 1, versions 1 to 7

Version	Total number of codewords	Error Correction Level	Number of error correction codewords	Number of error correction blocks	Error correction code per block [a]	Number of Remainder Codewords
1	26	L	7	1	(26,19,2) [b]	0
		M	10	1	(26,16,4) [b]	0
		Q	13	1	(26,13,6) [b]	0
		H	17	1	(26,9,8) [b]	0
2	46	L	10	1	(46,36,4) [b]	0
		M	16	1	(46,30,8)	0
		Q	22	1	(46,24,11)	0
		H	30	1	(46,16,15)	0
3	72	L	15	1	(72,57,7) [b]	0
		M	28	1	(72,44,14)	0
		Q	36	1	(72,36,18)	0
		H	48	1	(72,24,24)	0
4	100	L	20	1	(100,80,10)	0
		M	40	1	(100,60,20)	0
		Q	50	1	(100,50,25)	0
		H	66	1	(100,34,33)	0
5	134	L	26	1	(134,108,13)	0
		M	52	1	(134,82,26)	0
		Q	66	1	(134,68,33)	0
		H	88	2	(67,23,22)	0
6	170	L	34	1	(170,136,17)	0
		M	64	2	(85,53,16)	0
		Q	84	2	(85,43,21)	0
		H	112	2	(85,29,28)	0
7	212	L	42	1	(212,170,21)	0
		M	80	2	(106,66,20)	0
		Q	104	2	(106,54,26)	0
		H	138	3	(70,24,23)	2

[a] (c,k,r): c = total number of codewords
k = number of data codewords
r = number of error correction capacity

[b] Error correction capacity is less than half number of error correction codewords to reduce probability of misdecodes.

Table M.5 — Error correction characteristics for Model 1, versions 8 to 14

Version	Total number of codewords	Error correction level	Number of error correction codewords	Number of error correction blocks	Error correction code per block [a]	Number of Remainder Codewords
8	256	L	48	2	(128,104,12)	0
		M	96	2	(128,80,24)	0
		Q	128	2	(128,64,32)	0
		H	168	3	(85,29,28)	1
9	306	L	60	2	(153,123,15)	0
		M	120	2	(153,93,30)	0
		Q	150	3	(102,52,25)	0
		H	204	3	(102,34,34)	0
10	358	L	68	2	(179,145,17)	0
		M	136	2	(179,111,34)	0
		Q	174	3	(119,61,29)	1
		H	232	4	(89,31,29)	2
11	416	L	80	2	(208,168,20)	0
		M	160	4	(104,64,20)	0
		Q	208	4	(104,52,26)	0
		H	270	5	(83,29,27)	1
12	476	L	92	2	(238,192,23)	0
		M	184	4	(119,73,23)	0
		Q	232	4	(119,61,29)	0
		H	310	5	(95,33,31)	1
13	542	L	108	3	(180,144,18)	2
		M	208	4	(135,83,26)	2
		Q	264	4	(135,69,33)	2
		H	348	6	(90,32,29)	2
14	610	L	120	3	(203,163,20)	1
		M	240	4	(152,92,30)	2
		Q	300	5	(122,62,30)	0
		H	396	6	(101,35,33)	4

[a] (c,k,r): c = total number of codewords
k = number of data codewords
r = number of error correction capacity

M.6 Constructing the final message codeword sequence

The total number of codewords in the message shall always be equal to the total number of codewords capable of being represented in the symbol, as shown in Tables M.4 and M.5.

For Model 1 symbols, assemble the final sequence of *n* blocks of data and *n* blocks of error-correction codewords, adding the number of Remainder codewords defined in Table M.4 or M.5:

Data block 1, data block 2, ... data block *n*, error correction block 1, error correction block 2, ... error correction block *n*, Remainder codewords.

Example: Model 1 Symbol Version 10-H

Total capacity: 358 codewords

Data codewords: 124 (4 blocks of 31)

Error correction codewords: 232 (58 per block)

Remainder codewords required: 2

Error correction codewords 1 to 58 are calculated for data codewords 1 to 31, error correction codewords 59 to 116 for data codewords 32 to 62 and similarly for the remaining blocks, error correction codewords 117 to 174 are calculated for data codewords 63 to 93, error correction codewords 175 to 232 are calculated for data codewords 94 to 124.

The final message codeword sequence is therefore:

Data codeword 1, 2, ... 31, 32, ... 62, 63, ... 93, 94, ... 124, error correction codeword 1, 2, ... 58, 59, ... 116, 117, ... 174, 175, ... 232, remainder codeword 1, 2.

M.7 Codeword placement in matrix

M.7.1 Symbol character representation

There are two types of symbol character in the Model 1 QR Code symbol. The first codeword, consisting of four bits, shall be represented by a symbol character in the form of a 2 x 2 block of modules. All other codewords shall be represented in a 2 x 4 module block in the symbol. There are two ways of positioning these blocks, in a vertical arrangement (2 modules wide and 4 modules high) and in a horizontal arrangement (4 modules wide and 2 modules high). Figure M.5 below shows the arrangement of the modules in one symbol character for each arrangement. In the figure, "0" corresponds to the least significant bit and "7" to the most significant bit. The least significant bit shall always be positioned in the top left module of the symbol character and successive bits from left to right and top to bottom, ending with the most significant bit in the lower right module. “0” bits shall be represented by light modules and “1” bits by dark modules.

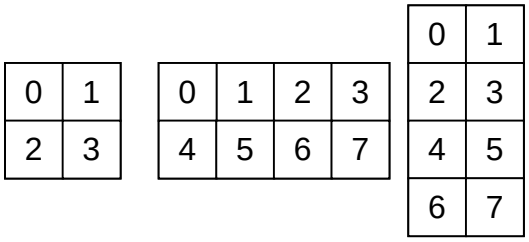


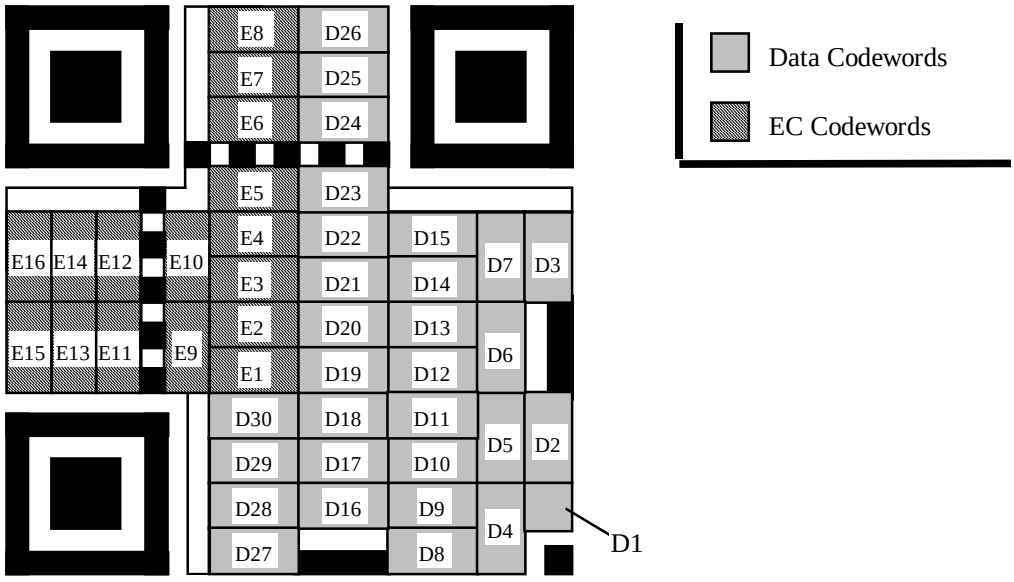
Figure M.5 — Module arrangement in one Model 1 symbol character

M.7.2 Function pattern placement

A square blank matrix shall be constructed with the number of modules horizontally and vertically corresponding to the Version in use. Positions corresponding to the Finder Pattern, Separator, Timing Pattern and Extension Pattern shall be filled with either dark modules or light modules as appropriate. Module positions for the Format Information shall be left temporarily blank. These positions are shown in Figure N.6 and N.7 and are common to all Versions.

M.7.3 Symbol character placement

In the encoding region of the Model 1 symbol, symbol characters are positioned from the bottom to the top and the right to the left starting from the right bottom corner of the symbol. The positions of symbol characters representing data codewords (indicated by D1, D2 ...) and symbol characters representing error correction codewords (indicated by E1, E2, ...) in Version 2-M and 5-H symbols are shown as examples in Figures M.6 and M.7 respectively. The first two columns of symbol characters, starting at the right, and the last four columns shall contain symbol characters in the vertical arrangement (with the exception of the first, 4 module, symbol character). All other symbol characters shall be in the horizontal arrangement.



NOTE The first data codeword D1 consists of only 4 bits.

Figure M.6 — Symbol character arrangement in Model 1 version 2-M symbol

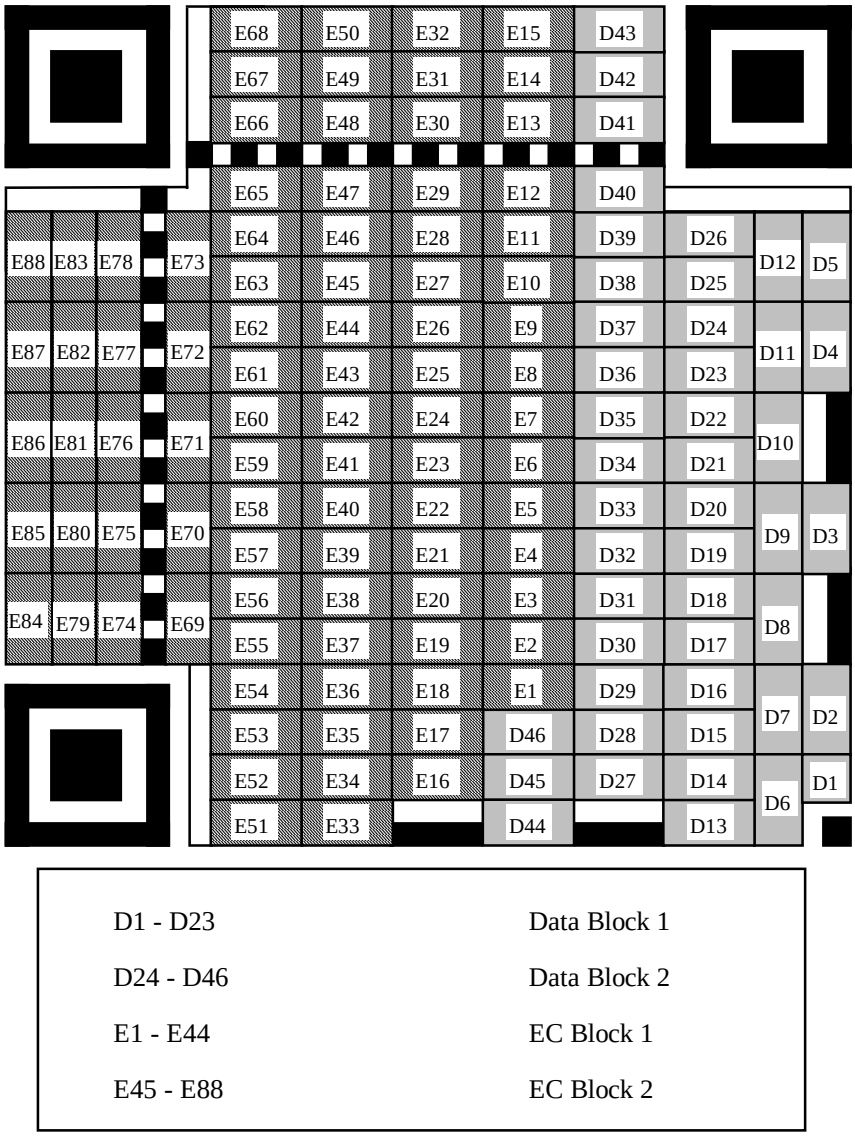
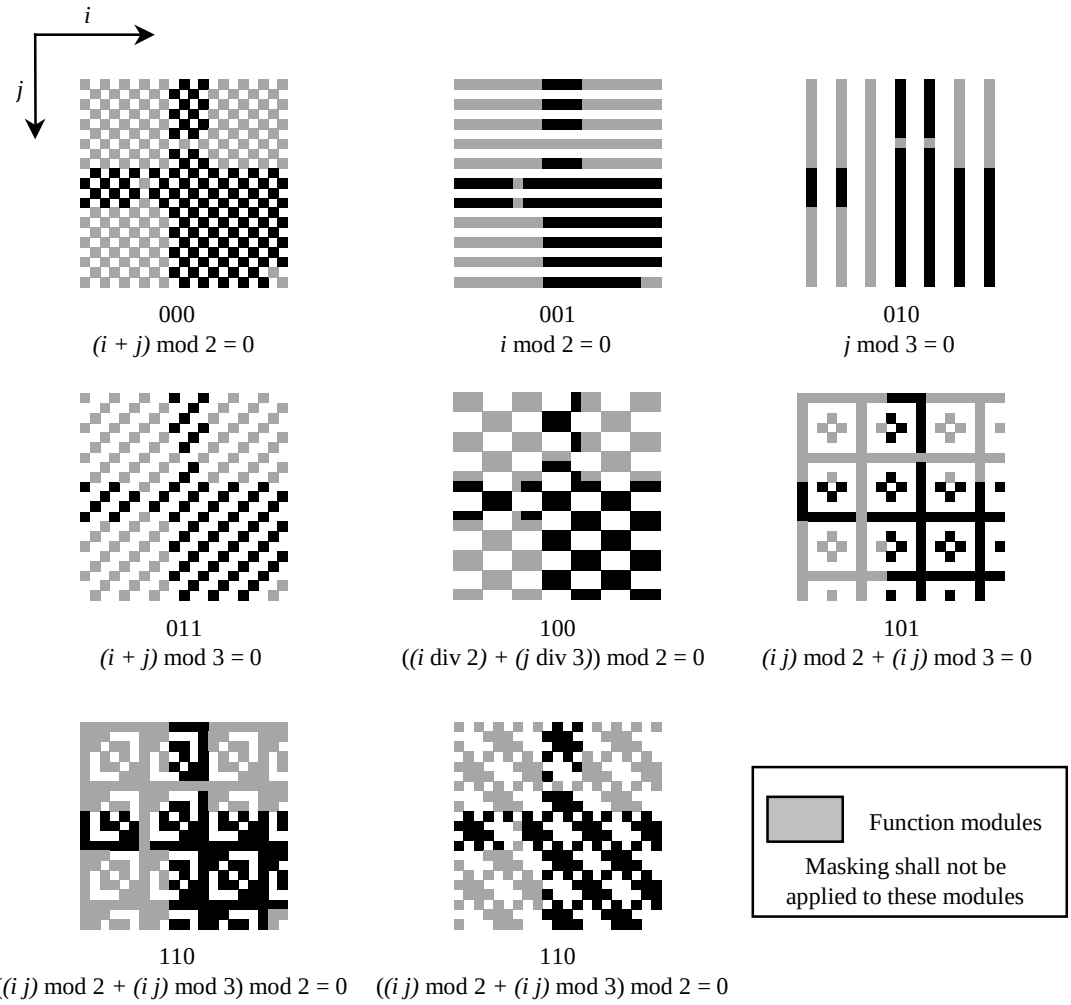


Figure M.7 — Symbol character arrangement in Model 1, version 5-H symbol

M.8 Masking

Masking shall be performed as defined in 8.8. Figures M.8 and M.9 show all Mask Patterns in a Model 1, Version 1 symbol and its masking simulation respectively.



NOTE 1 The three bits below each pattern represent the Mask Pattern Information.

NOTE 2 The equation below the Mask Pattern Information shows the mask pattern generation condition; modules which meet the condition are shown dark.

Figure M.8 — Mask patterns for Model 1, Version 1 symbol

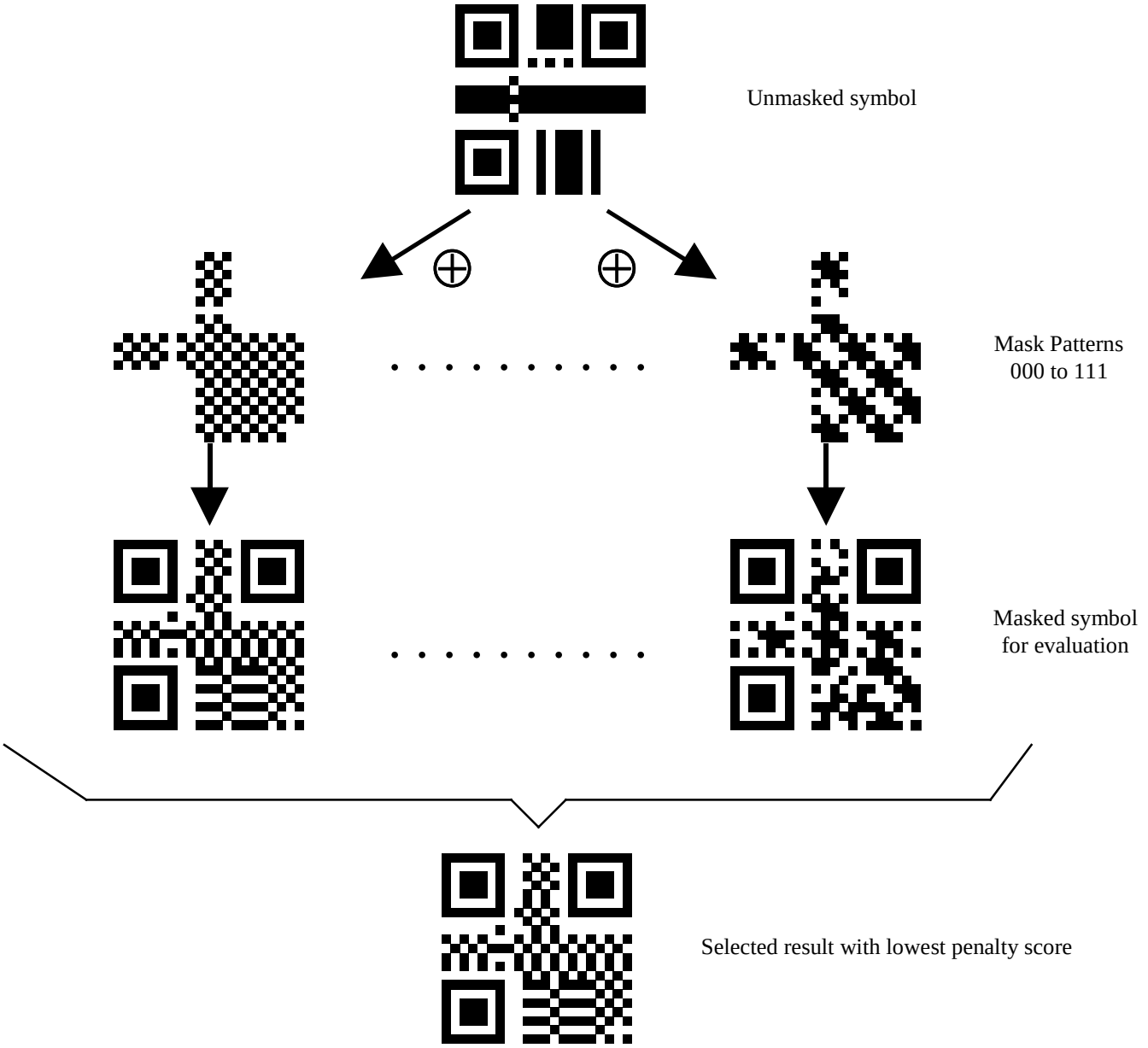


Figure M.9 — Masking simulation in Model 1

M.9 Format Information

The Format Information bits shall be calculated and, after masking, shall be mapped into the areas reserved for them in the symbol, as defined in 8.9

The mask pattern for the Format information in Model 1 symbols is **010100000100101**. The use of two different masking patterns ensures that the symbol Model may be autodiscriminated at the time of reading.

M.10 Structured Append

Structured Append for Model 1 follows precisely the same rules as for Model 2, as defined in 9. In Model 1 symbols, the Structured Append Mode Indicator **0011** is placed in the first (four module) symbol character position. This is immediately followed by two structured append codewords. In Model 1 these two codewords occupy the second and third symbol characters.

M.11 Symbol printing and marking

Clause 10 applies in its entirety.

M.12 Symbol quality

Clause 11 applies in its entirety.

M.13 Decoding procedure overview

Clause 12 applies in its entirety.

M.14 Reference decode algorithm

The reference decode algorithm described in 13 shall be applied, substituting the following two steps for steps 7 and 8 in the algorithm in 13:

7. Redefine X as the average spacing of the center points of the dark and light modules in the Timing Patterns. In a similar manner, calculate the Y dimension as the average spacing of the center points of the dark and light modules in the left side Timing Pattern.

8. Establish a sampling grid based on (a) the horizontal line through the upper Timing Pattern with lines parallel to it at the vertical spacing of Y, comprising six lines above the horizontal reference line and as many lines below it as are required for the version of the symbol and (b) the vertical line passing through the left side Timing Pattern with lines parallel to it at the horizontal spacing of X, comprising six lines to the left of the vertical reference line and as many lines to the right of it as are required for the version of the symbol.

Additionally, step 12 shall be amended by substituting a reference to this Annex for the reference to 8.7.3.

Step 13 shall not be applied.

M.15 Autodiscrimination capability

Clause 14 applies in its entirety

M.16 Transmitted data

Clause 15 applies, with the exception of sub-clauses 15.2 and 15.3, ignoring other references to Extended Channel Interpretations in the text.

M.17 Annexes

Annexes A, B, F, H, I, J, K, and L apply in their entirety.

Annexes D and E do not apply to Model 1 symbols.

Annex C applies, substituting the following for the final sentence in the first paragraph of C.1:

> Finally, masking shall be applied by XORing the bit string with **010100000100101** to ensure that the format information bit pattern is not all zeroes for any combination of Mask Pattern and Error Correction Level and to enable Model 1 symbols to be autodiscriminated from Model 2 symbols.

The last two lines of the example in Annex C.1 should be amended as follows, by substituting:

XOR with mask **010100000100101**

Result: **011110011111001**

Annex G applies, substituting the following step 3:

3. The Position Detection Patterns, Timing Patterns and Extension Pattern are placed in a blank 21 × 21 matrix and the module positions for the Format Information are left temporarily blank. The codewords from Step 2 are placed in the matrix in accordance with the placement rules in this Annex.

ICS 01.080.50; 35.040

Price based on 114 pages