INTERNATIONAL STANDARD

ISO/IEC 23941

First edition
2022-05

Information technology — Automatic identification and data capture techniques — Rectangular Micro QR Code (rMQR) bar code symbology specification

Technologies de l'information — Techniques d'identification automatique et de capture des données — Spécification de la symbologie de code à barres Rectangular Micro QR Code (rMQR)

Reference number
ISO/IEC 23941:2022(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published 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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

Rectangular Micro QR Code (rMQR) is a matrix symbology. The symbol consists of an array of nominally square modules, arranged in a rectangular pattern. Included is a unique finder pattern located at a single corner which is intended to assist in easy location of the symbols position, size, and inclination. A wide range of sizes of symbol is provided for, together with two levels of error correction. Module dimensions are user-specified to enable symbol production by a wide variety of techniques.

Information technology — Automatic identification and data capture techniques — Rectangular Micro QR Code (rMQR) bar code symbology specification

1 Scope

This document defines the requirements for the symbology known as rMQR. It specifies the rMQR symbology characteristics, data character encoding methods, symbol formats, dimensional characteristics, error correction rules, reference decoding algorithm, printing quality requirements and user-selectable application parameters.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 19762, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary*

ISO/IEC 8859-1, *Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1*

ISO/IEC 15415, *Information technology — Automatic identification and data capture techniques — Bar code symbol print quality test specification — Two-dimensional symbols*

3 Terms and definitions

For the purpose of this document, the terms and definitions given in ISO/IEC 19762 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
character count indicator
bit sequence which defines the data string length in a *mode* (3.6)

3.2
encoding region
region of the symbol not occupied by *function patterns* (3.4) and available for encoding of data and error correction codewords, and for *format information* (3.3)

3.3
format information
encoded pattern containing information on the error correction level and *version* (3.15) applied to symbol characteristics

3.4
function pattern
overhead component of the symbol [finder pattern, *separator* (3.12), *timing patterns* (3.14), alignment patterns, finder sub patterns and corner finder pattern] required for location of the symbol or identification of its characteristics to assist in decoding

3.5
masking
process of XORing the bit pattern in an area of the symbol with a mask pattern to equalize the number of light and dark modules

3.6
mode
method of representing a defined character set as a bit string

3.7
mode indicator
identifier indicating in which *mode* (3.6) the following data sequence is encoded

3.8
padding bit
zero bit, not representing data, used to fill empty positions of the final codeword during the encoding process

3.9
remainder bit
zero bit, not representing data, used to fill empty positions of the symbol *encoding region* (3.2) after the final symbol character, where the area of the *encoding region* (3.2) does not divide exactly into 8-bit symbol characters

3.10
remainder codeword
codeword, placed after the data codeword stream that was generated in data encoding process, used to fill empty codeword positions to meet the requirements of number of data codeword of the *version* (3.15) and error correction definitions

3.11
segment
sequence of data encoded according to the rules of one ECI or encoding mode

3.12
separator
function pattern (3.4) of all light modules, one module wide, used to separate the finder pattern from the rest of the symbol

3.13
terminator
bit pattern of defined number (depending on symbol) of all zero bits used to end the bit string representing data

3.14
timing pattern
alternating sequence of dark and light modules enabling module coordinates in the symbol to be determined

3.15
version
size of the symbol represented in terms of the number of modules in the vertical and horizontal axes, indicated, for example, as R7x59

Note 1 to entry: The error correction level applied to the symbol can be suffixed to the Version Indicator, e.g., Version R11x27-M.

3.16
version Indicator
five-bit identifier indicating symbol version used for a part of the *format information* (3.3)

3.17
error correction level indicator
one-bit identifier indicating error correction level used for a part of the *format information* (3.3)

4 Mathematical and logical symbols, abbreviated terms and conventions

4.1 Mathematical and logical symbols

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 ⊕.

4.2 Abbreviated terms

BCH Bose-Chaudhuri-Hocquenghem

ECI Extended Channel Interpretation

RS Reed-Solomon

4.3 Conventions

4.3.1 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.

4.3.2 Byte notation

Byte contents are shown as hexadecimal (hex) values.

4.3.3 Version references

Symbol versions are referred to in the form Version $RC_V xC_H$-E where C_V identifies the vertical number of modules (7, 9, 11, 13, 15, 17), C_H identifies the horizontal number of modules (27, 43, 59, 77, 99, 139), and E indicates the error correction level (M and H). For example, R13x27-M indicates a rectangular symbol that has 13 vertical modules, 27 horizontal modules, and an error correction level M. Versions may be referred to without error correction level. For example, R13x27.

NOTE For M and H, see 6.1 e).

5 Conformance

rMQR symbols (and equipment designed to produce or read rMQR symbols) shall be considered as conforming with this document if they provide or support the features defined in this document.

6 rMQR specifications

6.1 Basic characteristics

rMQR 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, as shown in Table 5.);

   3) byte data [default shall be the character set defined in Annex G; or other sets as otherwise defined (see 7.3.5)];

   4) Kanji characters (Characters can be compacted into 13 bits (see 7.3.6).

b) Representation of data:

   A dark module is nominally a binary one and a light module is nominally a binary zero. See 6.2 for details of reflectance reversal.

c) Symbol size (not including quiet zone):

   See Table 1 for the symbol sizes for 7 x 43 modules to 17 x 139 modules (Version R7x43 to R17x139).

d) Data characters per symbol:

   The maximum symbol size of Version R17x139-M is as specified below.

   — numeric data: 361 characters

   — alphanumeric data: 219 characters

   — Byte data: 150 characters

   — Kanji data: 92 characters

e) Selectable error correction:

   This symbology supports two levels of Reed-Solomon error correction, M and H, which allows the recovery of rMQR codewords up to the indicated rate below.

   — M 15 %

   — H 30 %

f) Code type:

   Matrix

g) Orientation independence:

   Yes (both rotation and reflection)

Figure 1 illustrates a Version R13x27 rMQR symbol in normal colour and with reflectance reversal (see 6.2), in both normal and mirror image orientations.

6.2 Summary of additional features

The use of the following additional features is optional in rMQR.

— Extended channel interpretations (ECI)

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. See 7.3.2.

— Reflectance reversal

Symbols are intended to be read when marked so that the image is either dark on light or light on dark (see Figure 1). The specifications in this document are based on dark images on a light background. In the case of symbols produced with reflectance reversal, references to dark or light modules should be taken as references to light or dark modules respectively.

— Mirror imaging

The arrangement of modules defined in this document represents the "normal" orientation of the symbol. It is, however, possible to achieve a valid decode of a symbol in which the arrangement of the modules has been laterally transposed. When viewed with the rMQR finder pattern at the top left, and the finder sub pattern at the bottom right corners of the symbol, the effect of mirror imaging is to interchange the row and column positions of the modules. (See Figure 1.)

a) Normal orientation and normal reflectance arrangement

b) Normal orientation and reversed reflectances

c) Mirror image orientation and normal reflectance arrangement

d) Mirror image orientation and reversed reflectances

NOTE The corner marks in Figures 1 indicate the extent of the quiet zone.

Figure 1 — Examples of rMQR symbol encoding the text "12345678901234567890123456"

6.3 Symbol structure

6.3.1 General

Each rMQR symbol shall be constructed of nominally square modules set out in a rectangular array and shall consist of an encoding region and function pattern. The function pattern shall contain a finder pattern, separator, timing patterns, alignment patterns, finder sub patterns and corner finder pattern. See Figure 2.

Function patterns do not encode data. The symbol shall be surrounded on all four sides by a quiet zone border.

Figures 2 to 7 illustrate the structure of a Version R7x43, R9x43, R11x43, R13x43, R15x43 and R17x43 symbols, respectively.

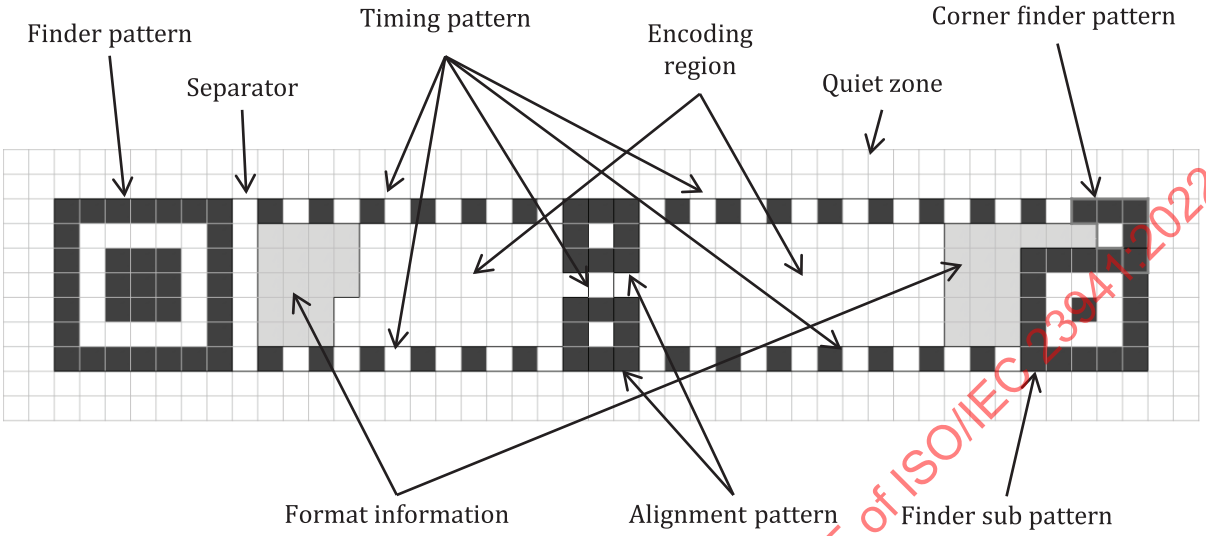


Figure 2 — Structure of Version R7x43 rMQR symbol

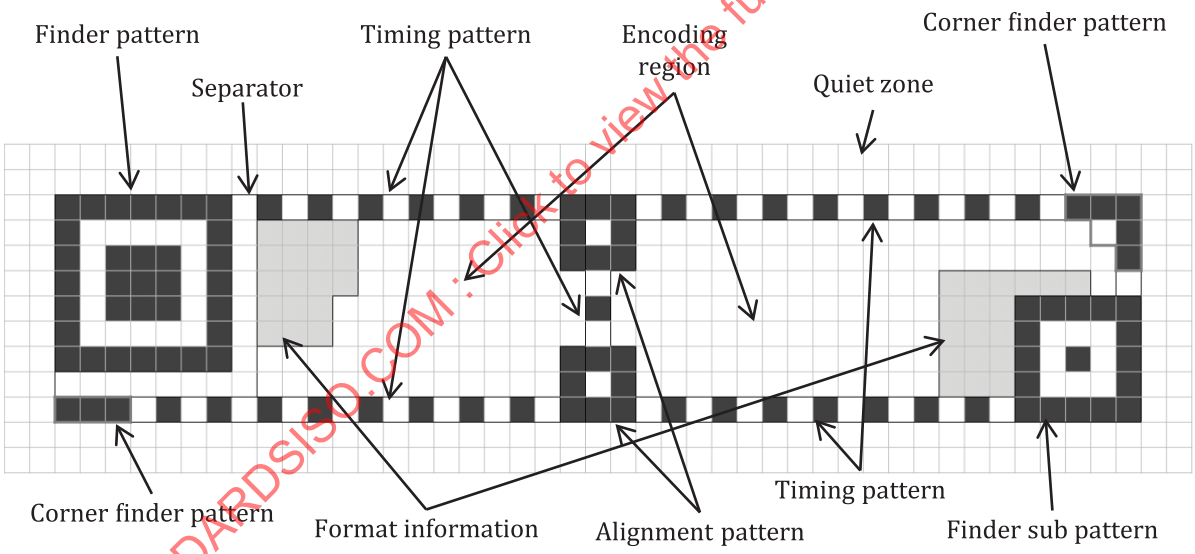


Figure 3 — Structure of Version R9x43 rMQR symbol

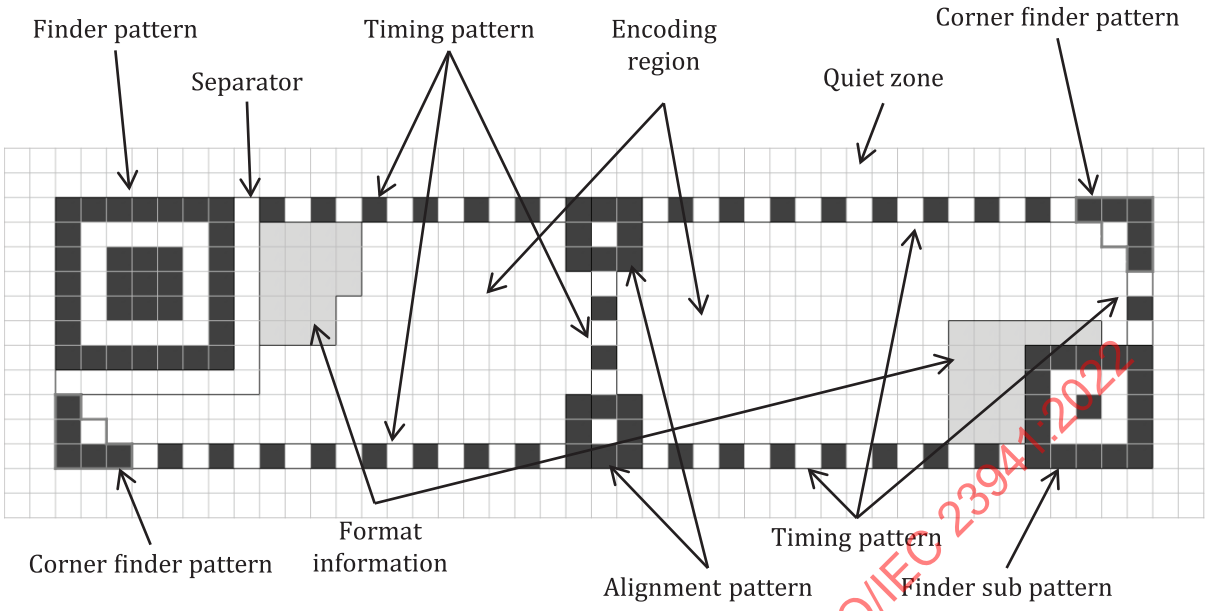


Figure 4 — Structure of Version R11x43 rMQR symbol

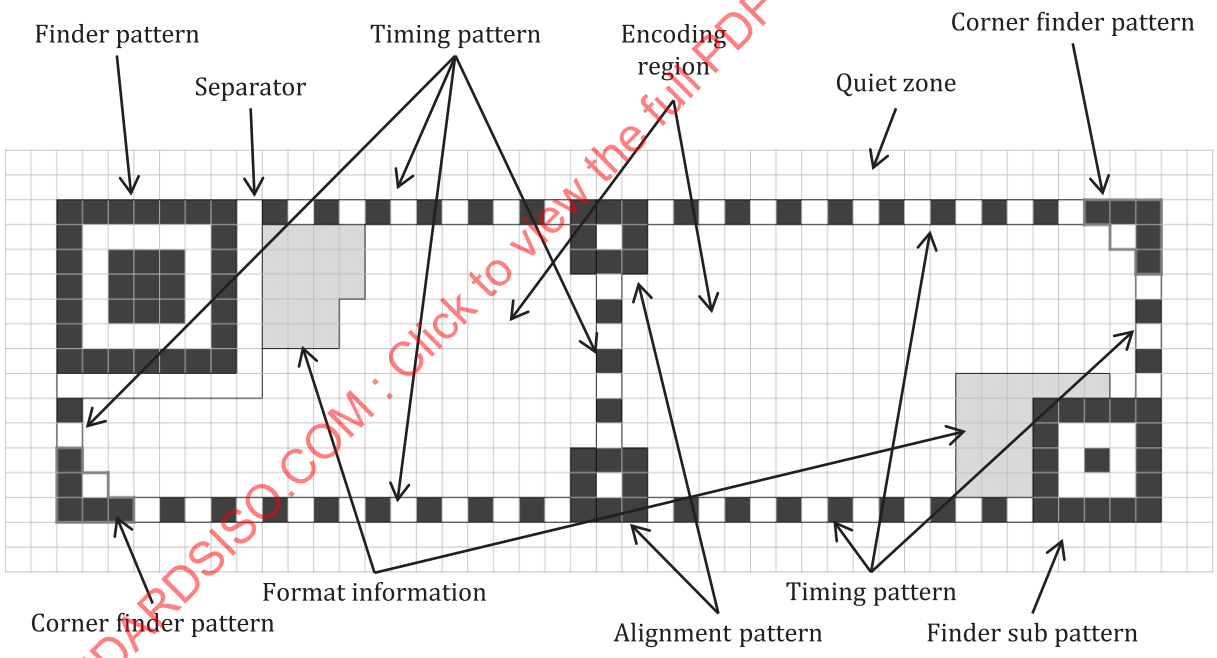


Figure 5 — Structure of Version R13x43 rMQR symbol

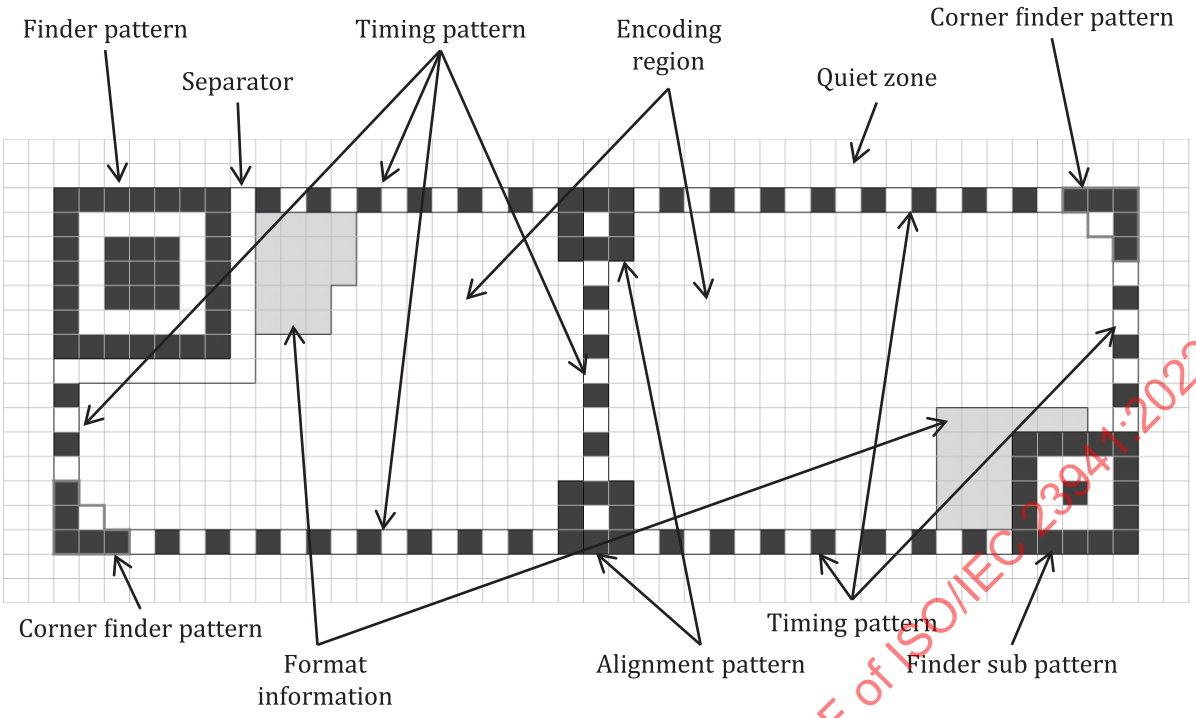


Figure 6 — Structure of Version R15x43 rMQR symbol

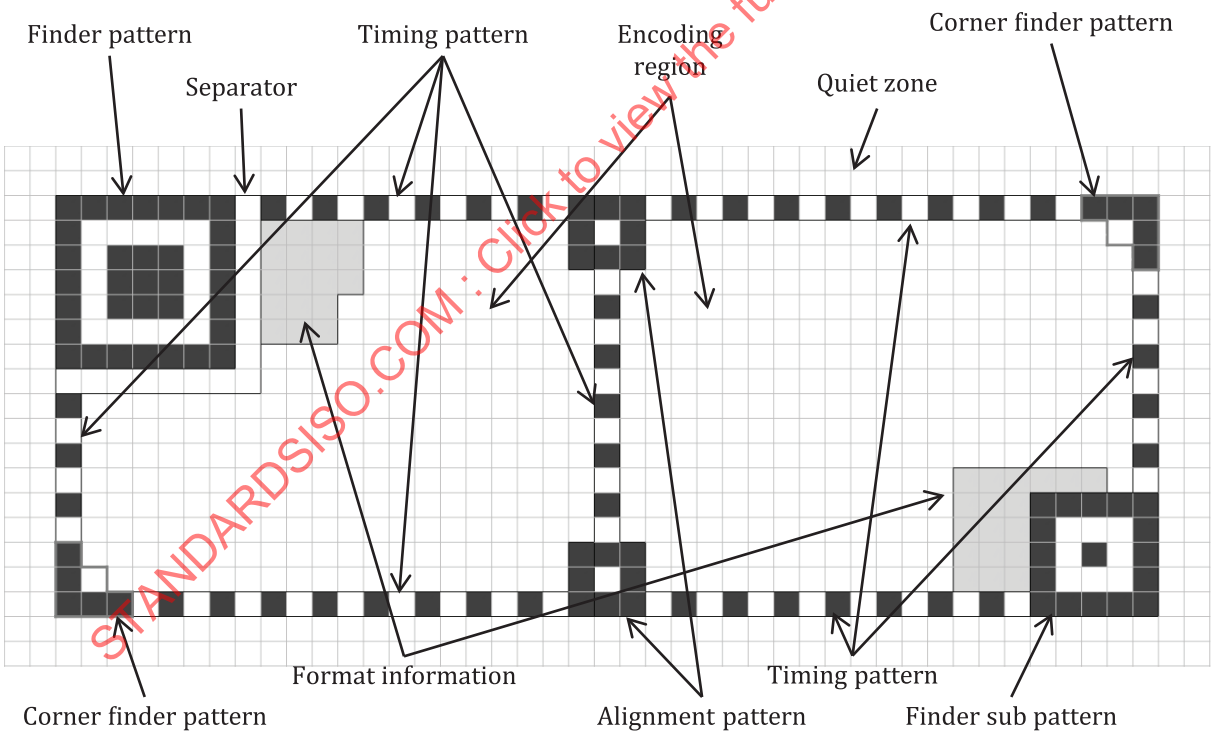


Figure 7 — Structure of Version R17x43 rMQR symbol

6.3.2 Symbol Versions and sizes

There are 32 sizes of rMQR symbol, referred to as Version R7x43 to R17x139. The vertical module has 6 sizes depending on the number of modules, e.g., 7, 9, 11, 13, 15, 17, and the horizontal module has 6 sizes depending on the number of modules, e.g., 27, 43, 59, 77, 99, 139. Table 1 shows code sizes for all versions. Figure 8 illustrates the structure of symbols with 11 vertical modules and 27 to 139

horizontal modules. Figure 9 illustrates the structure of symbols with 43 horizontal modules and 7 to 17 vertical modules.

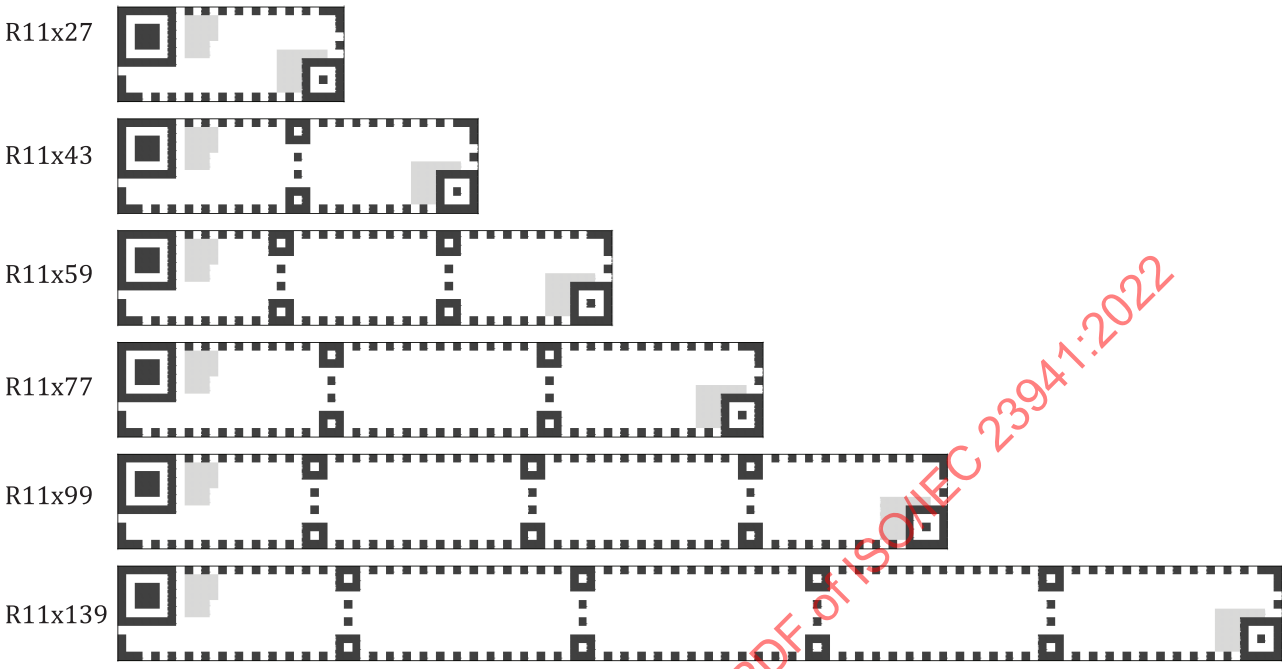


Figure 8 — rMQR symbols with 11 vertical modules and 27 to 139 horizontal modules

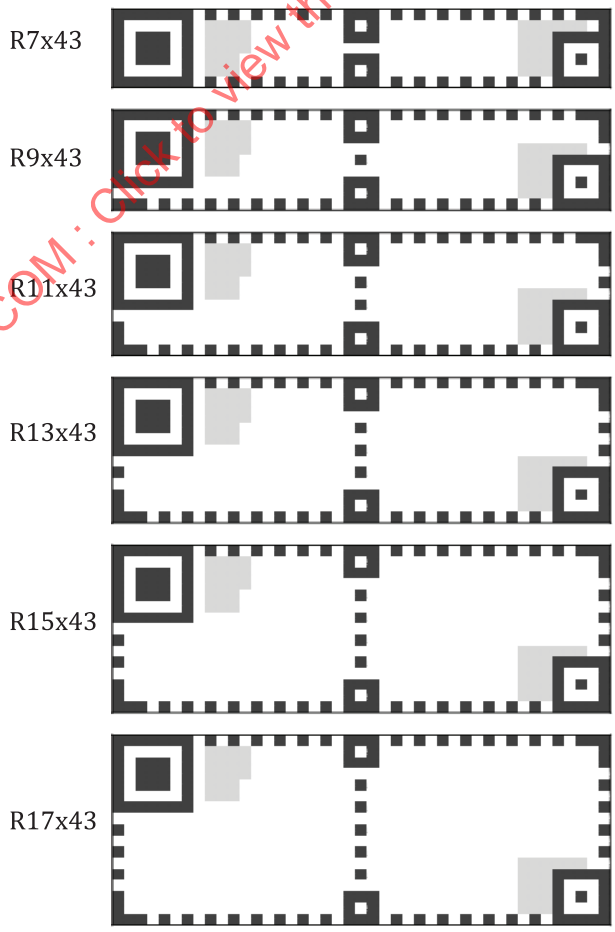
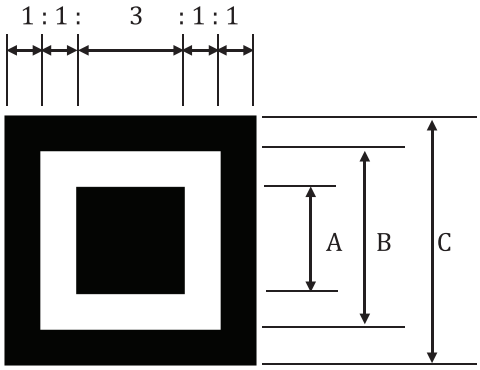


Figure 9 — rMQR symbols with 43 horizontal modules and 7 to 17 vertical modules

6.3.3 Finder pattern

A single finder pattern is located at the upper left corner of the symbol as illustrated in Figures 2 to 7. For a symbol with 7 vertical modules, it is located at the leftmost of the symbol. Each finder pattern may be viewed as three superimposed concentric squares and is constructed of 7 × 7 dark modules, 5 × 5 light modules and 3 × 3 dark modules. The ratio of module widths in each finder pattern is 1 : 1 : 3 : 1 : 1 as illustrated in Figure 10. Identification of the finder pattern together with the timing patterns unambiguously defines the size, location and rotational orientation of the symbol in the field of view.



Key

A 3 modules
B 5 modules
C 7 modules

Figure 10 — Structure of finder pattern

6.3.4 Separator

A one-module wide separator, constructed of all light modules, is placed between each finder pattern and the encoding region, as illustrated in Figures 2 to 7.

6.3.5 Timing pattern

The horizontal and vertical timing patterns respectively consist of a one-module wide row or column of alternating dark and light modules. They enable the symbol density and version to be determined and provide datum positions for determining module coordinates.

The horizontal timing pattern runs across row 0 of the symbol between a separator at the right of the finder pattern and a corner finder pattern at the right end of the symbol. It also runs across the lowermost row of the symbol between a corner finder pattern at the left end of the symbol and the finder sub pattern, excluding the position of the alignment pattern. The vertical timing pattern runs down column 0 of the symbol between a separator below the finder pattern and a corner finder pattern at the lowermost end of the symbol. It also runs down the rightmost column between a corner finder pattern at the upper end of the symbol and a finder sub pattern. The timing pattern also runs down a column of symmetrically located upper and lower alignment patterns, from the top to the bottom of the alignment patterns. Figure 11 illustrates the timing pattern area of Version R17x43 symbol. Timing patterns are areas enclosed by dotted lines in this figure. Symbols with 7 to 11 vertical modules do not have timing patterns at the column 0, nor do symbols with 7 and 9 vertical modules have timing patterns at the right end of the symbol. See Figures 2 to 7.

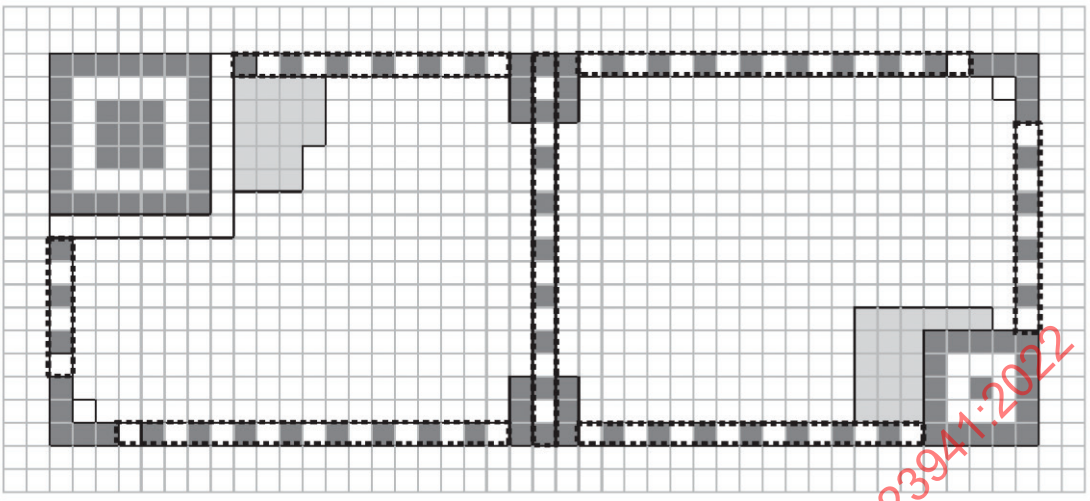
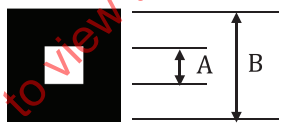


Figure 11 — Timing pattern area of Version R17x43

6.3.6 Alignment patterns

Alignment patterns are present in symbols with 43 or more horizontal modules. Each alignment pattern may be viewed as two superimposed concentric squares, constructed of 3 x 3 dark modules, and a single central light module. The structure of an alignment pattern is shown in Figure 12. The number of alignment patterns depends on the number of horizontal modules. The alignment patterns shall be placed at the position defined in Annex D, except for symbols with 27 horizontal modules.



Key

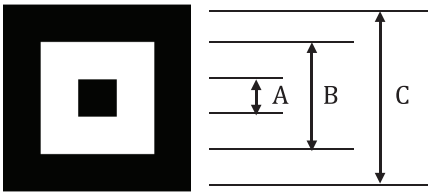
A 1 module

B 3 modules

Figure 12 — Structure of alignment pattern

6.3.7 Finder sub pattern

The finder sub pattern is placed at the right-hand bottom of the symbol, as illustrated in Figures 2 through 7. The finder sub pattern may be viewed as three superimposed concentric squares, and is constructed of 5 x 5 dark modules, 3 x 3 light modules, and a single central dark module. The structure of finder sub pattern is shown in Figure 13.



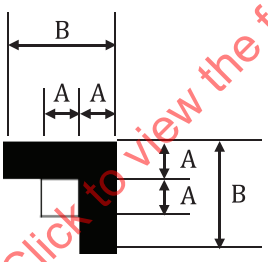
Key

A 1 module
B 3 modules
C 5 modules

Figure 13 — Structure of finder sub pattern

6.3.8 Corner finder pattern

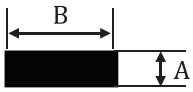
The corner finder pattern is located at the upper right and the lower left of rMQR symbol, as illustrated in Figures 2 through 7. The corner finder pattern is constructed of 3 dark modules in the horizontal direction, and 3 dark modules in the vertical direction that intersect at the upper right or lower left of the symbol, and 1 light module which is located so as to contact an angle made by the dark modules described above. See Figure 14. Symbols with 7 vertical modules do not have a lower left corner finder pattern. Symbols with 9 vertical modules have a lower left corner finder pattern of 3 dark modules in the horizontal direction, as shown in Figure 15.



Key

A 1 module
B 3 modules

Figure 14 — Structure of corner finder pattern



Key

A 1 module
B 3 modules

Figure 15 — Structure of lower left corner finder pattern of symbols with 9 vertical modules

6.3.9 Encoding region

This region shall contain the symbol characters representing data, those representing error correction codewords, and the format information. Refer to 7.7.1 for details of the symbol characters. Refer to 7.9 for details of the format information.

6.3.10 Quiet zone

The width of the quiet zone shall be 2X, surround the symbol on all four sides, and shall be free of all other markings. Its nominal reflectance value shall be equal to that of the light modules.

7 Requirements

7.1 Encode procedure overview

This clause provides an overview of the steps required to convert input data to a rMQR symbol. (For examples of encoding, see Annex I.)

Step 1 Data analysis

Analyze the input data stream to identify the variety of different characters to be encoded. The rMQR symbol format supports the extended channel interpretation feature, enabling data differing from the default character set to be encoded. rMQR includes several modes (see 7.3) 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 correction level. A list of rMQR symbol versions and their data capacities are shown in Table 1.

Step 2 Data encoding

Convert the data characters into a bit stream in accordance with the rules for the mode in force, as defined in 7.4.2 to 7.4.7, 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 remainder codewords as necessary to fill the number of data codewords (as defined in Table 6) required for the version and error correction level.

Step 3 Error correction coding

Divide the codeword sequence into the required number of blocks (as defined in Table 8) 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 7.6 (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, alignment patterns (if required), finder sub pattern and corner finder pattern. Refer to 7.7 for details of codeword placement in matrix.

Step 6 Data masking

Apply the data masking patterns to the encoding region of the symbol. Refer to 7.8 for details of data masking.

Step 7 Format information

Generate the format information and complete the symbol. Refer to 7.9 for details of format information.

Table 1 — Codeword capacity of all versions of rMQR symbols

Version	No. of modules/ side		Function pattern modules (C)	Format information modules (D)	Symbol capacity - Data modules except (D) (E = AB – C – D)	Data capacity [codewords] [a] (E)	Remainder bits
	Vertical (A)	Hori-zontal (B)					
R7x43	7	43	161	36	104	13	0
R7x59	7	59	206	36	171	21	3
R7x77	7	77	242	36	261	32	5
R7x99	7	99	299	36	358	44	6
R7x139	7	139	392	36	545	68	1
R9x43	9	43	181	36	170	21	2
R9x59	9	59	228	36	267	33	3
R9x77	9	77	264	36	393	49	1
R9x99	9	99	323	36	532	66	4
R9x139	9	139	418	36	797	99	5
R11x27	11	27	139	36	122	15	2
R11x43	11	43	188	36	249	31	1
R11x59	11	59	237	36	376	47	0
R11x77	11	77	273	36	538	67	2
R11x99	11	99	334	36	719	89	7
R11x139	11	139	431	36	1062	132	6
R13x27	13	27	143	36	172	21	4
R13x43	13	43	194	36	329	41	1
R13x59	13	59	245	36	486	60	6
R13x77	13	77	281	36	684	85	4
R13x99	13	99	344	36	907	113	3
R13x139	13	139	443	36	1328	166	0
R15x43	15	43	200	36	409	51	1
R15x59	15	59	253	36	596	74	4
R15x77	15	77	289	36	830	103	6
R15x99	15	99	354	36	1095	136	7
R15x139	15	139	455	36	1594	199	2
R17x43	17	43	206	36	489	61	1
R17x59	17	59	261	36	706	88	2
R17x77	17	77	297	36	976	122	0
R17x99	17	99	364	36	1283	160	3
R17x139	17	139	467	36	1860	232	4

[a] All codewords are 8 bits in length.

7.2 Data analysis

Analyse 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 7.4. Each mode in sequence from Numeric mode to Kanji mode progressively requires more bits per character. When modes are mixed, 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 does not always result in the shortest overall bit stream to change modes for a small number of characters. Also, because the capacity of symbols increases in discrete steps from one version to the next, it is not always necessary to achieve the maximum conversion efficiency in every case.

7.3 Modes

7.3.1 General

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.

NOTE $..._{HEX}$ data is figured in hexadecimal.

7.3.2 Extended channel interpretation (ECI) mode

The extended channel interpretation (ECI) protocol defined in References [21] to [23], 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. The ECI protocol provides a consistent method to specify particular interpretations of byte values before printing and after decoding.

The default interpretation for rMQR symbol shall be ECI 000003 representing the ISO/IEC 8859-1 character set.

International applications using other character sets should use the ECI protocol. For instance, the interpretation corresponding to the JIS8 and Shift JIS character sets is ECI 000020.

The effect of ECI mode is to insert an ECI escape sequence at that point in the data. It is immediately followed by another mode indicator (e.g., for efficient data encoding) and remains in force until the end of the message or a subsequent ECI mode indicator.

7.3.3 Numeric mode

Numeric mode encodes data from the decimal digit set (**0 - 9**) (byte values 30_{HEX} to 39_{HEX}). Normally, 3 data characters are represented by 10 bits.

7.3.4 Alphanumeric mode

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

7.3.5 Byte mode

In this mode, data is encoded at 8 bits per character.

In closed-system national or application-specific implementations of rMQR, an alternative 8-bit character set, for example as defined in an appropriate part of ISO/IEC 8859, may be specified for Byte mode. When an alternative character set is specified, however, the parties intending to read the rMQR symbols require to be notified of the applicable character set in the application specification or by bilateral agreement.

7.3.6 Kanji mode

The Kanji mode efficiently encodes Kanji characters in accordance with the Shift JIS system based on JIS X 0208. The Shift JIS Kanji coding values are shifted from the JIS X 0208 values. JIS X 0208:1997, Annex 1 gives details of the shift coded representation. Each two-byte character value is compacted to a 13-bit binary codeword. See Annex H for JIS8 and Shift JIS character sets.

When the character set specified for 8-bit byte mode makes use of byte values in the ranges 81_{HEX} to $9F_{HEX}$ and/or $E0_{HEX}$ to EB_{HEX}, it is not always possible to use Kanji mode unambiguously, as reading systems are unable to determine from the transmitted data whether such byte values are the lead byte of a double byte character. It may be possible to achieve a shorter bit stream by using the Kanji mode compaction rules when an appropriate sequence of byte values occurs in the data [i.e. lead bytes in the ranges 81_{HEX} to $9F_{HEX}$ and/or $E0_{HEX}$ to EA_{HEX}) followed by trailer bytes in the range 40_{HEX} to FC_{HEX},except $7F_{HEX}$, or EB_{HEX} followed by 40_{HEX} to BF_{HEX} except $7F_{HEX}$)]. Figure H.1 shows the byte combinations graphically.

7.3.7 Mixing modes

The rMQR symbols may contain sequences of data in a combination of any of the modes available for the version of the symbol as described in 7.3.2 to 7.3.6.

7.3.8 FNC1 mode

FNC1 mode is used for messages containing specific data formats. In the "first position" it designates data formatted in accordance with the GS1 General Specifications. In the "second position" it designates data formatted in accordance with a specific industry application previously agreed with AIM Inc. FNC1 mode applies to the entire symbol and is not affected by subsequent mode Indicators.

NOTE "First position" and "second position" do not refer to actual locations but are based on the positions of FNC1 character in Code 128 symbols, when used in an equivalent manner.

7.4 Data encoding

7.4.1 Sequence of data

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

The ECI header (if present) shall comprise:

— ECI mode indicator (3 bits)

— ECI designator (8, 16 or 24 bits)

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.

The remainder of the bit stream is then made up of segments each comprising: (see Figure 16):

— mode indicator;

— character count indicator;

— data bit stream.

ECI header		Segment 1			.....	Segment *n*			Terminator
ECI mode indicator	ECI designator	Mode indicator 1	Character count indicator	Data	.....	Mode indicator *n*	Character count indicator	Data	

Segment 1			Segment 2			.....	Segment *n*			Terminator
Mode indicator 1	Character count indicator	Data	Mode indicator 2	Character count indicator	Data	.....	Mode indicator *n*	Character count indicator	Data	

Figure 16 — Format of data stream with and without ECI header

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 7.4.2 to 7.4.7 shall be followed. Table 2 defines the mode indicators (3 bits) for each mode. Table 3 defines the length of the character count indicator, which varies according to the applied mode and symbol version.

Table 2 — Mode and Mode indicators for rMQR

Mode	Mode indicators (3 bits)
Numeric	001
Alphanumeric	010
Byte	011
Kanji	100
FNC1[a]	101 (First position) 110 (Second position)
ECI	111
Terminator (End of message)[b]	000

[a] See 7.4.8.2 and 7.4.8.3.

[b] The terminator is not a mode indicator as such and may be truncated.

Table 3 — Number of bits of character count indicator

Version	No. of modules/side		Numeric mode	Alphanumeric mode	Byte mode	Kanji mode
	Vertical	Horizontal				
R7x43	7	43	4	3	3	2
R7x59	7	59	5	5	4	3
R7x77	7	77	6	5	5	4
R7x99	7	99	7	6	5	5
R7x139	7	139	7	6	6	5
R9x43	9	43	5	5	4	3
R9x59	9	59	6	5	5	4

Table 3 *(continued)*

Version	No. of modules/side		Numeric mode	Alphanumeric mode	Byte mode	Kanji mode
	Vertical	Horizontal				
R9x77	9	77	7	6	5	5
R9x99	9	99	7	6	6	5
R9x139	9	139	8	7	6	6
R11x27	11	27	4	4	3	2
R11x43	11	43	6	5	5	4
R11x59	11	59	7	6	5	5
R11x77	11	77	7	6	6	5
R11x99	11	99	8	7	6	6
R11x139	11	139	8	7	7	6
R13x27	13	27	5	5	4	3
R13x43	13	43	6	6	5	5
R13x59	13	59	7	6	6	5
R13x77	13	77	7	7	6	6
R13x99	13	99	8	7	7	6
R13x139	13	139	8	8	7	7
R15x43	15	43	7	6	6	5
R15x59	15	59	7	7	6	5
R15x77	15	77	8	7	7	6
R15x99	15	99	8	7	7	6
R15x139	15	139	9	8	7	7
R17x43	17	43	7	6	6	5
R17x59	17	59	8	7	6	6
R17x77	17	77	8	7	7	6
R17x99	17	99	8	8	7	6
R17x139	17	139	9	8	8	7

The end of the data in the complete symbol is indicated by a terminator consisting of 3 zero bits (see Table 2), which is truncated if the remaining number of data bits after the data bit stream is less than the required bit length of terminator. The terminator is not a mode indicator as such.

7.4.2 Extended channel interpretation (ECI) mode

7.4.2.1 General

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 111.

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 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} can be encoded in numeric mode (see 7.4.3) as though it were a sequence of digits

0 – 9 even though it does 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.

7.4.2.2 ECI Designator

Each Extended Channel Interpretation is designated by a six-digit assignment number which is encoded in the rMQR symbol as the first one, two or three codewords following the ECI mode indicator. The encoding rules are defined in Table 4. The ECI designator appears in the data to be encoded as character $5C_{HEX}$ [\ or backslash (reverse solidus) in ISO/IEC 8859-1, ¥ or yen sign in JIS8] followed by the six-digit assignment number. Where $5C_{HEX}$ appears as true data it shall be doubled in the data string before encoding in symbols to which the ECI protocol applies.

When a single occurrence of $5C_{HEX}$ is encountered in the input to the decoder, an ECI mode indicator is inserted followed by the ECI designator. When a doubled $5C_{HEX}$ is encountered, it is encoded as two $5C_{HEX}$ bytes.

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/IEC 8859-7 (ECI 000009) in version R17x139-H symbol.

Input data: \000009**ΑΒΓΔΕ** (character values $A1_{HEX}$, $A2_{HEX}$, $A3_{HEX}$, $A4_{HEX}$, $A5_{HEX}$)

Bit sequence in symbol:

ECI mode indicator	**111**
ECI Assignment number (000009)	**0 0001001**
Mode indicator (byte)	**011**
Character count indicator (5)	**00000101**
Data:	**10100001 10100010 10100011 10100100 10100101**
Final bit string:	**111 00001001 011 00000101 10100001 10100010 10100011 10100100 10100101**

See 13.3 for example of transmission of this data following decoding.

7.4.2.3 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 can also be subject

to encryption or compaction using a transformation ECI which co-exist with the initial ECI, or a second character set ECI has the effect of terminating the first ECI and starting a new ECI segment. Where any ECI designator appears in the data, it shall be encoded in the rMQR symbol in accordance with 7.4.2.2 and shall commence a new mode segment.

7.4.3 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 mode indicator in the numeric mode is 001 (3 bits long as defined in Table 2), and the character count indicator has the number of bits defined in Table 3. The number of input data characters is converted to its binary equivalent and added as the character count indicator after the mode indicator and before the binary data sequence (see Figures 16 and 17).

EXAMPLE 1 (For Version R7x59 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 (5 bits for version R7x59):
   No. of input data characters: **16 → 10000**
5. Add mode indicator (001 for version R7x59-M) and character count indicator to binary data:
   001 10000 0000001100 0101011001 1010100110 1110000101 0011101010 0101

For any number of data characters, the length of the bit stream in numeric mode is given by Formula (1):

$$B = M + C + 10(D \text{ DIV } 3) + R \quad (1)$$

where

B is the number of bits in bit stream;

M is the number of bits in mode indicator (bit length defined in Table 2);

C is the number of bits in character count indicator (bit length defined in Table 3);

D is the 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.

		Segment 1																
		mode indicator 1	character count indicator	Data														
Mode indicator		**Numeric**																
No. of input data characters			**16**															
Input data				**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**0**	**1**	**2**	**3**	**4**	**5**
Divide into groups of three digits				012			345			678			901			234			5
Binary(10bit)				0000001100			0101011001			1010100110			1110000101			0011101010			
Last group	three digits, Binary(10bit)																		-
	two digits, Binary(7bit)																		-
	one digit, Binary(4bit)																		0101
Binary		001	10000	0000001100			0101011001			1010100110			1110000101			0011101010			0101

Figure 17 — Example of numeric mode format

7.4.4 Alphanumeric mode

Character values from 0 to 44 are assigned to each input data character 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 as 11-bit binary codes. The character value of the first character is multiplied by 45 and the character value of the second character 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 as a 6-bit binary number. The binary data is then concatenated and prefixed with the mode indicator and the character count indicator. The mode indicator in the alphanumeric mode is 010 (3 bits long as defined in Table 2), and the character count indicator has the number of bits defined in Table 3. The number of input data characters is converted to its binary equivalent and added as the character count indicator after the mode indicator and before the binary data sequence (see Figure 16).

In FNC1 mode symbols, the FNC1 character can occur in the data. It is represented in alphanumeric mode by the character %. Refer to 7.4.8.2, 7.4.8.3 and 13.4 for details of the encoding and transmission of FNC1 and %.

EXAMPLE (For Version R7x59 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 (5 bits for version R7x59):

No. of input data characters: **5 → 00101**

6. Add mode indicator 010 and character count indicator to binary data:

010 00101 00111001110 11100111001 000010

For any number of data characters, the length of the bit stream in alphanumeric mode is given by Formula (2):

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

where

B is the number of bits in bit stream;

M is the number of bits in mode indicator (bit length defined in Table 2);

C is the number of bits in character count indicator (bit length defined in Table 3);

D is the number of input data characters.

7.4.5 Byte mode

In this mode, one 8-bit codeword directly represents the byte value of the input data character specified in Table H.1 (i.e. a density of 8 bits/character).

The binary data is then concatenated and prefixed with the mode indicator and the character count Indicator. The mode indicator in the Byte mode is 011 (3 bits long as defined in Table 2), and the character count indicator has the number of bits defined in Table 3. The number of input data characters is converted to its binary equivalent and added as the character count indicator after the mode indicator and before the binary data sequence (see Figure 16).

For any number of data characters, the length of the bit stream in Byte mode is given by Formula (3):

$$B = M + C + 8D \quad (3)$$

where

B is the number of bits in bit stream;

M is the number of bits in mode indicator (bit length defined in Table 2);

C is the number of bits in character count indicator (bit length defined in Table 3);

D is the number of input data characters.

7.4.6 Kanji mode

In the Shift JIS system in JIS X 0208, Kanji characters are represented by a two-byte combination. These byte values are shifted from the JIS X 0208 values. JIS X 0208 Annex 1 gives details of the shift coded representation. 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 Mode Indicator in the Kanji mode is 100 (3 bits long as defined in Table 2), and the character count indicator has the number of bits defined in Table 3. The number of input data

characters is converted to its binary equivalent and added as the character count indicator after the mode indicator and before the binary data sequence (see Figure 16).

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

   1) subtract 8140_{HEX} from Shift JIS value;

   2) multiply most significant byte of result by $C0_{HEX}$;

   3) add least significant byte to product from 2);

   4) convert result to a 13-bit binary string.

b) For characters with Shift JIS values from $E040_{HEX}$ to $EBBF_{HEX}$:

   1) subtract $C140_{HEX}$ from Shift JIS value;

   2) multiply most significant byte of result by $C0_{HEX}$;

   3) add least significant byte to product from2);

   4) 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**

For all 8140_{HEX} to $9FFC_{HEX}$ Shift JIS characters and $E040_{HEX}$ to $EBBF_{HEX}$ Shift JIS characters, prefix binary sequence representing input data characters with mode indicator (from Table 2) and character count indicator binary equivalent (bit length defined in Table 3).

For any number of data characters, the length of the bit stream in Kanji mode is given by Formula (4):

$$B = M + C + 13D \tag{4}$$

where

B is the number of bits in bit stream;

M is the number of bits in mode indicator (bit length defined in Table 2);

C is the number of bits in character count indicator (bit length defined in Table 3);

D is the number of input data characters.

7.4.7 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 encoding. Each segment of data is encoded in the appropriate mode as indicated in 7.4.2 to 7.4.6, with the basic structure mode indicator/character count indicator/data and followed immediately by the mode indicator commencing the next segment. Figure 16 illustrates the structure of data containing n segments.

7.4.8 FNC1 modes

7.4.8.1 General

In rMQR symbols, there are two mode indicators which are used cumulatively with those defined in 7.3.2 to 7.3.8 and 7.4.2 to 7.4.8 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 13.2 and Annex E.

7.4.8.2 FNC1 in first position

NOTE "First position" is not used in a literal sense, but is a historical reference to the position of the FNC1 symbol character in Code 128 symbols.

This mode indicator identifies symbols encoding data formatted according to the GS1 Application Identifiers standard. For this purpose, it shall only be used once in a symbol and shall be placed immediately before the first mode indicator used for efficient data encoding (numeric, alphanumeric, byte or Kanji), and after any ECI header. Where the GS1 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), rMQR symbols shall use the % character in alphanumeric mode or character $^{G}_{S}$ ($1D_{HEX}$) in byte mode to perform this function. If the % character occurs as part of the data in Alphanumeric mode, it shall be encoded as %%. Decoders encountering % in these symbols shall transmit it as $1D_{HEX}$, and if %% is encountered it shall be transmitted as a single % character.

EXAMPLE 1 (For Version R15x139 symbol)

Input data:

0194912345123452 (Application identifier 01 = global trade item number (GTIN), fixed length; data: 94912345123452)

15180331 (Application identifier 15 = "Best before" date YYMMDD, fixed length; data: 31 March 2018)

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:

0194912345123452151803313012 8%10ABC123

Bit sequence in symbol:

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

001 (Mode indicator, numeric mode)

000011101 (Character count indicator, 29)

<data bits for **0194912345123452151803313012 8**>

010 (Mode indicator, alphanumeric mode)

00001001 (Character count indicator, 9)

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

Transmitted data (see 13.2 and Annex E)

] Q30194912345123452151803313012 8**<1D$_{HEX}$>**10ABC123

EXAMPLE 2 Encoding/transmission of % character in data:

Input data:	**123 %**
Encoded as:	**123 %%**
Transmitted as:	**123 %**

7.4.8.3 FNC1 in second position

NOTE "Second position" is not used in a literal sense, but is a historical reference to the position of the FNC1 symbol character in Code 128 symbols.

This mode indicator identifies symbols formatted in accordance with specific industry or application specifications previously agreed with References [21] to [23]. 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 References [21] to [23].. For this purpose, it shall only be used once in a symbol and shall be placed immediately before the first mode indicator used for efficient data encoding (numeric, alphanumeric, byte or Kanji). 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. Where the application specifications call for the FNC1 character (in other symbologies which use this special character) to be used as a data field separator, rMQR symbols shall use the % character in Alphanumeric mode or character $^{G}_{S}$ ($1D_{HEX}$) in 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 $1D_{HEX}$, and if %% is encountered, it shall be transmitted as a single % character.

EXAMPLE (For Version R17x139 symbol)

NOTE Application indicator 37 has not been assigned at the time of publication to any organization and the data content of the example is purely arbitrary.

Application indicator:	37
Input data:	AA1234BBB112text text text text<CR>
Bit sequence in symbol:	
	110 (Mode Indicator, FNC1 implied in 2nd position)
	00100101 (Application indicator, 37)
	010 (Mode indicator, alphanumeric mode)
	000001100 (Character count indicator, 12)
	<data bits for **AA1234BBB112**>
	011 (Mode indicator, byte mode)
	00010100 (Character count indicator, 20)
	<data bits for **text text text text<CR>** >
Transmitted data:	
	] Q537AA1234BBB112text text text text<CR>

7.4.9 Terminator

The end of data in the symbol is signalled by the terminator sequence of 0 bits, as defined in Table 2, appended to the data bit stream following the final mode segment. The terminator shall 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 the required bit length of terminator.

7.4.10 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 as defined in 7.4.9. 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 it does not end at a codeword boundary, padding bits with binary value 0 shall be added after the final bit (least significant bit) of the data stream to extend it to the codeword boundary. The data 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 Table 6, by adding the remainder codewords **11101100** and **00010001** alternately. The resulting series of codewords, the data codeword sequence, is then processed as described in 7.5 to add error correction codewords to the message. In certain versions of symbol, it may be necessary

to add remainder bits (all zeros) to the end of the message, after the final error correction codeword, in order exactly to fill the symbol capacity (see Table 1).

Table 6 — Number of data codewords and input data capacity for rMQR

Version	Error correction level	Number of data codewords	Number of data bits	Data capacity			
				Numeric	Alphanu-meric	Byte	Kanji
R7x43	M H	6 3	48 24	12 5	7 3	5 2	3 1
R7x59	M H	12 7	96 56	26 14	16 8	11 6	6 3
R7x77	M H	20 10	160 80	45 21	27 13	19 9	11 5
R7x99	M H	28 14	224 112	64 30	39 18	27 13	16 8
R7x139	M H	44 24	352 192	102 54	62 33	42 22	26 14
R9x43	M H	12 7	96 56	26 14	16 8	11 6	6 3

NOTE The number of data bits includes bits for mode indicator and character count indicator.

Table 6 *(continued)*

Version	Error correction level	Number of data codewords	Number of data bits	Data capacity			
				Numeric	Alphanu-meric	Byte	Kanji
R9x59	M H	21 11	168 88	47 23	29 14	20 10	12 6
R9x77	M H	31 17	248 136	71 37	43 23	30 16	18 9
R9x99	M H	42 22	336 176	97 49	59 30	40 20	25 12
R9x139	M H	63 33	504 264	147 75	89 46	61 31	38 19
R11x27	M H	7 5	56 40	14 9	8 6	6 4	3 2
R11x43	M H	19 11	152 88	42 23	26 14	18 10	11 6
R11x59	M H	31 15	248 120	71 33	43 20	30 14	18 8
R11x77	M H	43 23	344 184	100 52	60 31	41 21	25 13
R11x99	M H	57 29	456 232	133 66	81 40	55 27	34 17
R11x139	M H	84 42	672 336	198 97	120 59	82 40	51 25
R13x27	M H	12 7	96 56	26 14	16 8	11 6	6 3
R13x43	M H	27 13	216 104	62 28	37 17	26 12	16 7
R13x59	M H	38 20	304 160	88 45	53 27	36 18	22 11
R13x77	M H	53 29	424 232	124 66	75 40	51 27	31 17
R13x99	M H	73 35	584 280	171 80	104 49	71 33	44 20
R13x139	M H	106 54	848 432	251 126	152 76	104 52	64 32
R15x43	M H	33 15	264 120	76 33	46 20	31 13	19 8
R15x59	M H	48 26	384 208	112 59	68 36	46 24	28 15
R15x77	M H	67 31	536 248	157 71	95 43	65 29	40 18
R15x99	M H	88 48	704 384	207 111	126 68	86 46	53 28
R15x139	M H	127 69	1016 552	301 162	182 98	125 67	77 41
R17x43	M H	39 21	312 168	90 47	55 28	37 19	23 12
R17x59	M H	56 28	448 224	131 63	79 38	54 26	33 16
R17x77	M H	78 38	624 304	183 87	111 53	76 36	47 22

NOTE The number of data bits includes bits for mode indicator and character count indicator.

Table 6 *(continued)*

Version	Error correction level	Number of data codewords	Number of data bits	Data capacity			
				Numeric	Alphanu-meric	Byte	Kanji
R17x99	M H	100 56	800 448	236 131	143 79	98 54	60 33
R17x139	M H	152 76	1216 608	361 178	219 108	150 74	92 46
NOTE The number of data bits includes bits for mode indicator and character count indicator.							

7.5 Error correction

7.5.1 Error correction capacity

rMQR employs Reed-Solomon error control coding to detect and correct errors. A series of error correction codewords is generated, which are added to the data codeword sequence in order to enable the symbol to withstand damage without loss of data. There are two user-selectable levels of error correction, as shown in Table 7, offering the capability of recovery from the following amounts of damage:

Table 7 — Error correction levels

Error correction level	Recovery capacity % (approx.)
M	15
H	30

J.2 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 rMQR is a matrix symbology, a defect converting a module from dark to light or vice versa results 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 Formula (5):

$$e + 2t \leq d - p \tag{5}$$

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 misdecode protection codewords.

In the general case, $p = 0$. However, if most of the error correction capacity is used to correct erasures, then the possibility of an undetected error is increased. Whenever the number of erasures is more than half the number of error correction codewords, $p = 1$ or $p = 2$. For small symbols with less than 5 error correction codewords, erasure correction should not be used ($e = 0$ and $p > 0$).

For example, in a version R9x77-H symbol there is a total of 49 codewords, of which 32 are error correction codewords (leaving 17 data codewords). The 32 error correction codewords can correct 16 misdecodes or substitution errors, i.e. 16/49 or 32,6 % of the symbol capacity.

In Formula (5), the following values should be assigned to *p*:

— *p* = 2 in symbols of Version R11x27-M;

— *p* = 1 in symbols of Version R7x43-M, R7x59-M, R9x43-M, R13x27-M;

— *p* = 0 in all other cases.

Where *p* > 0, there are p (i.e. 1, or 2) codewords which act as error detection codewords and prevent transmission of data from symbols where the number of errors exceeds the error correction capacity, and *e* shall be less than *d*/2. In a Version R11x27-M symbol, for example, the total number of codewords is 15; of these, 7 are data codewords and 8 error correction codewords. From Table 8, the error correction capacity is 3 errors (where *e* = 0). Substituting in the formula above,

$$0 + (2 \times 3) \leq 8 - 2$$

meaning that the correction of the 3 errors requires only 6 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 3 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. Table 8 lists, 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 as defined in 3.9.

Table 8 — Error correction characteristics for rMQR

Version	Total number of codewords	Error correction level	Number of error correction codewords	Value of p	Number of error correction blocks	Error correction code per block (c, k, r) [a]
R7x43	13	M	7	1	1	(13,6,3) [b]
		H	10	0	1	(13,3,5)
R7x59	21	M	9	1	1	(21,12,4) [b]
		H	14	0	1	(21,7,7)
R7x77	32	M	12	0	1	(32,20,6)
		H	22	0	1	(32,10,11)
R7x99	44	M	16	0	1	(44,28,8)
		H	30	0	1	(44,14,15)
R7x139	68	M	24	0	1	(68,44,12)
		H	44	0	2	(34,12,11)
R9x43	21	M	9	1	1	(21,12,4) [b]
		H	14	0	1	(21,7,7)
R9x59	33	M	12	0	1	(33,21,6)
		H	22	0	1	(33,11,11)
R9x77	49	M	18	0	1	(49,31,9)
		H	32	0	1 1	(24,8,8) (25,9,8)

[a] c = total number of codewords, k = number of data codewords, r = error correction capacity
[b] Error correction capacity is less than half the number of error correction codewords to reduce the probability of misdecodes.

Table 8 *(continued)*

Version	Total number of codewords	Error correction level	Number of error correction codewords	Value of p	Number of error correction blocks	Error correction code per block (c, k, r) [a]
R9x99	66	M	24	0	1	(66,42,12)
		H	44	0	2	(33,11,11)
R9x139	99	M	36	0	1 1	(49,31,9) (50,32,9)
		H	66	0	3	(33,11,11)
R11x27	15	M	8	2	1	(15,7,3) [b]
		H	10	0	1	(15,5,5)
R11x43	31	M	12	0	1	(31,19,6)
		H	20	0	1	(31,11,10)
R11x59	47	M	16	0	1	(47,31,8)
		H	32	0	1 1	(23,7,8) (24,8,8)
R11x77	67	M	24	0	1	(67,43,12)
		H	44	0	1 1	(33,11,11) (34,12,11)
R11x99	89	M	32	0	1 1	(44,28,8) (45,29,8)
		H	60	0	1 1	(44,14,15) (45,15,15)
R11x139	132	M	48	0	2	(66,42,12)
		H	90	0	3	(44,14,15)
R13x27	21	M	9	1	1	(21,12,4) [b]
		H	14	0	1	(21,7,7)
R13x43	41	M	14	0	1	(41,27,7)
		H	28	0	1	(41,13,14)
R13x59	60	M	22	0	1	(60,38,11)
		H	40	0	2	(30,10,10)
R13x77	85	M	32	0	1 1	(42,26,8) (43,27,8)
		H	56	0	1 1	(42,14,14) (43,15,14)
R13x99	113	M	40	0	1 1	(56,36,10) (57,37,10)
		H	78	0	1 2	(37,11,13) (38,12,13)
R13x139	166	M	60	0	2 1	(55,35,10) (56,36,10)
		H	112	0	2 2	(41,13,14) (42,14,14)
R15x43	51	M	18	0	1	(51,33,9)
		H	36	0	1 1	(25,7,9) (26,8,9)

[a] c = total number of codewords, k = number of data codewords, r = error correction capacity
[b] Error correction capacity is less than half the number of error correction codewords to reduce the probability of misdecodes.

Table 8 *(continued)*

Version	Total number of codewords	Error correction level	Number of error correction codewords	Value of p	Number of error correction blocks	Error correction code per block (c, k, r) [a]
R15x59	74	M	26	0	1	(74,48,13)
		H	48	0	2	(37,13,12)
R15x77	103	M	36	0	1 1	(51,33,9) (52,34,9)
		H	72	0	2 1	(34,10,12) (35,11,12)
R15x99	136	M	48	0	2	(68,44,12)
		H	88	0	4	(34,12,11)
R15x139	199	M	72	0	2 1	(66,42,12) (67,43,12)
		H	130	0	1 4	(39,13,13) (40,14,13)
R17x43	61	M	22	0	1	(61,39,11)
		H	40	0	1 1	(30,10,10) (31,11,10)
R17x59	88	M	32	0	2	(44,28,8)
		H	60	0	2	(44,14,15)
R17x77	122	M	44	0	2	(61,39,11)
		H	84	0	1 2	(40,12,14) (41,13,14)
R17x99	160	M	60	0	2 1	(53,33,10) (54,34,10)
		H	104	0	4	(40,14,13)
R17x139	232	M	80	0	4	(58,38,10)
		H	156	0	2 4	(38,12,13) (39,13,13)

[a] c = total number of codewords, k = number of data codewords, r = error correction capacity
[b] Error correction capacity is less than half the number of error correction codewords to reduce the probability of misdecodes.

7.5.2 Generating the error correction codewords

The data codewords including remainder codewords as necessary shall be divided into the number of blocks shown in Table 8. Error correction codewords shall be calculated for each block and appended to the data codewords for each block.

The polynomial arithmetic for rMQR shall be calculated using bit-wise modulo 2 arithmetic and bytewise 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 zero-power term being the last data codeword before the first error correction codeword.

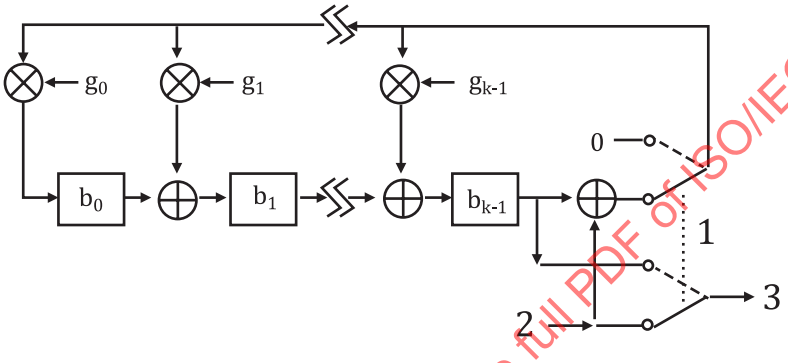
The error correction codewords are the remainder after dividing the data codewords by a polynomial $g(x)$ used for error correction codes (see Annex A). The highest order coefficient of the remainder is the

first error correction codeword and the zero-power coefficient is the last error correction codeword and the last codeword in the block.

NOTE If this calculation is performed by "long division", the symbol data polynomial needs to be first multiplied by x^k.

Fourteen different generator polynomials of Annex A shall be used for generating the error correction codewords for rMQR symbols.

This can be implemented by using the division circuit as shown in Figure 18. 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 ($n + 1 \ldots n + k$ clock pulses), with the switch in the up position, the error correction codewords $\varepsilon_{k-1} \ldots \varepsilon_0$ are generated by flushing the registers in order while keeping the data input at 0.



Key

⊕ = GF (256) addition

⊗ = GF (256) multiplication

Figure 18 — Error correction codeword encoding circuit

7.6 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 6 and 8.

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 Table 8 according to the version and error correction level.

2) For each data block, calculate a corresponding block of error correction codewords as defined in 7.5.2 and Annex A.

3) Assemble the final sequence by taking data and error correction codewords from each block in turn. For example, if there are three blocks, the sequence would be: data block 1 to 3, first codeword (D1); data block 1 to 3, second codeword (D12); .. . data block 1 to 3, final codeword (D35); ...; then error correction block 1 to 3, first codeword (E1), error correction block 1 to 3, second codeword (E27), ... and similarly to error correction block 1 to 3, final codeword (E78), which consist of rMQR symbol. rMQR symbols contain data and error correction blocks which always exactly fill the symbol codeword capacity. In certain rMQR versions, however, where the number of modules available for data and error correction codewords is not an exact multiple of 8, there may be a need for 1 to 7 remainder bits to be appended to the final message bit stream in order to fill exactly 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 R13x99-H symbol comprises three data blocks and error correction blocks, and the first one contains 11 data and 26 error correction codewords respectively, while the second and third pairs of blocks contain 12 data and 26 error correction codewords respectively. In this symbol, the character arrangement can be depicted as shown in Figure 19. Each row of the figure 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 figure in turn.

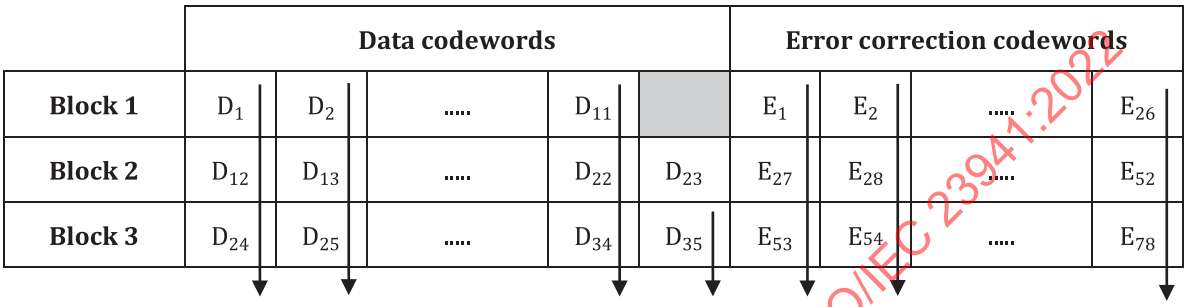


Figure 19 — Constructing the final message codeword sequence

The final message codeword sequence for the Version R13x99-H symbol is therefore:

The symbol module capacity is filled by adding 5 remainder (0) bits as needed after the final codeword.

7.7 Codeword placement in matrix

7.7.1 Symbol character representation

There are two types of symbol character in the rMQR 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.

7.7.2 Function pattern placement

A rectangular blank matrix shall be constructed with the number of horizontally and vertically positioned modules corresponding to the version in use. Positions corresponding to the finder pattern, separator, timing pattern, finder sub pattern, corner finder pattern and alignment patterns shall be filled with either dark modules or light modules as appropriate. Module positions for the format information shall be left temporarily blank. These blank positions are shown in Figures 23 and 24 and are common to all versions. Annex D defines the positioning of alignment patterns.

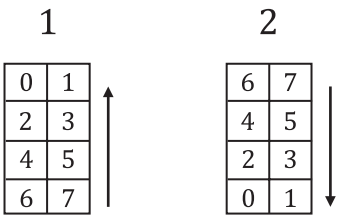
7.7.3 Symbol character placement

In the encoding region of the rMQR 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 23 and 24 illustrate Version R7x59 and Version R13x59 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 (see Figure 20).

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 (see Figure 20). The most significant bit therefore occupies the lower right module of a 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 upper left module of a symbol character if the previous character has ended in the right-hand module column (see Figure 22).



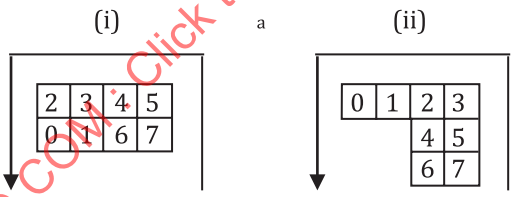
Key

1 upwards

2 downwards

Figure 20 — Bit placement in regular symbol character in upwards and downwards directions

c) When the upper or lower boundary of the symbol character region is reached (i.e. the edge of the timing pattern, corner finder pattern, format information, or separator), any remaining bits in the codeword shall be placed in the next column to the left. The direction of placement reverses (see Figure 21).



Key

a Upwards to downwards.

Figure 21 — Example of bit placement in symbol characters when direction of placement changes

d) When the right-hand module column of the symbol character column encounters an alignment pattern, or an area occupied by format information, bits are placed to form an irregular symbol character, extending along the single module column adjacent to the alignment pattern or format 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 (see Figure 22).

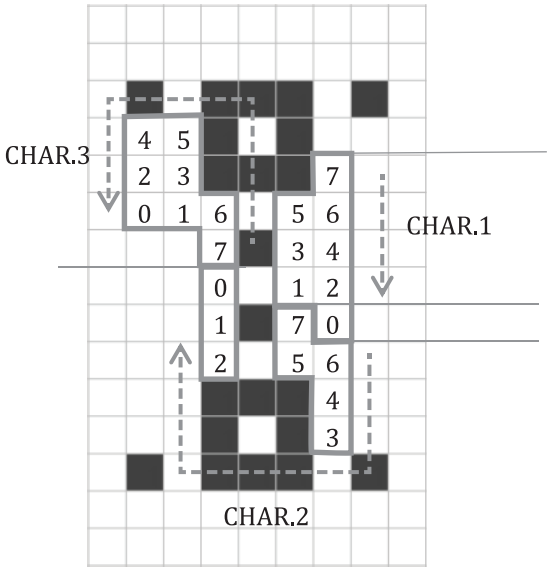
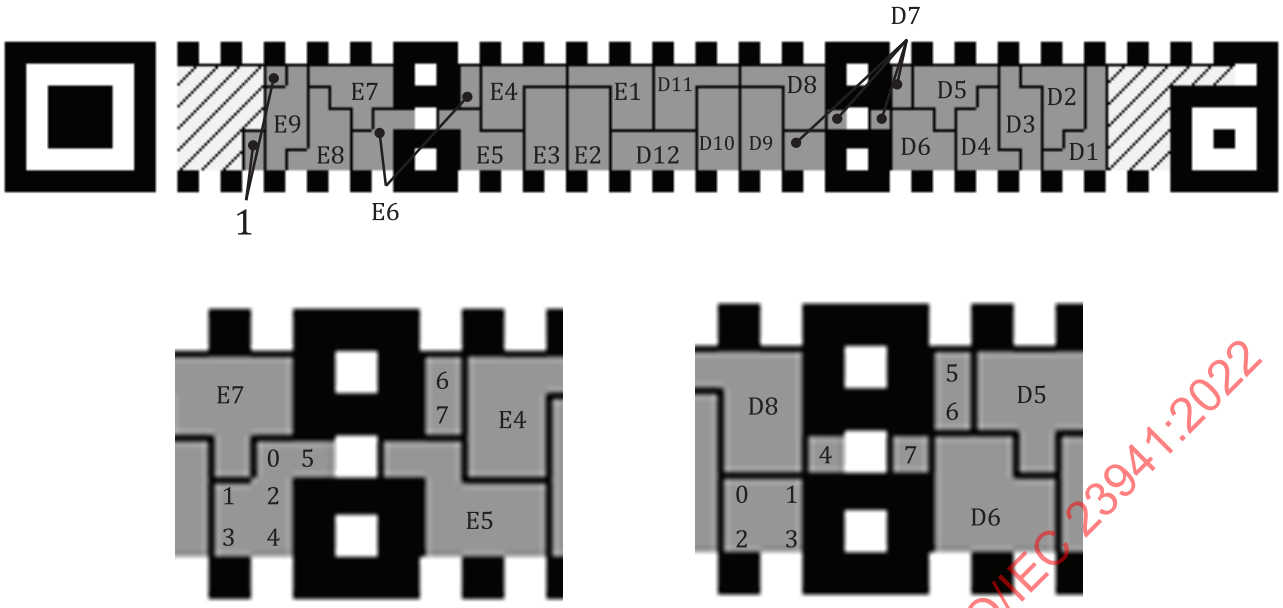


Figure 22 — Example of bit placement adjacent to alignment pattern

An alternative method for placement in the symbol, which yields the same result, is to regard the two-module wide, right and left 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 8-bit symbol characters, the appropriate number of remainder bits (1 to 7 as shown in Table 1) shall be used to fill the symbol capacity. These remainder bits shall always have the value 0 before data masking according to 7.8.



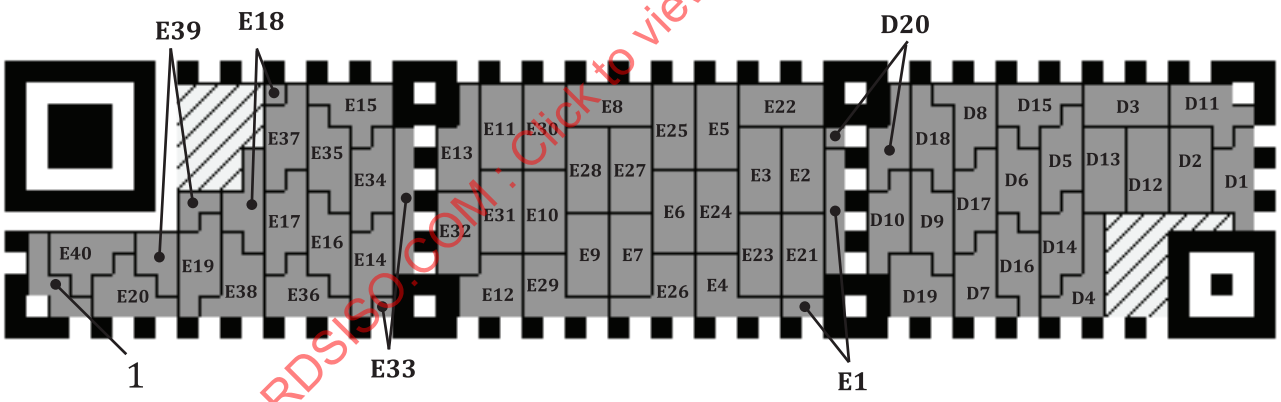
Key

1 remainder bit

D1-D12 : data codeword

E1-E9 : EC codeword

Figure 23 — Symbol character arrangement in Version R7x59-M symbol



Key

1 remainder bit

D1-D10 : data block1

D11-D20 : data block2

E1-E20 : EC block1

E21-E40 : EC block2

Figure 24 — Symbol character arrangement in Version R13x59-H symbol

7.8 Data masking

7.8.1 General

For reliable rMQR reading, it is preferable for dark and light modules to be arranged in a well-balanced manner in the symbol. To meet the above conditions, data masking should be applied according to the following steps.

1) Data masking is not applied to function patterns.

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

7.8.2 Data mask patterns

The data mask pattern generation condition is illustrated as below. A module filling the formula corresponds to the dark module.

$$((i \text{ DIV } 2) + (j \text{ DIV } 3)) \text{ MOD } 2 = 0$$

The data mask pattern is generated by defining as dark any module in the encoding region (excluding the area reserved for format 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. Figure 25 illustrates the mask pattern for Version R17x43 symbol.

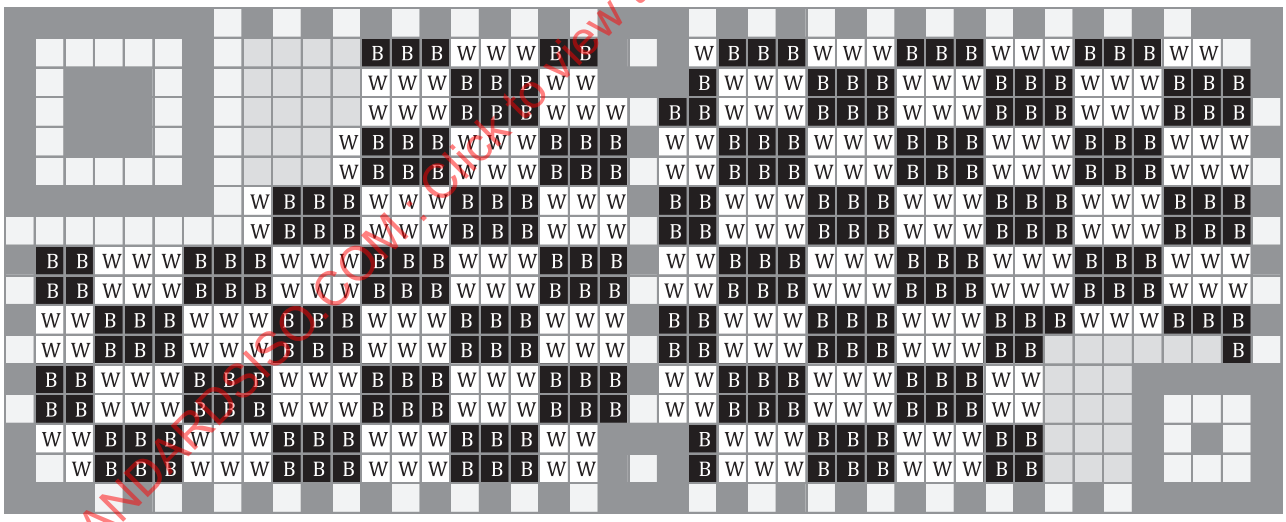


Figure 25 — Mask pattern for Version R17x43 rMQR symbol

7.9 Format information

The format information shall be an 18-bit sequence containing 6 data bits, with 12 error correction bits calculated using the (18, 6) Extended BCH code. For the detail of the error correction calculation for the format information, refer to Annex C. The first one data bit contains the error correction level of the symbol, indicated in Table 9.

Table 9 — Error correction level indicators

Error correction level	Binary indicator
M	0
H	1

The second to sixth data bits of the format information represent versions defined in Table 10.

Table 10 — Version indicator

Version	Version indicator
R7x43	00000
R7x59	00001
R7x77	00010
R7x99	00011
R7x139	00100
R9x43	00101
R9x59	00110
R9x77	00111
R9x99	01000
R9x139	01001
R11x27	01010
R11x43	01011
R11x59	01100
R11x77	01101
R11x99	01110
R11x139	01111
R13x27	10000
R13x43	10001
R13x59	10010
R13x77	10011
R13x99	10100
R13x139	10101
R15x43	10110
R15x59	10111
R15x77	11000
R15x99	11001
R15x139	11010
R17x43	11011
R17x59	11100
R17x77	11101
R17x99	11110
R17x139	11111

The 12 error correction bits shall be calculated as described in Annex C and appended to the 6 data bits. The 18-bit error corrected format information shall then be XORed with the bit pattern **011111101010110010** (for the finder pattern side) or **100000101001111011** (for the finder sub pattern side), in order to ensure that no combination of error correction level and version indicator results 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 26. 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 module numbered 0, and the most significant bit in the module numbered 17 in Figure 26.

EXAMPLE

error correction level H:	**1**
Version indicator:	**00000**
Data bit (Error correction level indicator, mask pattern reference):	**100000**
Extended BCH bits:	**1011011110000**
Unmasked bit sequence:	**100001011011110000**
Mask pattern for XOR operation (finder pattern side):	**011111101010110010**
Format information module pattern (finder pattern side):	**111111001101100111**
	| |
	bit 17 bit 0

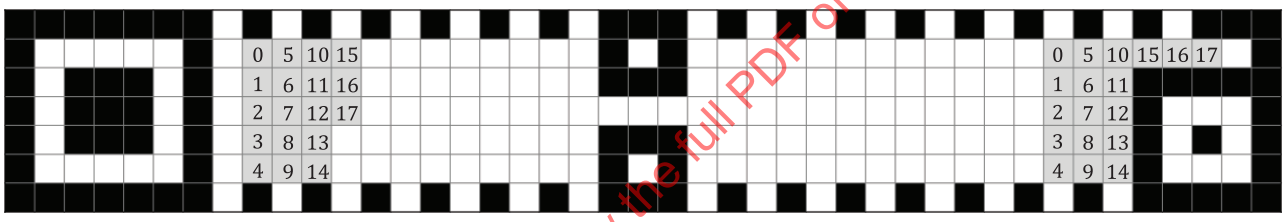


Figure 26 — rMQR symbol format information bit positions

8 Symbol printing and marking

8.1 Dimensions

rMQR 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 2X wide on all four sides.

8.2 Human-readable interpretation

Because rMQR symbols are capable of encoding thousands of characters, a human readable interpretation of the data characters can be impractical. 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 shall not interfere with neither the symbol itself nor the quiet zones.

8.3 Marking guidelines

rMQR symbols can be printed or marked using a number of different techniques. Annex J provides user guidelines.

9 Symbol quality

9.1 Methodology

rMQR symbols shall be assessed for quality using the 2D matrix bar code symbol print quality guidelines defined in ISO/IEC 15415, as augmented and modified below.

Some marking technologies can be unable to produce symbols conforming to this document without taking special precautions. Annex J gives additional guidance to help any printing system achieve valid rMQR symbols.

It is possible that direct part marked (DPM) symbols and/or symbols printed with disconnected dots do not pass the requirements of ISO/IEC 15415 and can be unreadable by rMQR scanners. Applications requiring such symbols should specify quality measurement criteria using ISO/IEC 29158, which is the quality extension of ISO/IEC 15415. The use of which may require specialized DPM scanners.

9.2 Symbol quality parameters

9.2.1 Fixed pattern damage

The measurement and grading basis for fixed pattern damage of Annex F shall apply.

9.2.2 Scan grade and overall symbol grade

The scan grade shall be the lowest of the grades for symbol contrast, modulation, fixed pattern damage, decode, axial non-uniformity, grid non-uniformity and unused error correction in an individual image of the symbol. The overall symbol grade is the arithmetic mean of the individual scan grades for a number of tested images of the symbol.

9.2.3 Grid non-uniformity

The ideal grid is calculated by using the finder pattern, timing pattern and alignment patterns as datum points, as located by the use of the reference decode algorithm (see Clause 11).

9.3 Process control measurements

A variety of tools and methods can be used to perform useful measurements for monitoring and controlling the process of creating rMQR symbols. These are described in Annex L. These techniques do not constitute a print quality check of the produced symbols (the method specified earlier in this Clause 9 and the required method for assessing symbol print quality of Annex F shall apply) but they individually and collectively yield good indications of whether the symbol print process is creating workable symbols.

10 Decoding procedure overview

The decoding steps from reading a rMQR symbol to outputting data characters are the reverse of the encoding procedure. Figure 27, and the following information, outline 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. Identify reflectance polarity (reflectance reversal or not) from finder pattern module colouring.

2) Read the format information. Release the masking pattern and perform error correction on the format information modules as necessary; If successful, symbol is in normal orientation, otherwise attempt mirror image decoding of format information. Identify version and error correction level.

3) Release the data masking by XORing the encoding region bit pattern with the data masking pattern.

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

5) Detect errors using the error correction codewords corresponding to the level information. If any error is detected, correct it.

6) Divide the data codewords into segments according to the mode indicators and character count indicators.

7) Finally, decode the data characters in accordance with the mode(s) in use and output the result.

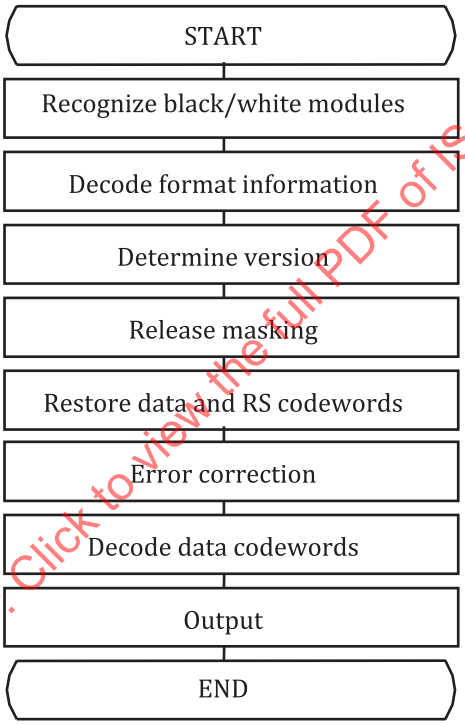


Figure 27 — rMQR decoding steps

11 Reference decode algorithm

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

a) Determine a global threshold according to the method defined in ISO/IEC 15415. Convert the image to a set of dark and light pixels using the global threshold.

b) Locate the finder pattern. The finder pattern in rMQR is a single finder pattern. As described in 6.3.3, module widths in each finder pattern form 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).

   1) 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 finder

pattern (see Figure 28). Repeat this for adjacent pixel lines in the image until all lines crossing the central box of the finder pattern in the x axis of the image have been identified.

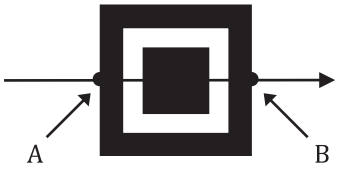


Figure 28 — Scan line in finder pattern

2) Repeat 1) for pixel columns crossing the central box of the finder pattern in the y axis of the image.

3) Locate the centre 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 finder 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 centre of the pattern is located at the intersection of these two lines.

4) If no candidate areas are detected, reverse the colouring of the light and dark pixels and recommence at the beginning of b) to attempt to decode the symbol as a symbol with reflectance reversal.

c) Determine the possible angles of rotation of the symbol by analysing the angles of the lines from b) 3) relative to the imaging sensor axes, as ϑ (see Figure 29), $\vartheta + 90°$, $\vartheta + 180°$ and $\vartheta + 270°$.

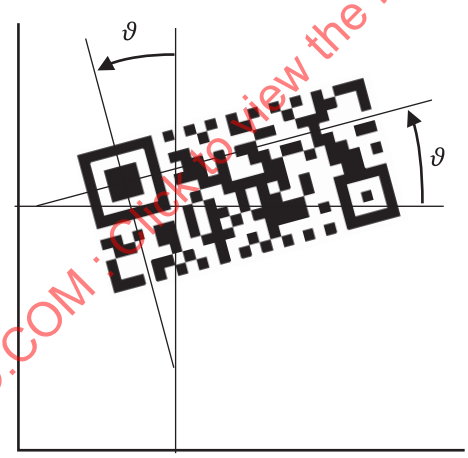


Figure 29 — angle ϑ relative to the imaging sensor axes

d) Plot three lines parallel to each axis of the finder pattern and equally spaced across the pattern and measure the distances from point A to point B on each line (see Figure 28). The spacing is not limited but three lines shall be in the finder pattern. Divide the average of these three distances A to B by 7 and obtain the provisional horizontal module dimension X of the symbol.

e) In the same procedure as d), calculate the value in the vertical direction and obtain the provisional vertical module dimension Y of the symbol.

f) Taking each side of the outer box of the finder pattern in turn, extend a line outward from the finder pattern in both directions, parallel to the edge and 0,5X and 0,5Y in from the edge. The intersection of these lines becomes the central coordinate of 4-corner module of finder pattern (see Figure 30).

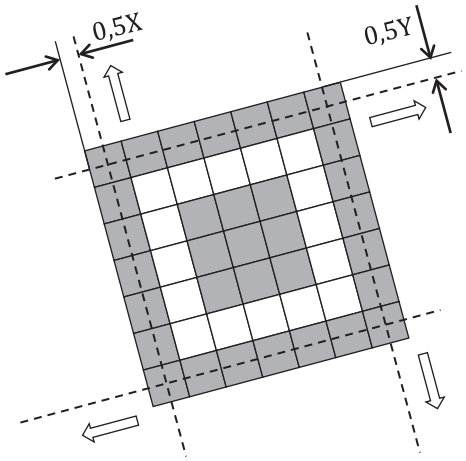


Figure 30 — Central coordinate of 4-corner module of finder pattern

g) To obtain the format information, create a sampling grid using the finder pattern and the module dimension X and Y, and expand it up to the format information area. Determine the format information bit string by taking the dark pixels as binary 1 and light pixels as binary 0.

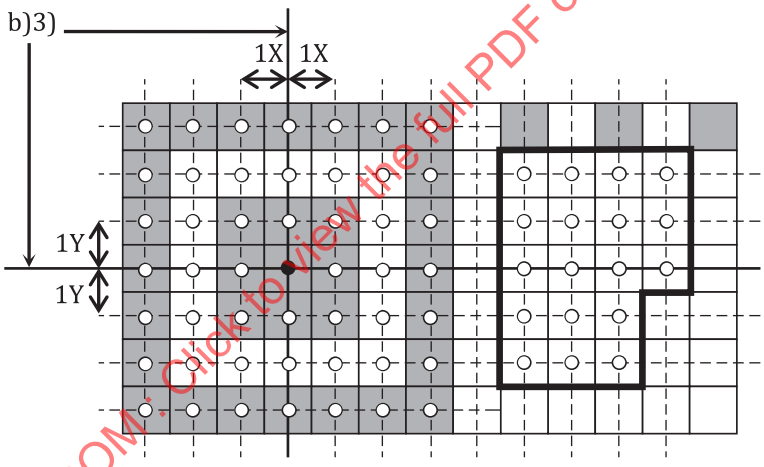


Figure 31 — Finder pattern and format information

h) Release masking of the format information by XORing the bit string with the pattern given in 7.9 and decode the format information (applying the error correction procedure given in Annex C if necessary) to yield the symbol number (and hence the version and error correction level of the symbol). If a valid format information bit string is obtained, change the module dimension accordingly so that the horizontal module dimension is X and the vertical module dimension is Y. If the format information bit string is not a valid sequence, repeat the procedure from Step g) in another direction. If a valid format information bit string cannot be derived, determine whether it is a valid sequence if read in the reverse direction and if so attempt to continue decoding as a mirror image symbol with the image row and column coordinates transposed.

i) Search for the horizontal timing pattern of the upper side

Based on the information of the number of horizontal modules obtained from the format information, check the horizontal separator adjacent to the finder pattern and the timing pattern.

1) The point to start searching the timing pattern is the central coordinate [the intersection coordinate on the line drawn in f)] of the upper right corner module of the finder pattern. The point spaced X from this point and on the line drawn in f) (a point outside the finder pattern) is the centre point of one module contained in a separator that separates the finder pattern

and data area. Although it is not a light module of the timing pattern, it is considered as a provisional light module of the timing pattern when searching so that the centre of the module of the timing pattern is adjusted to display dark and light modules alternately.

2) Set the provisional central coordinates of the module spaced with 1X from the starting point of the timing pattern. Check that 3 consecutive dark modules and 1 light module are found subsequent to the light module at a distance of 11X, 13X, 15X, 17X, or 19X corresponding to the number of horizontal modules. The first dark module of the 3 consecutive ones is the point to end searching. Also check that more than 75 % of the provisional central coordinates between the start and end points for searching have the dark and light assignment consistent with that for the timing pattern.

3) When the horizontal timing pattern is not found, repeat the procedure from Step g) in another direction. Based on the found horizontal timing pattern, it is considered as the normal orientation in Figure 1 a) and the subsequent processing is performed.

4) When the horizontal timing pattern is found, update the horizontal module dimension X by dividing the distance between the left edge of the finder pattern on the line drawn in f) and the left edge of the found timing pattern by the number of contained modules.

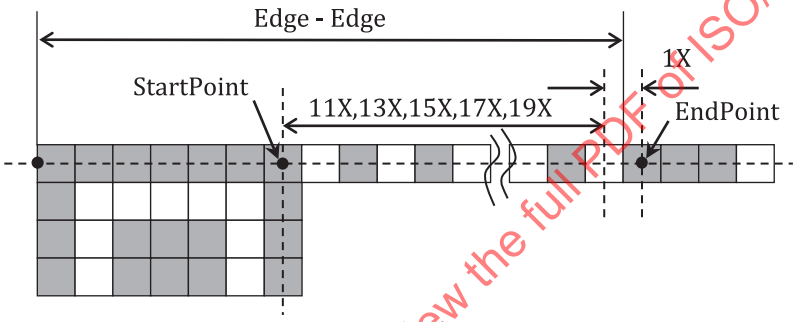


Figure 32 — Search for the horizontal timing pattern

j) Three consecutive dark modules obtained in i) are components of the corner finder pattern when the number of horizontal modules obtained from the format information is 27, and components of the alignment pattern when the number of horizontal modules is other than 27. Set central coordinates of each module contained in the corner finder pattern or the alignment pattern as described below and check the light and dark alternation.

1) Identify the central coordinate of the dark module in the centre of the three consecutive dark modules overlapping the horizontal timing pattern.

i) Obtain left and right edges (A, B) of the alignment pattern on the line as shown in 3) and Figure 34.

ii) Obtain upper and lower edges (C, D) of the centre dark module by drawing a line through the intermediate point of AB and perpendicular to Line AB.

iii) Take the intermediate point of CD as the central coordinate of the centre dark module.

2) When the number of horizontal modules is 27, check the upper right corner finder pattern and set the central coordinates as specified below:

i) Make a grid with lines spaced with 1X rightward and leftward from the central coordinate of 1) and parallel to CD and lines spaced with 1Y or 2Y downward and parallel to AB. Take the intersections of these lines as the central coordinates of each module of the corner finder pattern.

ii) Check that more than 75 % of the central coordinates have the dark and light assignment consistent with that for the corner finder pattern.

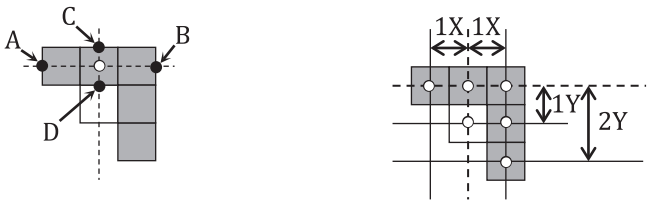


Figure 33 — Search for the corner finder pattern

3) When the number of horizontal modules is other than 27, check the alignment pattern and set the central coordinates as specified below:

   i) Make a grid with lines spaced with 1X rightward and leftward from the central coordinate of 1) and parallel to CD and lines spaced with 1Y or 2Y downward and parallel to AB. Take the intersections of these lines as the central coordinates of each module of the alignment pattern.

   ii) Check that more than 75 % of the central coordinates have the dark and light assignment consistent with that for the alignment pattern.

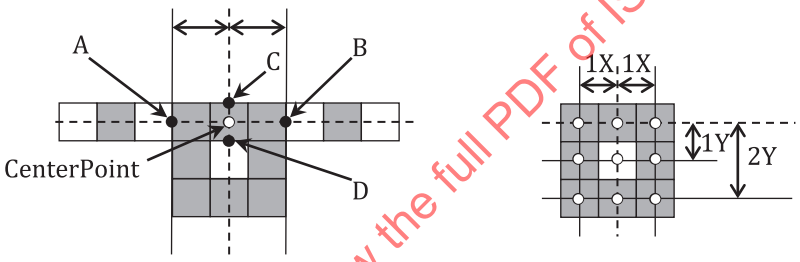


Figure 34 — Search for alignment pattern

k) Setting of the central coordinates of the timing pattern

Set central coordinates of each module contained in the timing pattern as described in Step 1) to 9).

1) Take the central coordinate of the upper right corner module of the finder pattern as the starting point and the central coordinate of the upper left corner module of the alignment pattern or corner finder pattern as the ending point.

2) On the line connecting the starting and ending points, put points P_1, P_2, ..., P_n in turn where modules alternate light to dark, and vice versa between starting and ending points.

3) When a distance is established between P_1 and P_3 and the deviation of its distance from 2X is 25 % or less, take the central coordinate between P_1 and the next point P_2 as the central coordinate of the module.

4) When a distance between P_1 and P_3 and a distance between P_2 and P_4 are established and the deviations of these distances from 2X are 25 % or less, take the central coordinate between P_2 and the next point P_3 as the central coordinate of the module.

5) Repeat Step 4) up to the last point, i.e. take the central coordinate between P_{n-2} and the next point P_{n-1} as the central coordinate of the module.

6) When a distance between P_{n-2} and P_n is established the deviation of its distances from 2X is 25 % or less, take the central coordinate between the last point P_{n-1} and P_n as the central coordinate of the module.

7) For central coordinates obtained in Step 3) through 6) between the start point and end point, if the deviation of a distance between the central coordinates from X is 50 % or more, calculate the provisional central coordinate in the procedure specified below:

   i) Divide a distance between central coordinates by X and round the obtained value to the nearest whole number. Take this value as the provisional number of modules.

   ii) Divide a distance between central coordinates equally by the provisional number of modules between central coordinates. Take the obtained value as the provisional central coordinate.

8) Implement the procedures in Step 7) for all distances between central coordinates when the deviation of them from X is 50 % or more.

9) Check that the number of central coordinates and the provisional central modules obtained in Step 1) through 8) is consistent with the number of the known timing patterns that can be obtained from the format information.

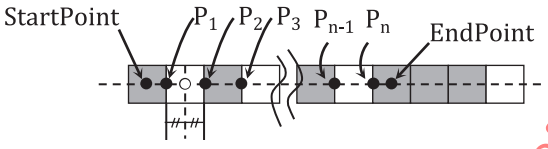


Figure 35 — The coordinates of the timing pattern

l) Search for the vertical timing pattern and corner finder pattern

Based on the number of vertical modules obtained from the format information, check the vertical separator, timing pattern and corner finder pattern adjacent to the finder pattern. As for a symbol with 7 vertical modules, the lower side of the symbol is equal to the lower side of the finder pattern. Proceed to Step m).

1) The point to start searching the timing pattern is the central coordinate [the intersection coordinate of lines drawn in f)] of the lower left corner module of the finder pattern. The point spaced Y from this point and on the line drawn in f) (a point outside of the finder pattern) is the centre point of one module contained in a separator that separates the finder pattern and data area. Although it is not a light module of the timing pattern, it is considered as a provisional light module of the timing pattern when searching so that the centre of the module of the timing pattern is adjusted to display dark and light modules alternately.

2) As for a symbol with 9 vertical modules, set the provisional central coordinate of a module spaced with 1Y from the start point for searching the timing pattern, and check that one dark module and one light module are found subsequent to the light module spaced with 1Y. As for a symbol with 11 or more vertical modules, check that 3 consecutive dark modules and 1 light module are found subsequent to the light module spaced with 1Y, 3Y, 5Y or 7Y corresponding to the number of vertical modules. The end point for search is the found dark module for a symbol with 9 vertical modules, and the first module of the 3 consecutive dark modules for a symbol with 11 or more vertical modules. Check that more than 75 % modules repeat dark and light alternation from the start to the end search points.

m) Update the vertical module dimension Y from the distance between the upper edge of the finder pattern and lower edge of the corner finder pattern (lower edge of the finder pattern for 7 vertical modules). As for 7 vertical modules, proceed to Step p).

n) Setting of the central coordinate of corner finder pattern (see Figure 36)

Set central coordinates of each module contained in the corner finder pattern in Step 1) to 4) to check the light and dark alternation.

1) Take the upper and lower end edges of the corner finder pattern on the line created in f) as AB.

2) Draw a line through the intermediate point of AB and perpendicular to Line AB and obtain CD, right and left edge of one dark module or 3 consecutive dark modules.

3) Create a grid taking the intermediate point of CD as the standard with lines spaced with 1X or 2X rightward and parallel to AB and lines spaced with 1Y upward and downward and parallel to CD. Take this intersection as the central coordinate of each module of the corner finder pattern.

4) Check that more than 75 % of the central coordinates have the dark and light assignment consistent with that for the corner finder pattern.

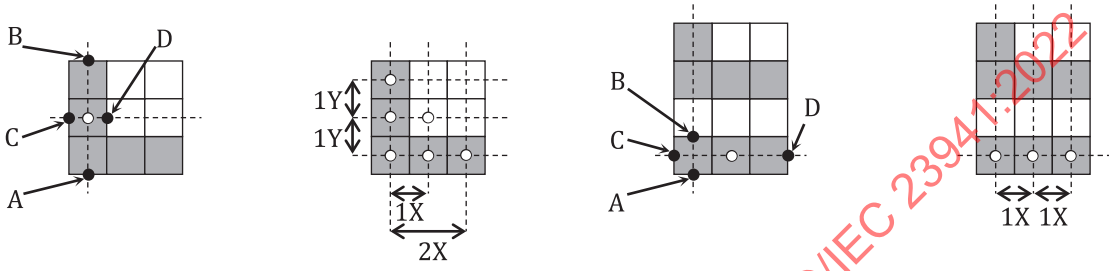


Figure 36 — corner finder pattern

o) Setting of the central coordinate of the vertical timing pattern

Take the centre point of the lower left corner module of the finder pattern as the starting point and the central coordinate of the upper left dark module of the corner finder pattern as the ending point. Set the central coordinate of the vertical timing pattern in accordance with Step k). In this case, Y is used as the module dimension.

p) Check of horizontal timing pattern of the lower side

1) Draw a line from the central coordinate of the lower left module of the symbol (lower left module of the finder pattern for 7 vertical modules) parallel to the timing pattern of the upper side of the symbol obtained in i). Search the lower horizontal timing pattern subsequent to the corner finder pattern (finder pattern and separator for 7 vertical modules) according to i) 2) up to the next alignment pattern (finder sub pattern for 27 horizontal modules). At this time, the starting point of timing pattern is a point 2X apart from the central coordinate of the lower left module of the lower left corner finder pattern obtained in n) (the centre of the lower right module of the finder pattern for 7 vertical modules). The distance from the starting point to the light module immediately before the 3 consecutive dark modules (5 consecutive dark modules for 27 horizontal modules) is the same as that of the upper timing pattern for 7 vertical modules. For 27 horizontal modules, this distance is the one with 2X added to the upper timing pattern. In other cases, the distance is the one with 4X added to the upper timing pattern (see Figure 37).

2) When a horizontal timing pattern is found, update the horizontal module dimension X by dividing a distance between the left edge of the corner finder pattern (finder pattern for 7 vertical modules) on the line drawn in 1) and the left edge of the found timing pattern by the number of contained modules.

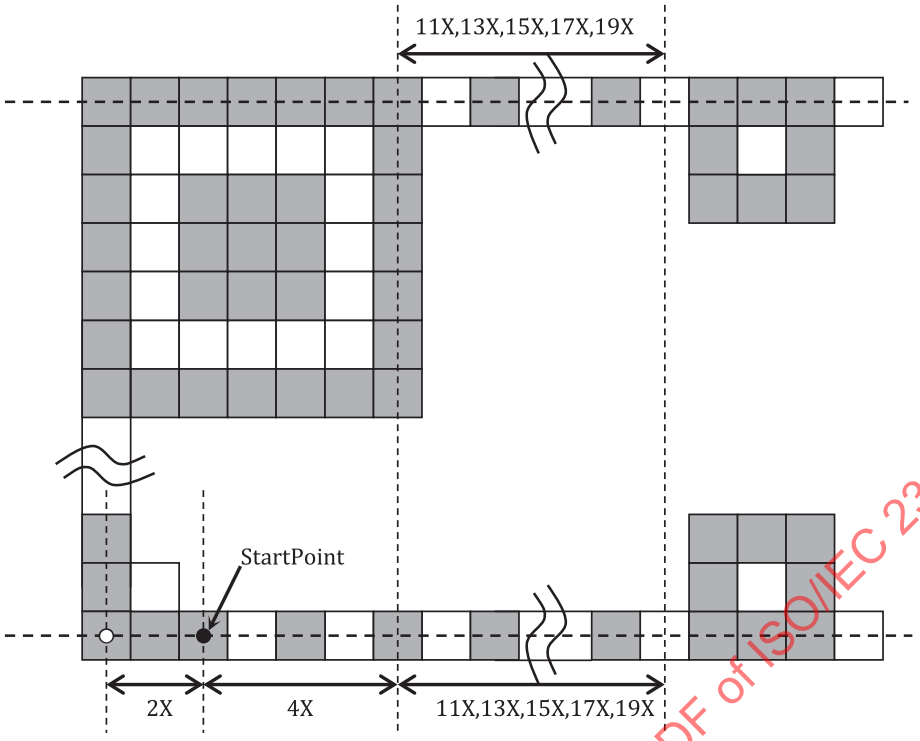


Figure 37 — Search for the horizontal timing pattern

q) If an alignment pattern subsequent to the lower horizontal timing pattern (finder sub pattern for horizontal 27 modules) is found, set each central coordinate of each module contained in the alignment pattern or the finder sub pattern through the procedures specified below to check the dark and light alternation.

1) For a symbol with other than 27 horizontal modules, set the central coordinate of the alignment pattern according to Step j) 1) and j) 3) to check the dark and light alternation.

2) For a symbol with 27 horizontal modules, set the central coordinate of the finder sub pattern according to the procedure specified below to check the dark and light alternation.

i) Take the left and right edges of the finder sub pattern on the line drawn in p) 1) as AB.

ii) Draw a line passing through the centre point of AB and perpendicular to Line AB and obtain the upper and lower edge on the dark module, CD.

iii) Take a point apart from 2Y from the centre point of CD upward as the centre point E of the finder sub pattern (see Figure 38).

iv) Draw a line passing through the centre point E and parallel to Line AB. Identify the second alternating point from dark to light when moving rightward on the line from the centre point E, and the first alternating point from light to dark when moving leftward on the line from the centre point E. Obtain the horizontal module dimension X_{sf} of the finder sub pattern by dividing the distance of these points by 4.

v) Draw a line passing through the centre point E and perpendicular to Line AB. Identify the second alternating point from dark to light when moving downward from the centre point E, and the first alternating point from light to dark when moving upward on the line from the centre point E. Obtain the vertical module dimension Y_{sf} by dividing the distance of these points by 4.

vi) Create a grid by drawing lines parallel to Line AB and $1Y_{sf}$ and $2Y_{sf}$ apart from the centre point E and lines perpendicular to Line AB and $1X_{sf}$ and $2X_{sf}$ apart from the centre point E.

Take the intersections as the central coordinates of the modules composing of the finder sub pattern.

vii) Check that more than 75 % of the central coordinates have the dark and light assignment consistent with that for the finder sub pattern.

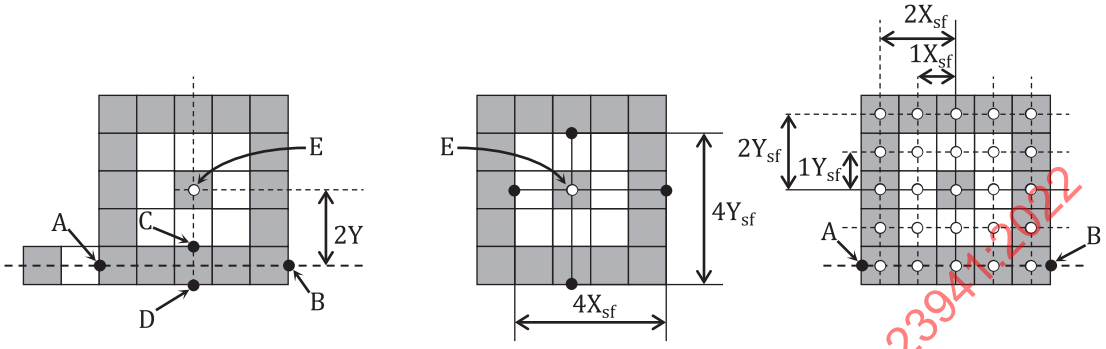


Figure 38 — Subfinder pattern

r) To set the central coordinate of the lower horizontal timing pattern, carry out Step k) 2) to k) 9) taking the starting point described in p) 1) and the central coordinate of the lower left corner module of the alignment pattern established in r) (finder sub pattern for 27 horizontal modules) as the ending point.

s) Check of the vertical timing pattern between alignment patterns (see Figure 39)

For a symbol with other than 27 horizontal modules, check the dark and light alternation of the timing pattern and set the central coordinate as described below.

1) Take the central coordinate of the centre module of the 3 consecutive dark modules overlapping the horizontal timing pattern of the upper alignment pattern as the starting point and take the central coordinate of the centre module of the 3 consecutive dark modules overlapping the horizontal timing pattern of the lower alignment pattern as the ending point. Draw a line by connecting these points.

2) Update the vertical module dimension Y by dividing a distance between the upper edge of the upper alignment pattern and the lower edge of the lower alignment pattern on this line by the number of modules.

3) Set the provisional central coordinate of the module spaced with 1Y from the starting point and judge the dark and light alternation. Check that more than 75 % of the central coordinates have the dark and light assignment consistent with that for the alignment pattern.

4) Carry out Step k) 2) to k) 9) and set the central coordinate of each module of the timing pattern.

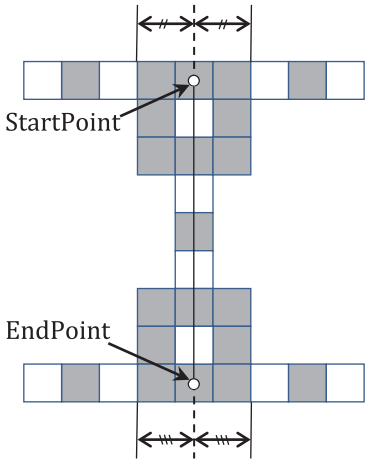


Figure 39 — Search for the vertical timing pattern

t) An area enclosed with finder pattern, corner finder pattern and/or alignment patterns obtained in a) to s) is named as small area. Four corners of the small area (see Figure 40) are a coordinate of the centre dark module of the alignment pattern obtained in j) 3) (a coordinate of the upper right module of the corner finder pattern obtained in j) 2) for a symbol with 27 horizontal modules), a coordinate of the upper left module of the finder pattern obtained in f), a coordinate of the lower left module of the corner finder pattern obtained in n) 3) (a coordinate of the lower left module of the finder pattern for a symbol with 7 vertical modules), and a coordinate of the centre dark module of the alignment pattern obtained in q) 1) (a central coordinate of the lower right module of the finder sub pattern obtained in q) 2) for a symbol with 27 horizontal modules). For a symbol with 27 horizontal modules, proceed to Step w) and for a symbol with 43 horizontal modules, proceed to Step v).

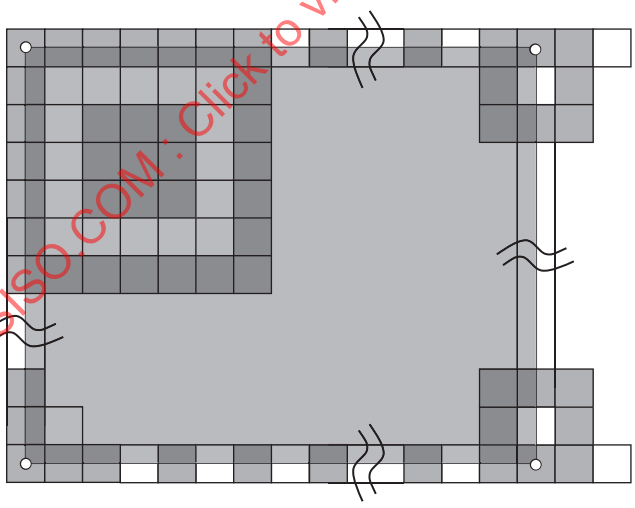


Figure 40 — small area

u) For a symbol with 59, 77, 99, or 139 horizontal modules, small areas with 4 alignment patterns at each corner are situated in the centre of the code. Search these small areas as described below.

1) To search the upper horizontal timing pattern of the small area, extend a line connecting the central coordinates of the upper left and the upper right modules of the adjacent left small area that has already been obtained.

2) Update the standard module size X by dividing the distance between the central coordinates of upper left and upper right modules of the adjacent left small area by the number of modules.

3) Search the horizontal timing pattern of the upper side by taking a point 1X apart from the centre of the upper right module of the adjacent left small area and on the line drawn in u) 1) as the starting point, based on the number of the horizontal module information obtained from the format information in the procedures specified in i) 2) along the line drawn in u) 1). Set the provisional central coordinate of the alignment pattern in accordance with j) 1) and j) 3).

4) Set the coordinate of the horizontal timing pattern in accordance with Step k) and check that more than 75 % of the central coordinates between the start and end points for searching have the dark and light assignment consistent with that for the timing pattern.

5) To search the horizontal timing pattern on the lower side of the small area, extend a line connecting the coordinates of the lower left and lower right modules of the adjacent left small area that has already been obtained.

6) Update the standard module size X by dividing the distance between the central coordinate of the lower left and lower right modules of the adjacent left small area by the number of modules.

7) Search the horizontal timing pattern of the lower side by taking a point 1X apart from the centre of the lower right module of the adjacent left small area and on the line drawn in u) 5) as the starting point, based on the number of the horizontal module information obtained from the format information in the procedures specified in i) 2) along the line drawn in u) 5). Set the provisional central coordinate of the alignment pattern in accordance with j) 1) and j) 3).

8) Set the coordinate of the horizontal timing pattern in accordance with Step k) and check that more than 75 % of the central coordinates between the start and end points for searching have the dark and light assignment consistent with that for the timing pattern.

9) Check the timing pattern between upper and lower alignment patterns and set the central coordinate in accordance with Step s). Update the module dimension Y by dividing a distance between the upper edge of the upper alignment pattern and the lower edge of the lower alignment pattern by the number of modules.

v) The small area at the rightmost code symbol contains the corner finder pattern and finder sub pattern but does not contain alignment pattern. Check the right side of the symbol as described below.

1) Check the upper right corner finder pattern and set the central coordinate as described in j) 2) i) and j) 2) ii).

2) Check the finder sub pattern and set the central coordinate as descried in q) 2) i) – q) 2) vii).

w) Check the vertical timing pattern and set the central coordinate as described below.

1) Take the centre point of the lower right module of the corner finder pattern as the starting point.

2) Take the centre point of the upper right module of the finder sub pattern as the ending point.

3) If there is any timing pattern between the starting point and the ending point, set the central coordinate in accordance with Step s) 2) to s) 4).

x) Set the sample grids in each small area as described below.

1) Create grids by connecting central coordinates of each module in the timing patterns that correspond to upper and lower, right and left respectively.

2) If any grid cannot be created in 1), create grids by connecting central coordinates of the module of corresponding fixed pattern.

y) Sample an area of 3 x 3 image pixels and calculate average, centred 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 modules as binary 1 and light modules as binary 0.

z) Release masking of the format information by XORing the bit strings given in 7.9 and decode the format information (applying the error correction procedure given in Annex C if necessary) with the format information area on the left side of finder sub pattern. Then, check that the symbol number (and hence the version and error correction level of the symbol) is the same as the already known one.

aa) XOR the data mask pattern with the encoding area of the symbol to release the data masking and restore the symbol characters representing data and error correction codewords. This reverses the effect of the data masking process applied during the encoding procedure.

bb) Determine the symbol codewords in accordance with the placement rules in 7.7.3.

1) Rearrange the codeword sequence into blocks as required for the symbol version and error correction level, by reversing the interleaving process defined in 7.6. Step 3).

2) Shall 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.

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

4) 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.

5) Decode each segment according to the rules for the mode in force.

cc) Calculate the module size of the upper side of the symbol from the distance of the central coordinates between left and right upper modules. Calculate the lower, right, and left side module sizes of a symbol in the same procedure and take the average value of upper, lower, right and left sides as the representative module size.

12 Auto-discrimination capability

rMQR can be used in an auto-discrimination environment with a number of other symbologies (see Annex K).

13 Transmitted data

13.1 General principles

All encoded data characters shall be included in the data transmission. The function patterns, format information, error correction characters, remainder codewords shall not be transmitted. The default transmission mode for all data shall be as bytes (8-bit coded value or 16-bit shift coded value). More complex interpretations including the transmission of data in an extended channel interpretation, are addressed below.

13.2 Symbology identifier

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. The symbology identifier and option values of Annex E shall apply to rMQR.

13.3 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 backslash character "\" in ISO/IEC 8859-1 and in the

AIM ECI specification and maps to the character "¥" in JIS X 0201). 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:

— the end of the encoded data;

— a change to a new ECI signaled by mode indicator 111, 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 $5C_{HEX}$ 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

a) Encoded data (hex): 41 42 43 5C 31 32 33 34

Transmitted data: 41 42 43 5C 31 32 33 34

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

Transmitted data: 41 42 43 5C 31 32 33 34 35 36 <further data>

EXAMPLE 2 (using data in 7.4.2.2)

The message contains ECI mode indicator/ECI Designator/mode indicator/character count indicator/data in the form of

111 00001001 011 00000101 10100001 10100010 10100011 10100100 10100101

Symbology Identifier]Q2 (see Annex E) shall 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: A B Γ Δ E

NOTE C_{HEX} is equivalent to the backslash character "\" in ISO/IEC 8859-1 and to "¥" in JIS X 0201.

13.4 FNC1

In the modes with implied FNC1 in either first or second position, this implied character cannot be transmitted directly as there is no byte 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 defined in Annex E shall be used). 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 byte mode by the character $^{G}_{S}$ (ASCII/JIS8 value $1D_{HEX}$). In both cases the decoder shall transmit $1D_{HEX}$.

If, in symbols in FNC1 mode, the character % needs to be encoded as data while in alphanumeric mode, it shall be 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.

Table A.1 shows the generator polynomials for the error correction codes which are used for each Version and Level, for all rMQR symbols. The number of error correction codewords required for a particular version and error correction level can be obtained from Table 8. In the Table A.1, α is the primitive element 2 under GF(2^8). Each generator polynomial is the product of the first-degree polynomials: x-α^0, x-α^1, ..., x-α^{n-1}; where n is the degree of the generator polynomial.

Table A.1 — Generator polynomials for Reed-Solomon error correction codewords

Number of error correction codewords	Generator polynomials
7	$x^7 + \alpha^{87}x^6 + \alpha^{229}x^5 + \alpha^{146}x^4 + \alpha^{149}x^3 + \alpha^{238}x^2 + \alpha^{102}x + \alpha^{21}$
8	$x^8 + \alpha^{175}x^7 + \alpha^{238}x^6 + \alpha^{208}x^5 + \alpha^{249}x^4 + \alpha^{215}x^3 + \alpha^{252}x^2 + \alpha^{196}x + \alpha^{28}$
9	$x^9 + \alpha^{95}x^8 + \alpha^{246}x^7 + \alpha^{137}x^6 + \alpha^{231}x^5 + \alpha^{235}x^4 + \alpha^{149}x^3 + \alpha^{11}x^2 + \alpha^{123}x + \alpha^{36}$
10	$x^{10} + \alpha^{251}x^9 + \alpha^{67}x^8 + \alpha^{46}x^7 + \alpha^{61}x^6 + \alpha^{118}x^5 + \alpha^{70}x^4 + \alpha^{64}x^3 + \alpha^{94}x^2 + \alpha^{32}x + \alpha^{45}$
12	$x^{12} + \alpha^{102}x^{11} + \alpha^{43}x^{10} + \alpha^{98}x^9 + \alpha^{121}x^8 + \alpha^{187}x^7 + \alpha^{113}x^6 + \alpha^{198}x^5 + \alpha^{143}x^4 + \alpha^{131}x^3 + \alpha^{87}x^2 + \alpha^{157}x + \alpha^{66}$
14	$x^{14} + \alpha^{199}x^{13} + \alpha^{249}x^{12} + \alpha^{155}x^{11} + \alpha^{48}x^{10} + \alpha^{190}x^9 + \alpha^{124}x^8 + \alpha^{218}x^7 + \alpha^{137}x^6 + \alpha^{216}x^5 + \alpha^{87}x^4 + \alpha^{207}x^3 + \alpha^{59}x^2 + \alpha^{22}x + \alpha^{91}$
16	$x^{16} + \alpha^{120}x^{15} + \alpha^{104}x^{14} + \alpha^{107}x^{13} + \alpha^{109}x^{12} + \alpha^{102}x^{11} + \alpha^{161}x^{10} + \alpha^{76}x^9 + \alpha^{3}x^8 + \alpha^{91}x^7 + \alpha^{191}x^6 + \alpha^{147}x^5 + \alpha^{169}x^4 + \alpha^{182}x^3 + \alpha^{194}x^2 + \alpha^{225}x + \alpha^{120}$
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	$x^{22} + \alpha^{210}x^{21} + \alpha^{171}x^{20} + \alpha^{247}x^{19} + \alpha^{242}x^{18} + \alpha^{93}x^{17} + \alpha^{230}x^{16} + \alpha^{14}x^{15} + \alpha^{109}x^{14} + \alpha^{221}x^{13} + \alpha^{53}x^{12} + \alpha^{200}x^{11} + \alpha^{74}x^{10} + \alpha^{8}x^9 + \alpha^{172}x^8 + \alpha^{98}x^7 + \alpha^{80}x^6 + \alpha^{219}x^5 + \alpha^{134}x^4 + \alpha^{160}x^3 + \alpha^{105}x^2 + \alpha^{165}x + \alpha^{231}$
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	$x^{28} + \alpha^{168}x^{27} + \alpha^{223}x^{26} + \alpha^{200}x^{25} + \alpha^{104}x^{24} + \alpha^{224}x^{23} + \alpha^{234}x^{22} + \alpha^{108}x^{21} + \alpha^{180}x^{20} + \alpha^{110}x^{19} + \alpha^{190}x^{18} + \alpha^{195}x^{17} + \alpha^{147}x^{16} + \alpha^{205}x^{15} + \alpha^{27}x^{14} + \alpha^{232}x^{13} + \alpha^{201}x^{12} + \alpha^{21}x^{11} + \alpha^{43}x^{10} + \alpha^{245}x^9 + \alpha^{87}x^8 + \alpha^{42}x^7 + \alpha^{195}x^6 + \alpha^{212}x^5 + \alpha^{119}x^4 + \alpha^{242}x^3 + \alpha^{37}x^2 + \alpha^{9}x + \alpha^{123}$

Table A.1 *(continued)*

Number of error correction codewords	Generator polynomials
30	$x^{30} + \alpha^{41}x^{29} + \alpha^{173}x^{28} + \alpha^{145}x^{27} + \alpha^{152}x^{26} + \alpha^{216}x^{25} + \alpha^{31}x^{24} + \alpha^{179}x^{23} + \alpha^{182}x^{22} + \alpha^{50}x^{21} + \alpha^{48}x^{20} + \alpha^{110}x^{19} + \alpha^{86}x^{18} + \alpha^{239}x^{17} + \alpha^{96}x^{16} + \alpha^{222}x^{15} + \alpha^{125}x^{14} + \alpha^{42}x^{13} + \alpha^{173}x^{12} + \alpha^{226}x^{11} + \alpha^{193}x^{10} + \alpha^{224}x^{9} + \alpha^{130}x^{8} + \alpha^{156}x^{7} + \alpha^{37}x^{6} + \alpha^{251}x^{5} + \alpha^{216}x^{4} + \alpha^{238}x^{3} + \alpha^{40}x^{2} + \alpha^{192}x + \alpha^{180}$

Annex B
(normative)

Error correction decoding steps

Take the Version R7x43-H symbol as an example. For the symbol, the (13, 3, 5) Reed-Solomon code under GF(2^8) is used for error correction. Provided that the code after releasing data masking from the symbol is:

$$R=(r_0 , r_1 , r_2 , \dots , r_{12})$$

That is,

$$R(x)=r_0 + r_1x + r_2x^2 + \dots + r_{12}x^{12}$$

$r_i(i=0\text{-}12)$ is an element of GF(2^8)

a) Calculate n syndromes.

Find the syndrome $S_i(i=0\text{-}9)$.

$$S_0 = R(1) = r_0 + r_1 + r_2 + \dots + r_{12}$$

$$S_1 = R(\alpha) = r_0+r_1\alpha+r_2\alpha^2 + \dots + r_{12}\alpha^{12}$$

...

...

$$S_9 = R(\alpha^9) = r_0 + r_1\alpha^9 + r_2\alpha^{18} + \dots + r_{12}\alpha^{108}$$

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

b) Find the error positions:

$$S_0\sigma_5 - S_1\sigma_4 + S_2\sigma_3 - S_3\sigma_2 + S_4\sigma_1 - S_5 = 0$$

$$S_1\sigma_5 - S_2\sigma_4 + S_3\sigma_3 - S_4\sigma_2 + S_5\sigma_1 - S_6 = 0$$

$$S_2\sigma_5 - S_3\sigma_4 + S_4\sigma_3 - S_5\sigma_2 + S_6\sigma_1 - S_7 = 0$$

$$S_3\sigma_5 - S_4\sigma_4 + S_5\sigma_3 - S_6\sigma_2 + S_7\sigma_1 - S_8 = 0$$

$$S_4\sigma_5 - S_5\sigma_4 + S_6\sigma_3 - S_7\sigma_2 + S_8\sigma_1 - S_9 = 0$$

Find the variable $\sigma_i(i = 1\text{-}5)$ 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.

$$\sigma(x) = \sigma_5 + \sigma_4x + \sigma_3x^2 + \sigma_2x^3 + \sigma_1x^4 + x^5$$

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

$$\sigma(\alpha) = 0.$$

c) Find the error size.

Supposing that an error is on the j^1, j^2, j^3, j^4, j^5 digits in (ii) above, then find the size of the error.

$$Y_1\alpha j^1 + Y_2\alpha j^2 + Y_3\alpha j^3 + Y_4\alpha j^4 + Y_5\alpha j^5 = S_0$$

$$Y_1\alpha^2 j^1 + Y_2\alpha^2 j^2 + Y_3\alpha^2 j^3 + Y_4\alpha^2 j^4 + Y_5\alpha^2 j^5 = S_1$$

$$Y_1\alpha^3 j^1 + Y_2\alpha^3 j^2 + Y_3\alpha^3 j^3 + Y_4\alpha^3 j^4 + Y_5\alpha^3 j^5 = S_2$$

$$Y_1\alpha^4 j^1 + Y_2\alpha^4 j^2 + Y_3\alpha^4 j^3 + Y_4\alpha^4 j^4 + Y_5\alpha^4 j^5 = S_3$$

$$Y_1\alpha^5 j^1 + Y_2\alpha^5 j^2 + Y_3\alpha^5 j^3 + Y_4\alpha^5 j^4 + Y_5\alpha^5 j^5 = S_4$$

Solve the above equations to find the size of each error $Y_i (i = 1\text{-}5)$.

d) Correct the error.

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

Annex C
(normative)

Format information

C.1 General

The format information consists of an 18-bit sequence comprising 6 data bits and 12 Extended BCH error correction bits. This annex describes the calculation of the error correction bits and the error correction decoding process.

C.2 Error correction bit calculation

The Extended 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) Extended BCH code string. Finally, masking shall be applied by XORing the bit string with **011111101010110010** (for format information at the finder pattern side) or **100000101001111011** (for format information at the finder sub pattern side) to ensure that the format information bit pattern is not all zeroes for any combination of version indicator and error correction level.

EXAMPLE (Format information at the finder pattern side)

Error correction level M;	Version R9x77
Binary string:	**000111**
Polynomial:	$\mathbf{x^2 + x + 1}$
Raise power to the (18 - 6) th:	$\mathbf{x^{14} + x^{13} + x^{12}}$
Divide by G(x):	$\mathbf{= (x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^5 + x^2 + 1)\ x^2 + (x^{11} + x^{10} + x^7 + x^4 + x^2)}$
Add coefficient string of above remainder polynomial to format information data string:	
000111+ 110010010100→	**000111110010010100**
XOR with mask	**011111101010110010**
Result:	**011000011000100110**

Place these bits in the format information areas as described in 7.9.

C.3 Error correction decoding steps

Release the masking of the format information modules by XORing the bit sequence with the mask pattern **011111101010110010** (for format information at the finder pattern side) or **100000101001111011** (for format information at the finder sub pattern side).

The Hamming distance of the error correction code used in the format information is 8, which enables up to 3 bit errors to be corrected. There are 64 valid bit sequences for the format information, so decoding by using Table C.1 as a look-up table is efficient. Bit sequences read from the format information area of the symbol are compared with the 64 valid format information bit strings in Table C.1 on a bit by bit basis. The bit string from Table C.1 closest to the bit string read from the symbol is taken, provided the strings differ by 3 bits or less.

EXAMPLE

Bit string read from format information area: **101<u>0</u>000110011<u>1</u>10101**

Closest bit string from table: **101<u>1</u>000110010<u>1</u>0101**

Since only 2 bits differ between the two bit strings, the comparison is successful, so the symbol format is confirmed as utilising error correction level H with Version R13x77.

Table C.1 — Valid format information bit sequences

Sequence before masking		Sequence after masking (Finder pattern side)		Sequence after masking (Finder sub pattern side)	
Data bits	Error correction bits	Binary	Hex	Binary	Hex
000000	000000000000	011111101010110010	1FAB2	100000101001111011	20A7B
000001	111100100101	011110010110010111	1E597	100001010101011110	2155E
000010	000101101111	011101101111011101	1DBDD	100010101100010100	22B14
000011	111001001010	011100010011111000	1C4F8	100011010000110001	23431
000100	001011011110	011011100001101100	1B86C	100100100010100101	248A5
000101	110111111011	011010011101001001	1A749	100101011110000000	25780
000110	001110110001	011001100100000011	19903	100110100111001010	269CA
000111	110010010100	011000011000100110	18626	100111011011101111	276EF
001000	010110111100	010111111100001110	17F0E	101000111111000111	28FC7
001001	101010011001	010110000000101011	1602B	101001000011100010	290E2
001010	010011010011	010101111001100001	15E61	101010111010101000	2AEA8
001011	101111110110	010100000101000100	14144	101011000110001101	2B18D
001100	011101100010	010011110111010000	13DD0	101100110100011001	2CD19
001101	100001000111	010010001011110101	122F5	101101001000111100	2D23C
001110	011000001101	010001110010111111	11CBF	101110110001110110	2EC76
001111	100100101000	010000001110011010	1039A	101111001101010011	2F353
010000	101101111000	001111000111001010	0F1CA	110000000100000011	30103
010001	010001011101	001110111011101111	0EEEF	110001111000100110	31E26
010010	101000010111	001101000010100101	0D0A5	110010000001101100	3206C
010011	010100110010	001100111110000000	0CF80	110011111101001001	33F49
010100	100110100110	001011001100010100	0B314	110100001111011101	343DD
010101	011010000011	001010110000110001	0AC31	110101110011111000	35CF8
010110	100011001001	001001001001111011	0927B	110110001010110010	362B2
010111	011111101100	001000110101011110	08D5E	110111110110010111	37D97
011000	111011000100	000111010001110110	07476	111000010010111111	384BF
011001	000111100001	000110101101010011	06B53	111001101110011010	39B9A
011010	111110101011	000101010100011001	05519	111010010111010000	3A5D0
011011	000010001110	000100101000111100	04A3C	111011101011110101	3BAF5
011100	110000011010	000011011010101000	036A8	111100011001100001	3C661
011101	001100111111	000010100110001101	0298D	111101100101000100	3D944
011110	110101110101	000001011111000111	017C7	111110011100001110	3E70E
011111	001001010000	000000100011100010	008E2	111111100000101011	3F82B
100000	100111010101	111111001101100111	3F367	000000001110101110	003AE
100001	011011110000	111110110001000010	3EC42	000001110010001011	01C8B
100010	100010111010	111101001000001000	3D208	000010001011000001	022C1

Table C.1 *(continued)*

Sequence before masking		Sequence after masking (Finder pattern side)		Sequence after masking (Finder sub pattern side)	
Data bits	Error correction bits	Binary	Hex	Binary	Hex
100011	011110011111	111100110100101101	3CD2D	000011110111100100	03DE4
100100	101100001011	111011000110111001	3B1B9	000100000101110000	04170
100101	010000101110	111010111010011100	3AE9C	000101111001010101	05E55
100110	101001100100	111001000011010110	390D6	000110000000011111	0601F
100111	010101000001	111000111111110011	38FF3	000111111100111010	07F3A
101000	110001101001	110111011011011011	376DB	001000011000010010	08612
101001	001101001100	110110100111111110	369FE	001001100100110111	09937
101010	110100000110	110101011110110100	357B4	001010011101111101	0A77D
101011	001000100011	110100100010010001	34891	001011100001011000	0B858
101100	111010110111	110011010000000101	33405	001100010011001100	0C4CC
101101	000110010010	110010101100100000	32B20	001101101111101001	0DBE9
101110	111111011000	110001010101101010	3156A	001110010110100011	0E5A3
101111	000011111101	110000101001001111	30A4F	001111101010000110	0FA86
110000	001010101101	101111100000011111	2F81F	010000100011010110	108D6
110001	110110001000	101110011100111010	2E73A	010001011111110011	117F3
110010	001111000010	101101100101110000	2D970	010010100110111001	129B9
110011	110011100111	101100011001010101	2C655	010011011010011100	1369C
110100	000001110011	101011101011000001	2BAC1	010100101000001000	14A08
110101	111101010110	101010010111100100	2A5E4	010101010100101101	1552D
110110	000100011100	101001101110101100	29BAE	010110101101100111	16B67
110111	111000111001	101000010010001011	2848B	010111010001000010	17442
111000	011100010001	100111110110100011	27DA3	011000110101101010	18D6A
111001	100000110100	100110001010000110	26286	011001001001001111	1924F
111010	011001111110	100101110011001100	25CCC	011010110000000101	1AC05
111011	100101011011	100100001111101001	243E9	011011001100100000	1B320
111100	010111001111	100011111101111101	23F7D	011100111110110100	1CFB4
111101	101011101010	100010000001011000	22058	011101000010010001	1D091
111110	010010100000	100001111000010010	21E12	011110111011011011	1EEDB
111111	101110000101	100000000100110111	20137	011111000111111110	1F1FE

Annex D
(normative)

Position of alignment patterns

The alignment patterns are positioned symmetrically on the same column of the top and the bottom rows so that it is overlapped with a timing pattern. The number and position of alignment patterns are common to each number of horizontal modules.

Table D.1 below shows, for each horizontal module, the number of alignment patterns and the column coordinates, starting at zero, of the centre module of each alignment pattern.

Table D.1 — Column coordinates of centre module of alignment patterns

Number of horizontal modules	Number of alignment patterns	Column coordinates of centre module			
27	0	—	—	—	—
43	2	21	—	—	—
59	4	19	39	—	—
77	4	25	51	—	—
99	6	23	49	75	—
139	8	27	55	83	111

Annex E
(normative)

Symbology identifier

The symbology identifier assigned to rMQR, which should be added as a preamble to the decoded data by a suitably programmed decoder, is as shown below:

]Qm

where

] is the symbology identifier flag (ASCII value 93);
Q is the code character for the rMQR symbology;
m is the modifier character with one of the values defined in Table E.1.

Table E.1 — Symbology identifier options and modifier values

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

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

Annex F
(normative)

rMQR print quality – symbology – specific aspects

F.1 General

Because of differences in symbology structures and reference decode algorithms, the effect of certain parameters on a symbol's reading performance may vary. ISO/IEC 15415 provides for symbology specifications to define the grading of certain symbology-specific attributes. This annex therefore defines the method of grading fixed pattern damage and additional parameters (format information) to be used in the application of ISO/IEC 15415 to rMQR.

F.2 Fixed pattern damage

F.2.1 Features to be assessed

The features to be assessed are:

a) upper left corner segment, including:

   1) finder pattern;

   2) the 1X wide separators surrounding the one or two inner sides of the finder pattern;

   3) part of the quiet zone of a minimum of two modules width (or more if specified by the application);

b) lower right corner segment, including:

   1) finder sub pattern;

   2) part of the quiet zone of a minimum of two modules width (or more if specified by the application);

c) upper right and lower left corner segments, each including:

   1) corner finder pattern;

   2) part of the quiet zone of a minimum of two module width (or more if specified by the application) adjacent to the corner finder pattern;

   3) vertical timing pattern of alternating dark and light modules adjacent to the corner finder pattern;

d) alignment pattern and part of the quiet zone of a minimum of two module width (or more if specified by the application) adjacent to the alignment pattern;

e) the horizontal timing patterns of alternating dark and light modules passing through the sides from the finder pattern or the finder sub pattern rightward/leftward. The vertical timing pattern between alignment patterns;

The features listed above shall be assessed as four (A, B1, B2, B3, C, D1, D2) segments (see Figure F.1 and F.2), viz.:

a) For upper left corner (finder pattern with their associated separators and a part of surrounding quiet zone), 104 modules are applicable (However, when vertical modules are 7, 114 modules are applicable). (Segment A);

b) Unit segment containing 49 modules (Segment B1) of the lower right corner (associated quiet zone and finder sub pattern), at least 22 modules (Segment B2) of upper right corner (upper right corner finder pattern, associated quiet zone and adjacent vertical timing pattern), and at least 15 modules (Segment B3) of lower left corner (lower left corner finder pattern, associated quiet zone and adjacent vertical timing pattern) (Segment B). When the vertical modules are 7, B3 is not situated in the code symbol (see Figure F.3.) and a vertical timing pattern is not contained in B2. When the vertical modules are 7, 9, and 11, a vertical timing pattern is not contained in Segment B3 (See Figure F.3 to F.5.);

c) Unit segment containing all alignment patterns and their adjacent quiet zones (Segment C), when the horizontal modules are 27, C is not applicable;

d) Unit segment containing individual horizontal timing patterns (Segment D1) located on the periphery of code and between the finder pattern/ the finder sub pattern and the corner finder pattern and separated by the alignment patterns (if they exist), and timing patterns (Segment D2) located inside the code and between the upper and lower alignment patterns. (Segment D).

For example, Version R7x43 and R9x43 contain 114 modules and 104 modules in A segment respectively. B1 of both versions contains 49 modules, and B2 of Version R7x43 and R9x43 contain 22 modules and 23 modules respectively. (See Figure F.3 and F.4.)

Version R7x43 has no lower left corner finder pattern. Version R9x43 does not have L shaped lower left corner finder pattern, and the Segment B3 area is as shown in Figure F.4.

The range of the vertical timing pattern (Segment D2) is from the upper end and to the lower end of the alignment patterns and has the same number of modules as those for the vertical modules. (See Figure F.6.)

NOTE For rMQR symbol, its width of Quiet Zone shall be 2X. Figure F.1 to F.6 show segments that shall be checked at fixed pattern print quality assessment.

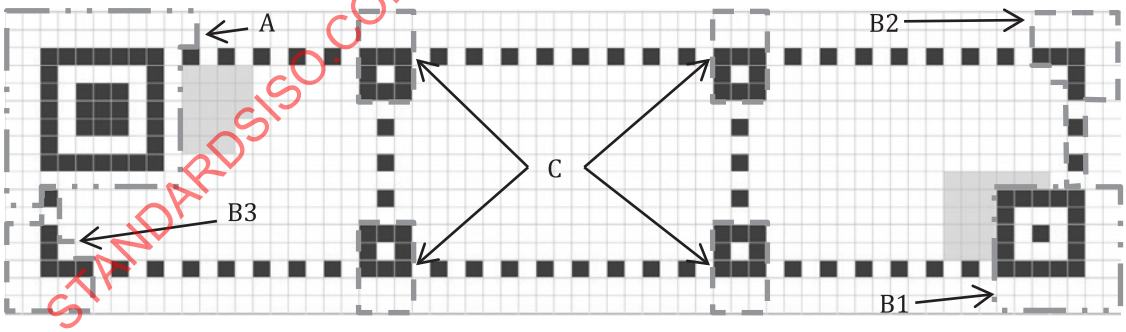


Figure F.1 — rMQR fixed pattern (Segment A, B, C) inspection area