INTERNATIONAL STANDARD

ISO/IEC 23634

First edition
2022-04

Information technology — Automatic identification and data capture techniques — JAB Code polychrome bar code symbology specification

Reference number
ISO/IEC 23634:2022(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

JAB Code is a colour-based, two-dimensional matrix symbology whose basic symbols are made up of colourful modules, arranged in either square or rectangle grids. JAB Code has two types of basic symbols: a primary symbol and the secondary symbol. A JAB Code contains one primary symbol, and optionally, multiple secondary symbols. A primary symbol contains four finder patterns, located at the corners of the symbol. Secondary symbols contain finder pattern.

A secondary symbol can be docked to a primary symbol, or another docked secondary symbol, in either a horizontal or vertical direction. JAB Code can encode from small to large amounts of data, correlated to user-specified percentages of the error correction.

Both manufacturers and users of bar code equipment require publicly available symbology standards when developing equipment and application standards. The publication of standardised symbology specifications, such as this one, are designed to achieve this.

Information technology — Automatic identification and data capture techniques — JAB Code polychrome bar code symbology specification

IMPORTANT — The electronic file of this document contains colours which are considered to be useful for the correct understanding of the document. Users should therefore consider printing this document using a colour printer.

1 Scope

This document defines the requirements for the symbology known as JAB Code. It specifies the JAB Code symbology characteristics, symbol structure, symbol dimensions, symbol cascading rules, data character encodation, error correction rules, user-selectable application parameters, print quality requirements and a reference decode algorithm.

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 646, *Information technology — ISO 7-bit coded character set for information interchange*

ISO/IEC 10646, *Information technology — Universal coded character set (UCS)*

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

ISO/IEC 15424, *Information technology — Automatic identification and data capture techniques — Data Carrier Identifiers (including Symbology Identifiers)*

ISO/IEC 15434, *Information technology — Automatic identification and data capture techniques — Syntax for high-capacity ADC media*

3 Terms, definitions, abbreviated terms and symbols

For the purposes of this document, the following terms and definitions 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 Terms and definitions

3.1.1
module
single square in one colour within the matrix pattern that is the elemental entity used to encode data

3.1.2
finder pattern
fixed reference pattern at predefined positions in a matrix symbology, which enables the decode software to locate the JAB symbol in an image

3.1.3
alignment pattern
fixed reference pattern at predefined positions in a matrix symbology, which enables the decode software to resynchronize the coordinate mapping of the *modules* (3.1.1) in the event of moderate amounts of distortion of the image

3.1.4
data interleaving
procedure which pseudo-randomly arranges the data in a matrix symbology

3.1.5
colour palette
set of reference *modules* (3.1.1) of colours used in the symbol, which is located at predefined positions in a matrix symbology

3.1.6
padding bits
bits which do not represent data and are used to fill empty positions of the available encoding capacity after the final bit of encoded data

3.1.7
primary symbol
main symbol which contains finder patterns and is used to locate the whole JAB code

3.1.8
secondary symbol
appending symbols which may be used to encode more data with a lower overhead in terms of auxiliary *modules* (3.1.1)

3.1.9
host symbol
symbol, either primary or secondary, in a JAB Code which docks secondary symbols on its horizontal or vertical sides

3.1.10
JAB Code
colour two-dimensional matrix symbology

Note 1 to entry: JAB code is used for "just another bar code".

3.2 Abbreviated terms

LDPC	low-density parity-check
SS	in primary symbol: symbol shape flag / in secondary symbol: same shape and size flag
MSK	masking reference
SE	same error correction level
V	side-version
E	error correction parameter
S	secondary positions
m	raw data bits
c	transmitted codeword

r received codeword

L number of iterations

FPDB fixed pattern damage in segment B

ECI extended channel interpretation

FNC1 function code one

GNU grid non-uniformity

UEC unused error correction

CSL centre surrounding layer

3.3 Mathematical symbols

For the purposes of this document, the following mathematical symbols apply:

N_c module colour mode indicating the number of module colours in the symbol

C symbol capacity in number of bits

P_n symbols net payload (the number of raw data bits)

P_g symbols gross payload (the number of encoded data bits)

P_e length of the encoded message including the metadata of docked secondary symbols and the flag bit

K number of error correction bits in the symbol, equal to P_g-P_n

H parity check matrix of LDPC code

w_r number of 1's in each row in H (the parity check matrix of LDPC code)

w_c number of 1's in each column in H (the parity check matrix of LDPC code)

3.4 Mathematical and logical operations

For the purposes of this document, the following mathematical and logic operations apply.

max(x,y) is the greater of x and y

div is the integer division operator

mod is the remainder after division

XOR is the exclusive-or logic function that outputs one only when the two inputs differ.

$\log_2(x)$ is the logarithm function to base 2.

ln(x) is the logarithm function to base e.

4 Symbol description

4.1 Basic characteristics

a) Encodable character set:

1) numeric data;

2) uppercase letters;

3) lowercase letters;

4) punctuation marks;

5) mixed characters;

6) alphanumeric data;

7) byte data (default interpretation: UTF-8 specified in ISO/IEC 10646);

8) ECI and FNC1.

b) Symbol type

1) In JAB Code there are two types of symbols: primary symbol and secondary symbol.

2) A JAB Code contains one primary symbol and optionally multiple secondary symbols.

c) Symbol shape

1) The primary symbol and secondary symbol in a JAB Code may be either square or rectangle.

2) Primary symbol and secondary symbol in a JAB Code may be of different shapes.

d) Symbol size

1) The smallest primary or secondary JAB Code symbol side size is 21 and the largest is 145. The smallest square symbol is 21 × 21 modules and the largest is 145 × 145 modules.

2) No quiet zone is required for the symbol.

e) Module colour

1) The number of module colours is configurable in two modes: 4 or 8 colours.

2) Guidelines for colour selection are given in Annex G.

f) Representation of data

1) A module represents $\log_2(N_c)$ binary bits. See 5.7.

2) The binary bits that a module represents correspond to the index value of the module colour in the colour palette.

g) Data capacity

1) The data capacity of JAB Code depends on the symbol size, the number of module colours, and the error correction level.

2) The capacity of a single-symbol square code is listed in Table 1.

h) Selectable error correction

1) User-selectable error correction levels are supported.

2) In one JAB Code, different error correction levels may be configured in each symbol.

i) Symbol cascading

1) Secondary symbols can be docked to the side of a primary symbol, or other secondary symbols.

2) JAB Code may have an arbitrary form by cascading primary and secondary symbols in horizontal and vertical directions while adhering to the order in Figure 14.

j) Code type: Matrix

k) Orientation independent: Yes

4.2 Summary of additional features

The use of the following additional features is optional in JAB Code:

a) Mirror Imaging: When JAB Code is obtained in mirror reversal, it is still possible to achieve a valid decode of a symbol with the standard reader. Refer to 6.2.

b) Extended Channel Interpretation: The ECI mechanism enables data using character sets other than the default encodable set (e.g., Arabic, Chinese, Cyrillic, Greek, Hebrew) and other data interpretations or industry-specific requirements to be represented.

4.3 Symbol structure

4.3.1 Square primary symbol

The structure of a square JAB Code primary symbol is shown in Figure 1 and given in Annex A. A square primary symbol shall consist of finder pattern, alignment pattern (starting with Side-Version 6, and larger), colour palette, encoded data and optionally metadata region. Four finder patterns are located at the four symbol corners respectively, with one module between the outermost layer and the border. No quiet zone surrounding the symbol is required. The primary symbol illustrated in Figure 1 is a square symbol of Side-Version 2, whose width and height are 25 modules.

4.3.2 Rectangle primary symbol

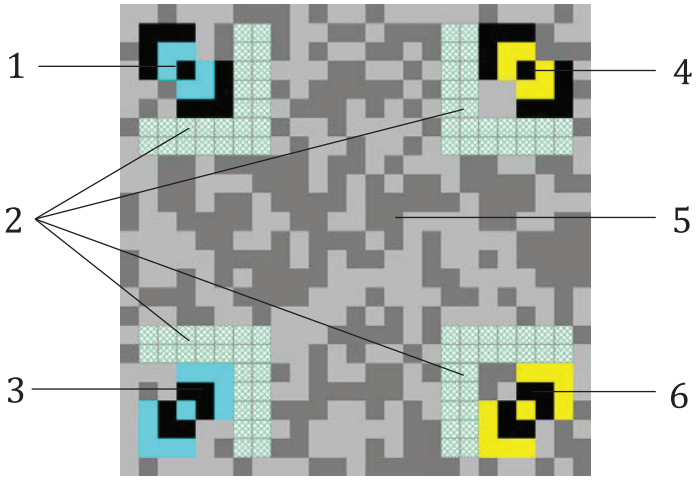
The structure of a rectangle JAB Code primary symbol is shown in Figure 2. The structure of a rectangle primary symbol is the same as a square primary symbol, except that the horizontal and vertical distance between the finder patterns are not equal. Like square symbols, no quiet zone is required for rectangle primary symbols. The primary symbol illustrated in Figure 2 is a rectangle symbol of a combination of Side-Version 5 and 2, of which the width is 37 modules and the height is 25 modules.

4.3.3 Square secondary symbol

The structure of a square JAB Code secondary symbol is shown in Figure 3. Except for finder patterns, secondary symbols contain the same patterns as the primary symbol, including alignment pattern, colour palette and encoded data region. In secondary symbols, the four finder patterns are replaced by four alignment patterns. Like the primary symbol, no surrounding quiet zone is required for secondary symbols. The secondary symbol illustrated in Figure 3 is a square symbol of Side-Version 2, whose width and height are 25 modules.

4.3.4 Rectangle secondary symbol

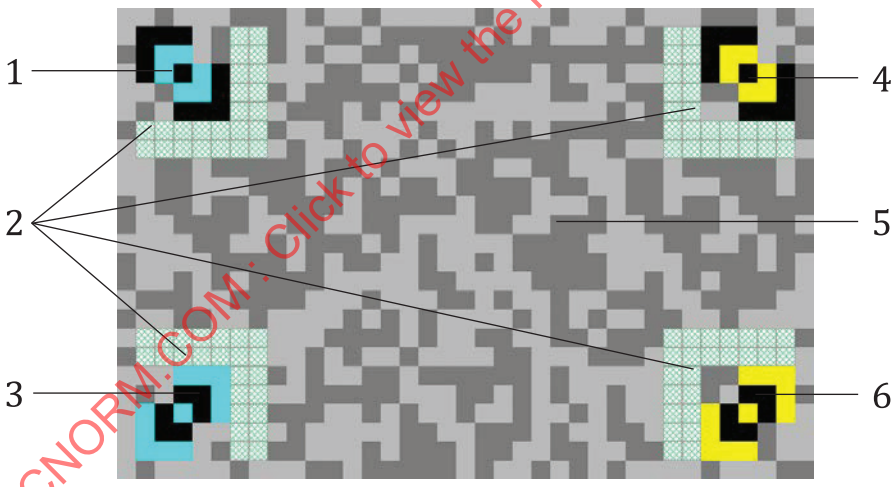
The structure of a rectangle JAB Code secondary symbol is shown in Figure 4. The structure of the rectangle secondary symbol is the same as the rectangle primary symbol, except that the finder patterns are replaced by four alignment patterns. No quiet zone is required for rectangle secondary symbols. The secondary symbol illustrated in Figure 4 is a rectangle symbol of a combination of Side-Version 5 and 2, whose width is 37 modules and whose height is 25 modules.



Key

1 Finder pattern UL
2 metadata and colour palette
3 Finder pattern LL
4 Finder pattern UR
5 encoded data
6 Finder pattern LR

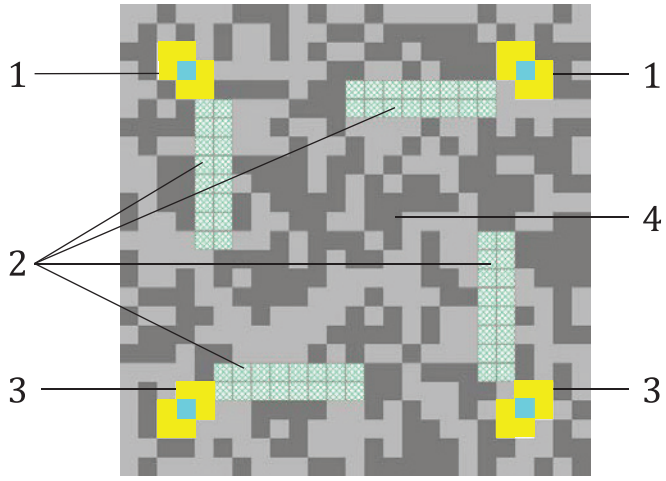
Figure 1 — Structure of square primary symbol



Key

1 Finder pattern UL
2 metadata and colour palette
3 Finder pattern LL
4 Finder pattern UR
5 encoded data
6 Finder pattern LR

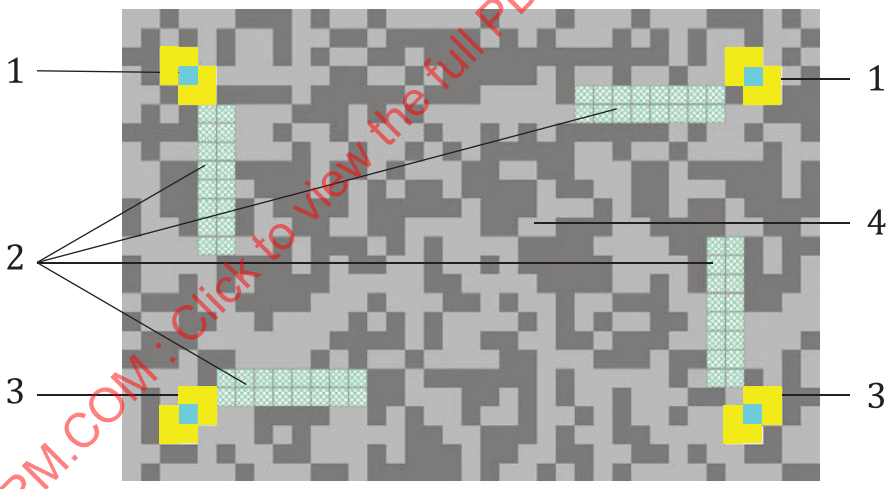
Figure 2 — Structure of rectangle primary symbol



Key

1 alignment pattern U
2 colour palette
3 alignment pattern L
4 encoded data

Figure 3 — Structure of square secondary symbol



Key

1 alignment pattern U
2 colour palette
3 alignment pattern L
4 encoded data

Figure 4 — Structure of rectangle secondary symbol

4.3.5 Symbol side size

The side of a JAB Code symbol may have one of 32 different sizes, referred to as Side-Version 1, Side-Version 2, … Side-Version 32. See Table 1. The side size increases in steps of 4 modules, from 21 modules in Side-Version 1, to 145 modules in Side-Version 32. A square symbol has the same Side-Version for both the horizontal and vertical sides, while a rectangle symbol may have any combination of two different side-versions for the horizontal and vertical sides. The smallest square symbol measures 21 × 21 modules and the largest square symbol measures 145 × 145 modules. The smallest rectangle

symbol measures 21 × 25 modules and the largest rectangle symbol measures 141 × 145 modules. The rectangle symbol of 21 × 145 or 145 × 21 modules has the maximal proportion between the horizontal and vertical sides.

The capacities listed in Table 1 are based on the recommended error correction level 3 for square symbols, and a default of 8 colours. The number of data modules can be calculated according to the following formulae:

Distance of Finder pattern Centre to Border: $DFCB = 4$

Minimum Distance between Alignment patterns: $MDBA = 16$

Number of alignment pattern modules:

$$a_x = max(0, \lfloor (SideSize_x - DFCB \times 2 + 1) / MDBA - 1 \rfloor)$$

$$a_y = max(0, \lfloor (SideSize_y - DFCB \times 2 + 1) / MDBA - 1 \rfloor)$$

$$a = ((a_x + 2) \times (a_y + 2) - 4) \times 7$$

Number of colours used in JAB code:

$$NumberOfModuleColor = \begin{cases} 4, if\ 4\ color\ selected \\ 8, if\ 8\ color\ selected \end{cases}$$

Number of colour palette modules: $C_{Palette} = (NumberOfModuleColor - 2) \times 4$

Number of finder pattern modules: $F_{Primary} = 4 \times 17; F_{Secondary} = 4 \times 7$

Number of Metadata modules:

$$Metadata = \frac{MetadataLength}{\log_2(NumberOfModuleColor)} * log_2(2)$$

Number of data modules in primary: $SideSize_x \times SideSize_y - a - C_{Palette} - F_{Primary} - Metadata$

Number of data modules in secondary: $SideSize_x \times SideSize_y - a - C_{Palette} - F_{Secondary}$

Table 1 — Symbol side versions and square symbol capacity of JAB Code (default metadata)

Side-Version	Side size (in modules)	Number of data modules				Symbol net payload P_n (in bits)			
		Square Primary		Square Secondary		Square Primary		Square Secondary	
		4	8	4	8	4	8	4	8
1	21	338	349	405	389	676	1 047	810	1 167
2	25	522	533	589	573	1 044	1 599	1 178	1 719
3	29	738	749	805	789	1 476	2 247	1 610	2 367
4	33	986	997	1 053	1 037	1 972	2 991	2 106	3 111
5	37	1 266	1 277	1 333	1 317	2 532	3 831	2 666	3 951
6	41	1 543	1 554	1 610	1 594	3 086	4 662	3 220	4 782
7	45	1 887	1 898	1 954	1 938	3 774	5 694	3 908	5 814
8	49	2 263	2 274	2 330	2 314	4 526	6 822	4 660	6 942
9	53	2 671	2 682	2 738	2 722	5 342	8 046	5 476	8 166
10	57	3 062	3 073	3 129	3 113	6 124	9 219	6 258	9 339
11	61	3 534	3 545	3 601	3 585	7 068	10 635	7 202	10 755
12	65	4 038	4 049	4 105	4 089	8 076	12 147	8 210	12 267
13	69	4 574	4 585	4 641	4 625	9 148	13 755	9 282	13 875
14	73	5 079	5 090	5 146	5 130	10 158	15 270	10 292	15 390
15	77	5 679	5 690	5 746	5 730	11 358	17 070	11 492	17 190
16	81	6 311	6 322	6 378	6 362	12 622	18 966	12 756	19 086
17	85	6 975	6 986	7 042	7 026	13 950	20 958	14 084	21 078

Table 1 *(continued)*

Side-Version	Side size (in modules)	Number of data modules				Symbol net payload P_n (in bits)			
		Square Primary		Square Secondary		Square Primary		Square Secondary	
		4	8	4	8	4	8	4	8
18	89	7 594	7 605	7 661	7 645	15 188	22 815	15 322	22 935
19	93	8 322	8 333	8 389	8 373	16 644	24 999	16 778	25 119
20	97	9 082	9 093	9 149	9 133	18 164	27 279	18 298	27 399
21	101	9 874	9 885	9 941	9 925	19 748	29 655	19 882	29 775
22	105	10 607	10 618	10 674	10 658	21 214	31 854	21 348	31 974
23	109	11 463	11 474	11 530	11 514	22 926	34 422	23 060	34 542
24	113	12 351	12 362	12 418	12 402	24 702	37 086	24 836	37 206
25	117	13 271	13 282	13 338	13 322	26 542	39 846	26 676	39 966
26	121	14 118	14 129	14 185	14 169	28 236	42 387	28 370	42 507
27	125	15 102	15 113	15 169	15 153	30 204	45 339	30 338	45 459
28	129	16 118	16 129	16 185	16 169	32 236	48 387	32 370	48 507
29	133	17 166	17 177	17 233	17 217	34 332	51 531	34 466	51 651
30	137	18 127	18 138	18 194	18 178	36 254	54 414	36 388	54 534
31	141	19 239	19 250	19 306	19 290	38 478	57 750	38 612	57 870
32	145	20 383	20 394	20 450	20 434	40 766	61 182	40 900	61 302

4.3.6 Module dimension

JAB Code symbols (both primary and secondary symbols) shall conform to the following dimensions:

X dimension: the width of a module shall be specified by the application, taking into account the technologies used to produce and scan the symbol.

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

No limit is placed on the module size in this specification. However, all modules in the symbols of a JAB Code shall be of the same size.

4.3.7 Finder pattern

There are four types of finder patterns in JAB Code, i.e., Finder Pattern UL, Finder Pattern UR, Finder Pattern LR and Finder Pattern LL, located at the upper left, the upper right, the lower right and lower left corners respectively as illustrated in Figure 6. Each finder pattern contains two equal square references made up of 3 × 3 modules as illustrated in Figure 5 for the UL finder pattern, connected with each other by an overlapping module (core module).

The finder patterns have different orientations. The core module of Finder Pattern UL and Finder Pattern UR shall be the lower right module of the upper reference, and the upper left module of the lower reference, respectively. In Finder Pattern LR and Finder Pattern LL, the core shall be the lower left module of the upper reference and the upper right module of the lower reference.

Each square reference in a finder pattern is constructed of three layers; all the same width of one module. The layers in the two references are symmetric with respect to the core module, as illustrated in Figure 5. Each finder pattern contains two colours. The colour of each layer is different from its adjacent layers. Finder Pattern UL and Finder Pattern LL consist of black and cyan layers. Finder Pattern UR and Finder Pattern LR are made up of yellow and black layers. Finder Pattern UL and UR have a black core. Finder Pattern LR has a yellow core and Finder Pattern LL has a cyan core.

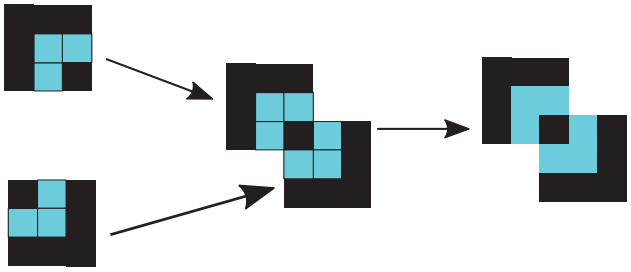
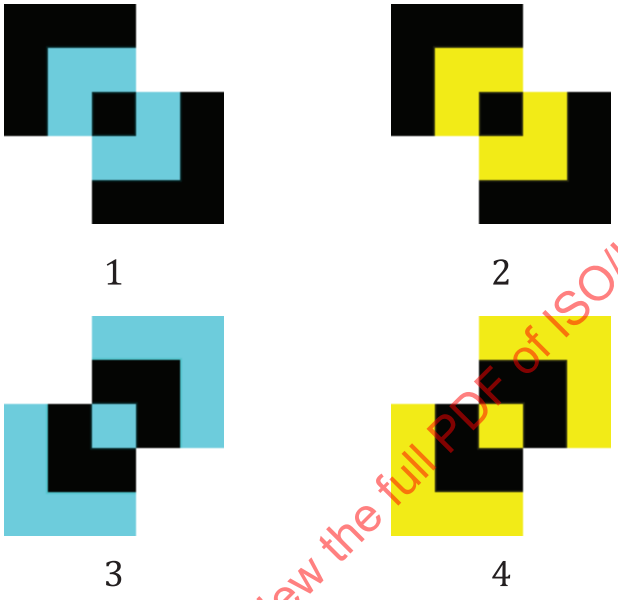


Figure 5 — Constructing the UL finder pattern



Key

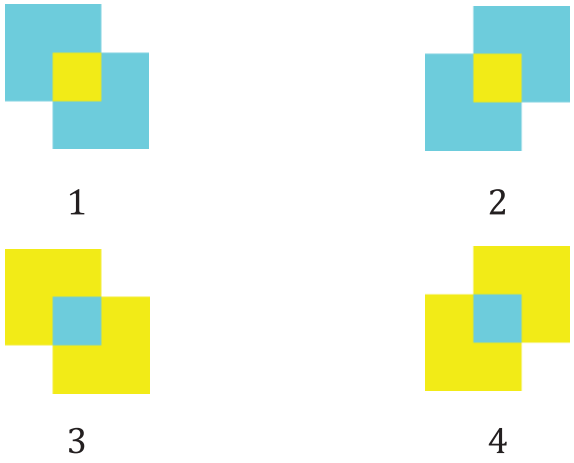
1 Finder pattern UL
2 Finder pattern UR
3 Finder pattern LL
4 Finder pattern LR

Figure 6 — Finder patterns

4.3.8 Alignment pattern

There are four types of alignment patterns in JAB Code, i.e., Alignment Pattern X0, Alignment Pattern X1, Alignment Pattern U and Alignment Pattern L, as illustrated in Figure 7. Each alignment pattern is constructed of two square references (2 × 2 modules) consisting of two layers connected by an overlapping core module. Alignment Pattern U and Alignment Pattern L have a yellow outer layer and cyan cores, Alignment Pattern X0 and Alignment Pattern X1 have a cyan outer layer and yellow cores.

Alignment Pattern U shall be placed in secondary symbols at the same positions as Finder Pattern UL and Finder Pattern UR in primary symbols. Alignment Pattern L shall be placed in secondary symbols at the same positions as Finder Pattern LR and Finder Pattern LL in primary symbols as illustrated in Figure 4. Alignment Pattern X0 and X1 shall be placed between finder patterns in primary symbols and between Alignment Pattern U and Alignment Pattern L in secondary symbols as illustrated in Figure 8.



Key

1 alignment pattern X0
2 alignment pattern X1
3 alignment pattern U
4 alignment pattern L

Figure 7 — Alignment patterns

Alignment Pattern X0 and Alignment Pattern X1 are present only in JAB Code symbols which has Side-Version 6 or larger. The number of alignment patterns depends on the side-version of each symbol side. The alignment patterns are spaced as evenly as possible. For each side-version, the number of alignment patterns and the column/row coordinates of the core module of each alignment pattern are specified in Table 2, where the coordinate of the top-left module in the symbol is defined as (1, 1).

In either primary or secondary symbols, alignment patterns shall be placed on the intersections of columns and rows of the coordinates listed in Table 2 except the positions where a finder pattern is located. For example, Table 2 indicates the coordinates 4, 14, 32 and 54 for Side-Version 10. Therefore, in a square primary symbol of Side-Version 10, the 12 alignment patterns are centred at (column, row) positions (4, 14), (4, 32), (14, 4), (14, 14), (14, 32), (14, 54), (32, 4), (32, 14), (32, 32), (32, 54), (54, 14), (54, 32).

The first alignment pattern, which is located next to Finder Pattern UL in primary symbols or next to Alignment Pattern U at the top-left corner in secondary symbols in either horizontal or vertical direction, shall be Alignment X1. At the following placement positions of alignment patterns, Alignment Pattern X0 and Alignment Pattern X1 shall be placed alternately in both horizontal and vertical directions, as illustrated in Figure 8.

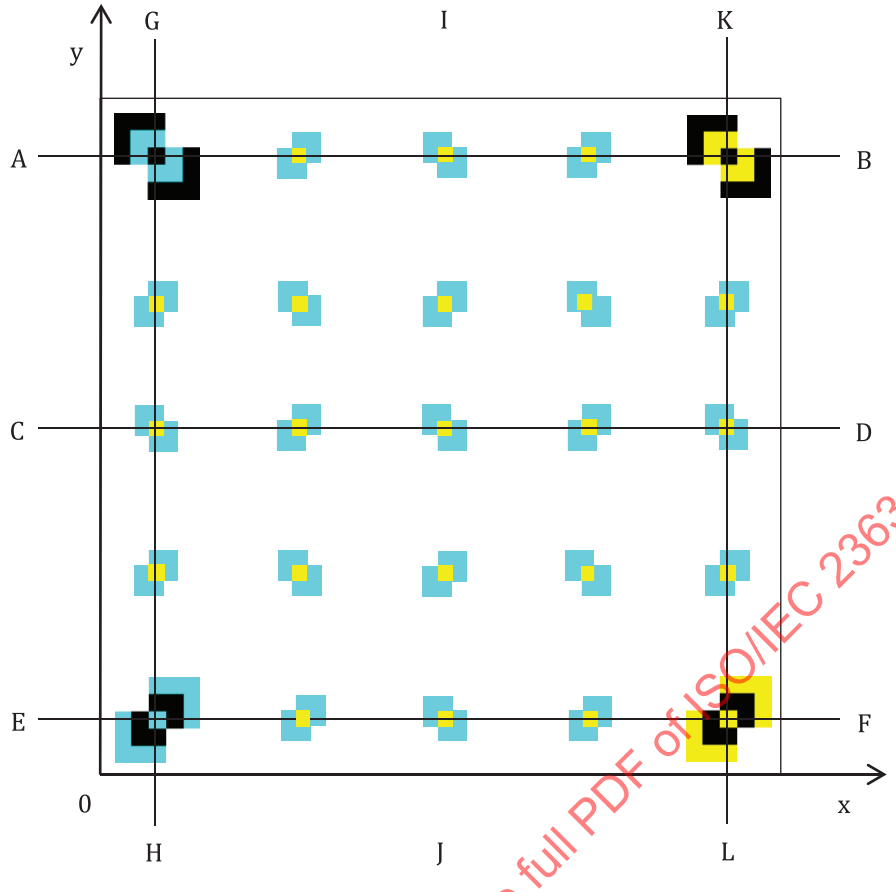


Figure 8 — Alignment pattern placement

Table 2 — Positions of alignment patterns

Side-Version	Side size (in modules)	Number of alignment patterns	Column/Row (x/y) coordinates of core module								
1	21	0	4	18							
2	25	0	4	22							
3	29	0	4	26							
4	33	0	4	30							
5	37	0	4	34							
6	41	5	4	17	38						
7	45	5	4	20	42						
8	49	5	4	23	46						
9	53	5	4	26	50						
10	57	12	4	14	32	54					
11	61	12	4	17	39	58					
12	65	12	4	20	46	62					
13	69	12	4	23	44	66					
14	73	21	4	26	37	51	70				
15	77	21	4	14	36	58	74				
16	81	21	4	17	39	56	78				
17	85	21	4	20	42	63	82				

Table 2 *(continued)*

Side-Version	Side size (in modules)	Number of alignment patterns	Column/Row (x/y) coordinates of core module								
18	89	32	4	23	38	54	70	86			
19	93	32	4	26	38	56	77	90			
20	97	32	4	14	33	53	72	94			
21	101	32	4	17	38	59	79	98			
22	105	45	4	20	36	53	70	86	102		
23	109	45	4	23	36	55	74	93	106		
24	113	45	4	26	36	58	79	100	110		
25	117	45	4	14	36	58	80	92	114		
26	121	60	4	17	34	52	70	88	99	118	
27	125	60	4	20	37	54	72	89	106	122	
28	129	60	4	23	38	56	74	92	113	126	
29	133	60	4	26	36	58	78	98	120	130	
30	137	77	4	14	32	49	67	84	102	112	134
31	141	77	4	17	35	53	71	89	107	119	138
32	145	77	4	20	38	55	73	91	108	126	142

4.3.9 Colour palette

The colour palette provides reference module colour values for symbol decoding. As listed in Table 6, two module colour modes are specified, allowing for minimally 4, and maximally 8 colours to be used in a symbol. Hence the colour palette has a minimal size of 4, and maximal size of 8, containing up to 8 colours, which are indexed from 0 to 7. Table 3 shows an example of an 8-colour palette, corresponding to module colour mode 2, and Table 4 shows the 4-colour palette. Refer to Annex G for guidelines on module colour selection and colour palette construction.

In either primary or secondary symbols, four colour palettes shall be placed near to the finder patterns or the alignment pattern U or L, which are located in different reserved regions. Each placed colour palette contains all available module colours except the two colours used by the nearest finder pattern or alignment pattern U or L. The module placement of colour palettes is specified in 4.4.4.

Table 3 — Colour palette for 8-colour symbols

Index	c0	c1	c2	c3	c4	c5	c6	c7
Colour	Black	Blue	Green	Cyan	Red	Magenta	Yellow	White

Table 4 — Colour palette for 4-colour symbols

Index	c0	c1	c2	c3
Colour	Black	Cyan	Magenta	Yellow

4.3.10 Metadata

The metadata defines the symbol properties, like flexible symbol encoding and decoding, including the number of module colours, the symbol shape and size, the error correction parameters, the masking type and the code structure. The metadata structure and placement which is different for primary and secondary symbols, and are defined in 4.4.

The metadata of primary symbols are encoded in the reserved modules around the four finder patterns, as defined in 4.4.4. The metadata of secondary symbols are encoded at the end of the data stream of

its host symbol. The length of metadata in both primary and secondary symbols is variable, thus the number of modules needed to encode these metadata also varies.

The metadata are secured by error correction codes. See 4.4.3.

4.3.11 Encoded data

Excluding the modules used for finder patterns, alignment patterns, colour palettes and metadata, all the remaining modules shall be used to encode data, including the error correction.

The placement of encoded data is specified in 5.7.

4.4 Metadata structure

4.4.1 Metadata of a primary symbol

4.4.1.1 Metadata structure

The metadata of a primary symbol defines the number of module colours, the symbol shape, the symbol size, the masking type, the error correction level and the positions of docked secondary symbols. The metadata are divided into three parts: Part I, Part II and Part III. See Table 5. The total length of a primary symbol metadata are 26 bits.

The metadata structure of a primary symbol is shown in Table 5. Part I defines the module colour mode (N_c) indicating the number of module colours. Part II defines three variables, including symbol side-version (V), error correction parameters (E) and masking reference (MSK). Part III defines the positions of docked secondary symbols (S).

Part I and Part II are optional. If not specified by the user, the default values (as specified below) in Part I, and Part II shall be used. If the user specified any variable, Part I and Part II shall be placed in the metadata modules as specified in 4.4.4. The bit length of Part I and Part II is fixed.

Part III shall be appended to the data stream of the primary symbol as specified in 4.4.1.6 and encoded together with the data.

Table 5 — Metadata structure of a primary symbol

Metadata					
Part I		**Part II**		**Part III**	
Variable	Length (Bits)	Variable	Length (Bits)	Variable	Length (Bits)
module colour mode N_c	3	symbol side-version V	10	positions of the docked secondary symbols S	4
		error correction parameters E	6		
		masking reference MSK	3		
Sum	3	Sum	19	Sum	4
Total length: 26 bits					

4.4.1.2 Module colour mode

The variable N_c in Part I defines the module colour mode. Table 6 lists the number of the available module colours. It takes 3 bits and defines 8 colour modes. N_c shall be encoded in a three-colour mode which uses only the first, the fourth and the seventh module colour in the colour palette defined in Table 3. The three used colours shall be black, cyan and yellow in all colour modes. After error correction

encoding is performed, N_c shall be encoded with these three colours according to Table 7. Excluding N_c, other metadata shall be encoded in multi-colour mode using all available colours.

Place Part I using black, cyan and yellow	Place colour palettes	Place Part II, encoded message and Part III using all available colours of the selected colour mode

Colour modes 0, 3, 4, 5, 6 and 7 are reserved for future extensions. These colour modes can also be used for user-defined colour modes. See Annex G for additional guidance when using these colour modes in user-defined conditions.

If not specified, the default module colour mode, mode 2, shall be used.

Table 6 — Part I module colour modes of a primary symbol

Variable	Value (binary)	Colour mode	Number of module colours
N_c	000	0	reserved
	001	1	4
	010 (default)	2	8
	011	3	reserved
	100	4	reserved
	101	5	reserved
	110	6	reserved
	111	7	reserved

Table 7 — Module colour representing bit sequence for N_c

Bit sequence	First module colour	Second module colour
000	black	black
001	black	cyan
010	black	yellow
011	cyan	black
100	cyan	cyan
101	cyan	yellow
110	yellow	black
111	yellow	cyan

4.4.1.3 Symbol size

The symbol size is specified by the side-version value V that indicates the version number. The length of V is 10 bits as shown in Table 5, Table 8 and Table 9, where the horizontal and vertical side-versions shall be encoded respectively. The first 5 bits encode the horizontal side-version and the last 5 bits encode the vertical side-version.

4.4.1.4 Error correction level

The error correction level is determined by the variable E, which specifies the error correction parameters. The first half of E stores the parameter (w_c-3) and the second half stores the parameter (w_r-4) as defined in 5.4.2.

If not specified, E takes the default value of $(001101)_{BIN}$, corresponding to level 3 in Table 21, where $w_c = 4$ and $w_r = 9$. The first 3 bits of E result in 1, i.e., (w_c-3 = 1), and the next 3 bits result in 5, i.e., (w_r-4 = 5).

4.4.1.5 Masking type

The variable MSK in Part II contains the data mask pattern reference from Table 23. This variable exists only in the primary symbol in a JAB Code. All secondary symbols share the same mask type as the primary symbol.

If not specified, MSK takes the default value of $(111)_{BIN}$.

4.4.1.6 Positions of docked secondary symbols

The variable S in Part III indicates the positions of the docked secondary symbols. The first to the fourth bits of S stands for the upper side, the lower side, the left side, and the right side, respectively. A binary one indicates there is a docked secondary symbol at the corresponding side, while a binary zero indicates there is no docked symbol at that side.

S is appended at the end of the data stream and is placed in a bitwise reversed order, followed by a flag bit. The flag bit shall be a binary one. If there are padding bits following the flag bit in the net payload, the padding bits shall consist of only binary zeros.

If there are docked secondary symbols, the metadata of available secondary symbols shall be placed between the encoded message and S in the same bitwise reversed order one after another, as illustrated below.

encoded message	metadata of secondary 4	metadata of secondary 3	metadata of secondary 2	metadata of secondary 1	S	flag bit

Table 8 — Flag values in Part II of a primary symbol

Variable	Value (binary)	Description
V	0000000000 - 1111111111	The first 5 bits stores the side version in the horizontal direction; the next 5 bits stores the side version in the vertical direction. Side-Version x is assigned to the binary representation of (x-1), e.g., side-version 1 is represented by $(00000)_{BIN}$, side-version 6 by $(00101)_{BIN}$ and side-version 32 by $(11111)_{BIN}$.
MSK	000 – 111 (default value: 111)	Mask pattern reference. See Table 23.
E	000100 – 011011 (default value: 001101, corresponding to the error correction level 3 in Table 21)	Error correction level determined by (w_c-3) and (w_r-4) respectively. The first 3 bits stores (w_c-3) and the next 3 bits stores (w_r-4).

4.4.2 Metadata of a secondary symbol

4.4.2.1 Metadata structure

The metadata of a secondary symbol defines the symbol shape, the symbol size, the error correction level and the positions of docked secondary symbols. The metadata are divided into three parts: Part I, Part II, and Part III. The data in the preceding parts determine the length of variables in the following parts. The total length of a secondary symbol metadata varies from 5 to 16 bits.

The metadata structure of a secondary symbol is shown in Table 9. Part I defines two flags. The first one (*SS*) indicates whether the symbol shape and size are identical to the host symbol. The second one (*SE*) indicates whether the secondary symbol shares the same error correction level as the host symbol. Part II defines two variables, the symbol side-version (*V*) and the error correction level (*E*). Part III contains a single variable, (*S*), which defines the positions of docked secondary symbols.

Table 9 — Metadata structure of a secondary symbol

Metadata					
Part I		**Part II**		**Part III**	
Variable	Length (Bits)	Variable	Length (Bits)	Variable	Length (Bits)
same shape and size flag *SS*	1	symbol side-version *V*	variable (0 or 5), depending on SS	positions of docked secondary symbols *S*	3
same error correction level flag *SE*	1	error correction parameters *E*	variable (0 or 6), depending on SE		
Sum	2	Sum	0 to 11	Sum	3
Total length: variable from 5 to 16 bits					

Part I and Part II shall be appended in the data stream of its host symbol preceding the variable S of the host symbol metadata, and encoded together with the data of the host symbol. If the host symbol has multiple docked secondary symbols, the order defined in 4.4.1.6 shall apply. The bit length of Part I is fixed and the bit-length of Part II is determined by the flag values in Part I.

Part III shall be appended to the data stream of the secondary symbol as specified in 4.4.1.6 and encoded together with the data.

4.4.2.2 Symbol shape and size

The secondary symbol may have a different shape than its host symbol. However, the docking side, which is docked to the host symbol, shall have the same size as the corresponding side of the host symbol. In other words, the secondary symbol has only one customizable side.

The symbol shape and size flag (SS) in Part I indicates whether the shape and the size of the secondary symbol is identical to the host symbol. As shown in Table 10, SS takes 1 bit and when SS is binary zero, no further data are required to specify the symbol shape and size, therefore the length of *V* is 0. The corresponding metadata from the host symbol shall be used in the decoding. When SS is binary one, the variable *V* takes 5 bits to specify the side-version of the customizable side.

4.4.2.3 Error correction level

The secondary symbol may either share the same error correction level as its host symbol or use its own error correction level to encode data. The flag SE in Part I indicates whether a different error correction level is specified in the secondary symbol, as shown in Table 10. If a different error correction level is specified, the variable *E* in Part II defines the error correction parameters in the same way as in the primary symbol.

4.4.2.4 Positions of docked secondary symbols

Secondary symbols may dock additional secondary symbols on the three free sides. The variable S in Part III contains 3 bits, which represent the positions of docked secondary symbols. The first to the third bits stand for the three free sides in the same order as defined in 4.4.1.6, skipping the docking side to the host symbol. A binary one indicates there is a docked secondary symbol on the corresponding side, while a binary zero indicates there is no docked symbol on that side. The placement of S and the metadata of further docked secondary symbols is the same as defined in 4.4.1.6.

Table 10 — Flag values in Part I and variable length in Part II of a secondary symbol metadata

Flag	Value	Length (Bits)	Description
SS	0	V: 0	The secondary symbol has the identical shape and size as the host symbol.
	1	V: 5	The side of the secondary symbol that is not docked to the host has a different side-version specified by *V*.
SE	0	E: 0	The secondary symbol shares the same error correction level as the host symbol.
	1	E: 6	The secondary symbol uses a different error correction level specified by *E*.

4.4.3 Metadata error correction encoding

Part I and Part II of the primary symbol metadata are encoded using the LDPC code separately, which results in a doubled bit length. Refer to Annex C for more details of error correction encoding of each part of the metadata.

The error correction bits for Part I and Part II of the primary symbol metadata shall be calculated as described in Annex C, and appended to the metadata bits. Table 11 lists the possible bit length for each metadata part in the primary symbol. The total length of the final encoded metadata are 44 bits.

Part III of the primary symbol metadata and the secondary symbol metadata are encoded together with the data stream of corresponding symbols as specified in 5.7.

Table 11 — Lengths of encoded metadata parts of a primary symbol

Metadata part	Original length (Bits)	Encoded length (Bits)	Total
Part I	3	6	44
Part II	19	38	

4.4.4 Reserved modules for metadata and colour palette

In primary symbols, Part I and Part II metadata and colour palettes shall be encoded using the modules at reserved positions. In secondary symbols, the colour palettes shall be placed at reserved positions.

According to the encoded length listed in Table 11, in the primary symbol metadata, Part I requires 4 modules as it shall be encoded in three-colour mode as specified in 4.4.1.2. Part II require 19 modules if they are encoded using four colours in module colour mode 1, which have the lowest encoding capacity per module. Therefore, up to 23 modules are needed to encode Part I and Part II metadata into a primary symbol.

In addition, up to 24 modules are needed to store the four colour palettes, in either a primary or a secondary symbol, for eight module colours. Therefore, up to 41 modules shall be reserved in a primary symbol for metadata, including error correction bits and colour palettes. Twenty-four modules shall be reserved in a secondary symbol for colour palettes. Table 12 lists the maximal number of reserved modules in primary symbol for each module colour mode.

Table 12 — Number of reserved modules for metadata and colour palettes in a primary symbol

Module colour mode	Number of module colours	Maximum number of modules for metadata	Number of modules for colour palettes	Total
1	4	23	8	31
2	8	17	24	41

Following the placement order shown in Figure 9, for primary symbols, the string of metadata bits, including the error correction bits, shall be encoded into the reserved modules. Each module contains one or more bits. The remaining bits in the last-used module shall be filled with binary zeros.

When the Part I and Part II metadata are available, Part I shall be placed first in the modules from 0 to 3.

Following Part I, the colour palettes shall be placed with each colour taking one module. When using 8-colours, the colour palette, as defined in Table 3, shall be placed clockwise, beginning in the upper left corner, in the order c5 c5 c5 c5 c6 c3 c3 c6 c1 c1 c1 c1 c2 c2 c2 c2 c4 c4 c4 c4 c7 c7 c7 c7. When using 4-colours, the colour palette, as defined in Table 4, shall be placed clockwise, starting in the upper left corner, in the order c2 c2 c2 c2 c3 c1 c1 c3. This guarantees the existence of all colours in each corner.

After the colour palettes, the Part II metadata shall be placed in the following reserved modules.

When the Part I and Part II metadata are not available, the colour palettes shall be placed starting from the first reserved module.

Following the placement order shown in Figure 10 for secondary symbols, when using 8-colours, the colour palette, as defined in Table 3, shall be placed in the reserved modules clockwise, beginning in the upper left corner, in the order c5 c5 c5 c5 c0 c0 c0 c0 c1 c1 c1 c1 c2 c2 c2 c2 c4 c4 c4 c4 c7 c7 c7 c7. When using 4-colours, the colour palette, as defined in Table 4, shall be placed clockwise, starting in the upper left corner, in the order c2 c2 c2 c2 c0 c0 c0 c0.

If the number of required modules are less than the reserved ones, the unused modules shall be used for data encoding in either primary or secondary symbols.

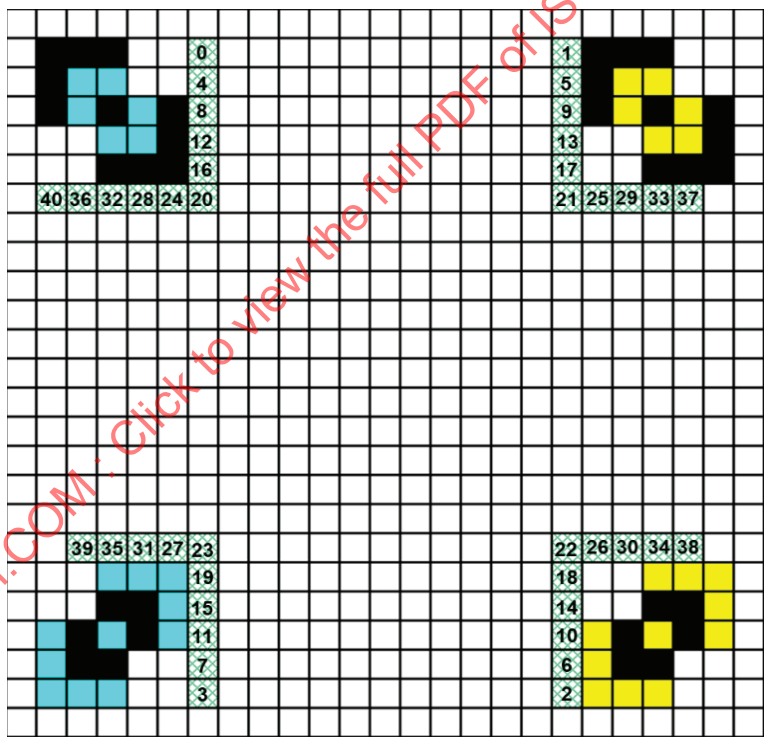


Figure 9 — Metadata and colour palette module placement in a primary symbol

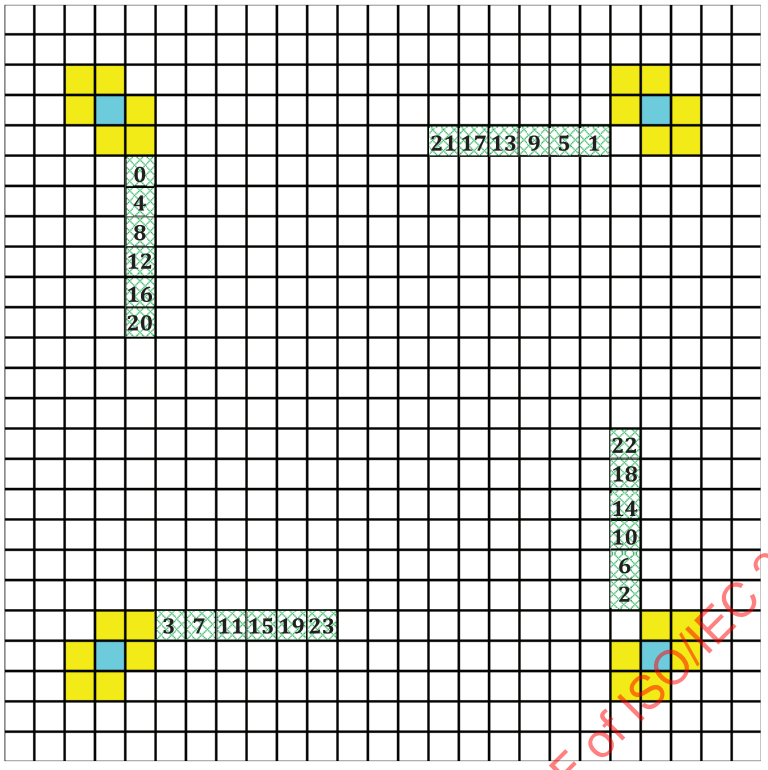


Figure 10 — Colour palette module placement in a secondary symbol

4.5 Symbol Cascading

4.5.1 Symbol docking rules

JAB Code may have arbitrary forms by cascading primary and secondary symbols in horizontal and vertical directions. A JAB Code shall contain one and only one primary symbol and may, optionally, have multiple secondary symbols. Secondary symbols shall be docked to the primary symbol, or other secondary symbols.

While the primary and secondary symbols may be of different shapes, square or rectangular, the docking side between two adjacent symbols shall share the same Side-Version. It is recommended that the primary symbol possess the largest symbol size.

Figure 11, Figure 12 and Figure 13 illustrate three examples of JAB Code with cascaded symbols. The primary and secondary symbols in Figure 11 have the same shape and size. Figure 12 shows a code containing primary and secondary symbols of different shapes and sizes. Figure 13 shows a recursive symbol docking example, in which the primary symbol has secondary symbols docked to it, and one of the secondary symbols has another secondary symbol docked to it.

4.5.2 Symbol decoding order

1) When decoding JAB Code, always start with the primary symbol. If more than one secondary symbol is docked to the primary symbol, the decoding shall follow the order top-bottom-left-right. If any secondary symbol docked to the primary symbol have additional secondary symbols docked to them, the decoding shall follow the order defined below. The secondary symbols that are directly docked to the primary symbol shall be decoded first, according to the top-bottom-left-right order. In Figure 14, they are primary (0), top (1), bottom (2), left (3), right (4). These are denoted as the first layer.

2) According to the top-bottom-left-right order, the secondary symbols in the first layer shall be checked in turn. If there are further docked secondary symbols, they shall be decoded according

to the top-bottom-left-right order. In Figure 14, starting with the first secondary symbol (1), they are top (5), bottom, (primary, so no symbol - skip), left (6), right (7). These secondary symbols are denoted as the second layer.

3) Apply the same order to the other docked secondary symbols in further layers until all the secondary symbols are decoded. In Figure 14, starting with the 2nd secondary symbol (2), they are; top (primary, so no decode - skip), bottom (8), left (9), right (10). This is the 3rd layer. NOTE: Whenever the top-bottom-left-right decode cycle hits a previously decoded symbol, it does not count that module and proceeds to the next step in the cycle. If the cycle is done, then it proceeds to the next group of secondary symbols to decode in the same top-bottom-left-right manner. For instance, the 4th layer (starting with secondary symbol #3) would decode top (6, already decoded if symbol #1 exists, so skip), bottom (9, already decoded if symbol #2 exists, so skip), left (11), right (primary, already decoded, so skip).

The indices of the first 60 secondary symbols are defined in Figure 14. However, it is important to note that cascading should be limited such that the reader is able to resolve at least 5 pixels per module.

Figure 15 illustrates an example of the decoding order of cascaded symbols. The indices of the symbols indicate the decoding order. The symbols with a smaller index shall be decoded first.

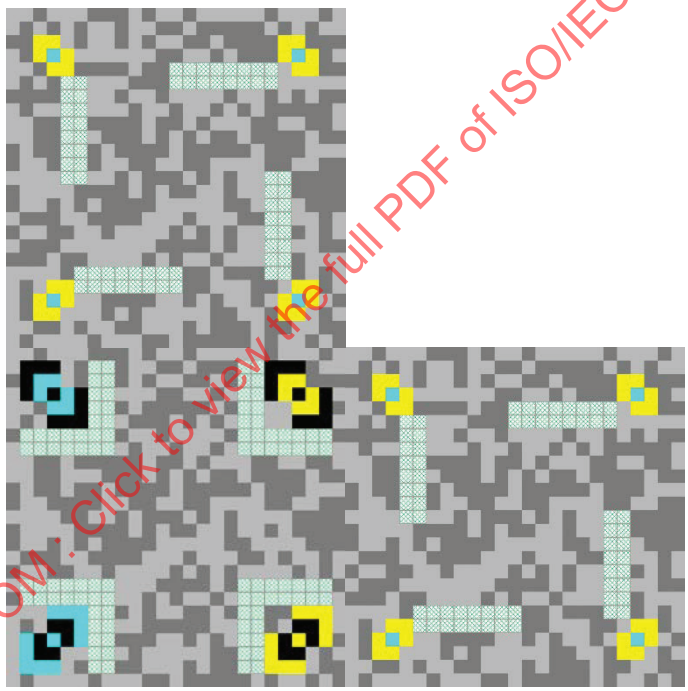


Figure 11 — JAB Code with one square primary and two square secondary symbols

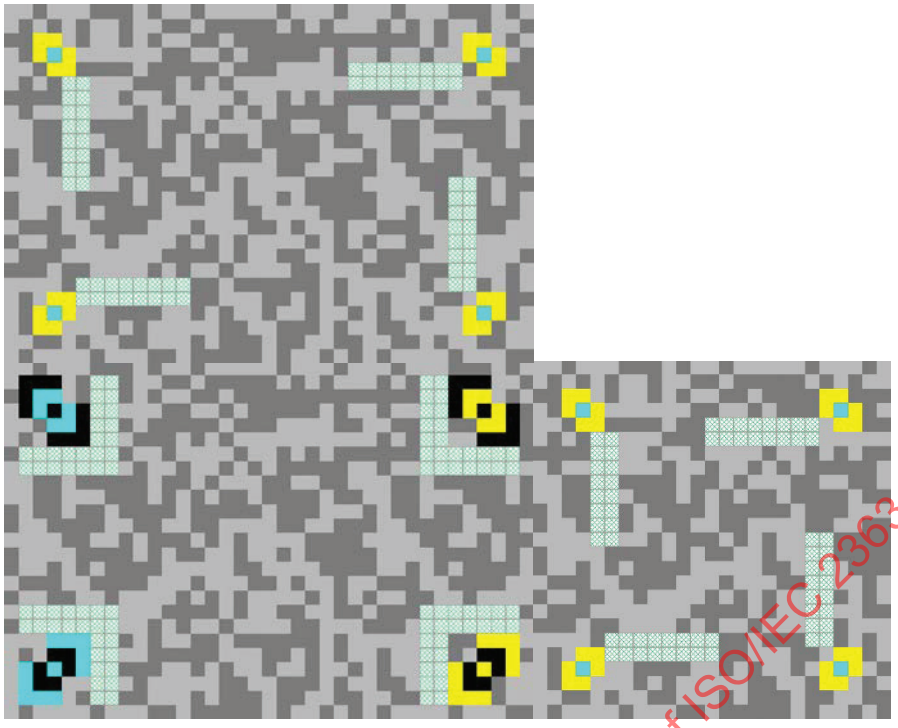


Figure 12 — JAB Code with one rectangle primary symbol and two secondary symbols

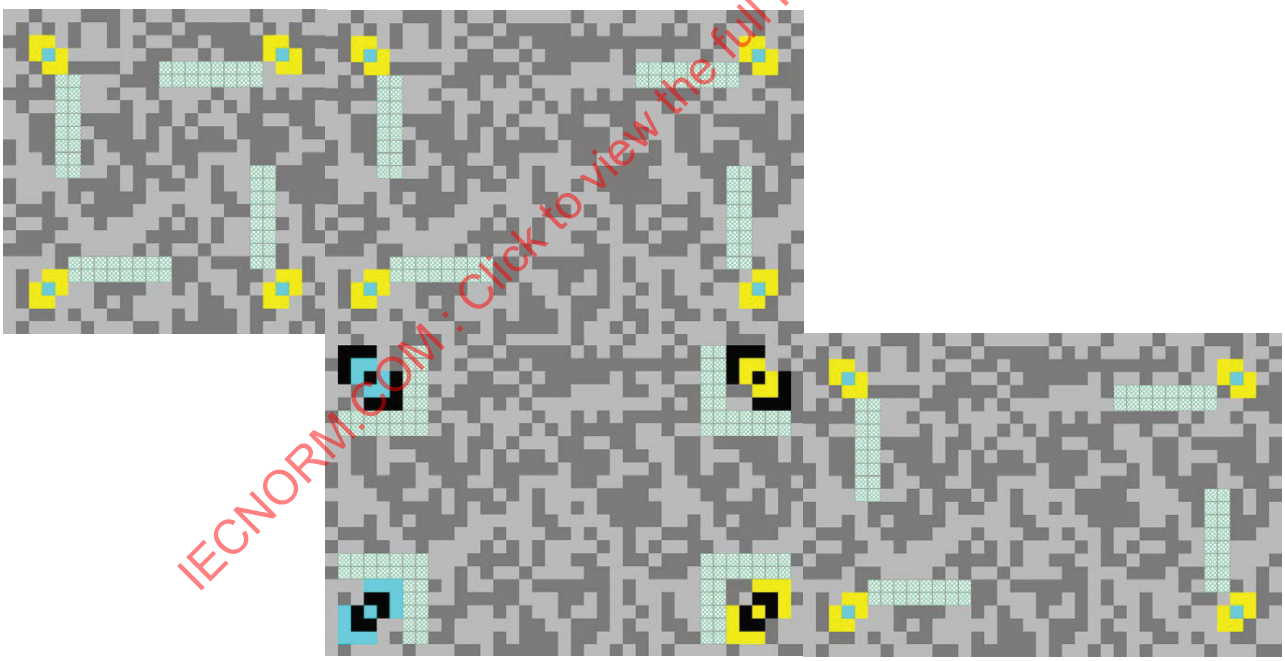
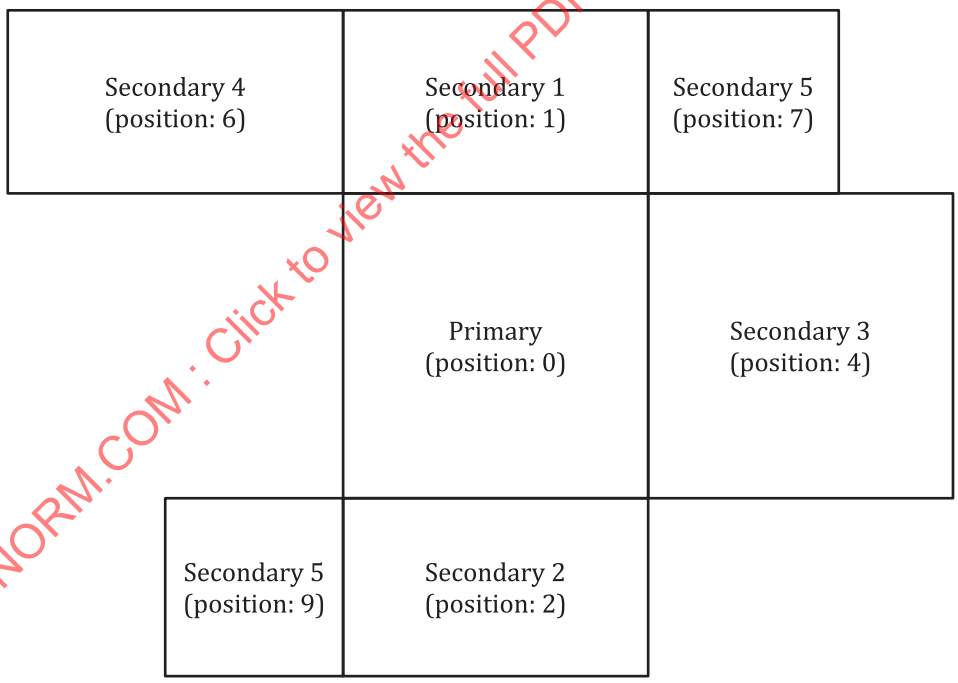


Figure 13 — JAB Code with one rectangle primary symbol and multiple square and rectangle secondary symbols

					41					
				42	25	43				
			44	26	13	27	45			
		46	28	14	5	15	29	47		
	48	30	16	6	1	7	17	31	49	
59	39	23	11	3	0	4	12	24	40	60
	57	37	21	9	2	10	22	38	58	
		55	35	19	8	20	36	56		
			53	33	18	34	54			
				51	32	52				
					50					

Figure 14 — Positions of cascaded primary and secondary symbols



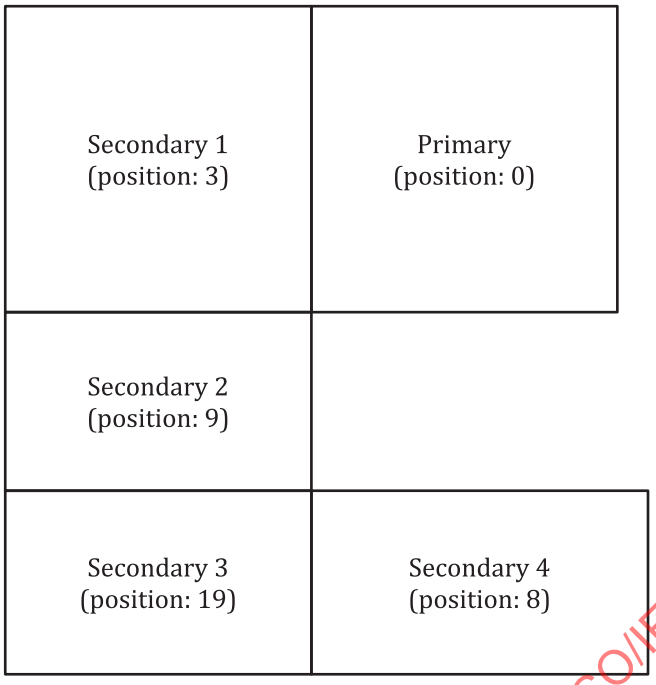


Figure 15 — Examples of decoding order of cascaded primary and secondary symbols

5 Symbol generation

5.1 Encoding procedure overview

The following steps are required to convert input data to a JAB Code symbol. See Annex D for an example.

1) Data analysis

   Analyse the input data to identify the most efficient encoding modes. Seven default encoding modes are supported to convert input data into a binary string, which can be switched as needed by the shift and latch mode switches. Additional character sets are enabled by supporting Extended Channel Interpretation (ECI) and FNC1 encoding. See 5.3.

2) Data encoding

   Convert the data characters into a binary stream using the selected encoding modes as specified in 5.3.

3) Error correction coding

   Encode the binary stream using LDPC for error correction and append the parity data to the end of the source binary stream, as specified in 5.4.

4) Data interleaving

   Interleave the encoded data in each symbol and add padding bits if necessary. See 5.5.

5) Metadata module reservation (optional)

   If other than the default metadata are selected, calculate the actual metadata size based on the parameters, including the number of module colours, symbol shape, symbol size, error correction level and code structure, and reserve the modules, which are required to accommodate the metadata, following the sequence specified in 4.4.4.

6) Data module placement

Place the finder patterns, the alignment patterns and the colour palettes in the matrix. Then place the data modules (including the modules for error correction bits) in the remaining area of the matrix (skipping the reserved modules for metadata). See 5.7.

7) Data masking

Apply every available data mask pattern onto the data modules and evaluate the masking results. Select the masking pattern, which results in the most balanced module colour distribution and minimizes the occurrence of undesirable patterns. See 5.8.

8) Metadata generation and module placement (optional)

If other than the default metadata are selected, the code parameters, including the number of module colours, symbol shape, symbol size, error correction level, masking type and code structure, are used to generate the metadata information for each symbol as defined in 4.4.1 and 4.4.2 and to encode the generated metadata as specified in 4.4.3. Finally, the encoded metadata information is placed into the reserved modules, following the sequence specified in 4.4.4.

5.2 Data analysis

Analyse the character types of the input data and choose appropriate encoding modes in order to encode the input data with the shortest bit stream. Annex F presents an algorithm for selecting the most appropriate encoding modes to minimize the bit stream length.

5.3 Encoding modes

5.3.1 Encoding modes and character set

There are nine encoding modes in JAB Code: uppercase mode, lowercase mode, numeric mode, punctuation mode, mixed mode, alphanumeric mode, byte mode, ECI mode and FNC1. The first six character encoding modes are defined in Table 13. In JAB Code, the default interpretation is UTF-8 in accordance with the ISO/IEC 10646 character set.

Encoding can be switched from one mode to another mode as often as needed using two methods; shift and latch. Shift indicates a temporary switch only for the next character, e.g., shift to the punctuation mode (P/S) and shift to the uppercase mode (U/S). Latch indicates a permanent switch for the following characters until another mode switch is encountered, e.g., latch to the lowercase mode (L/L), latch to the numeric mode (N/L).

Table 13 — JAB Code character encoding modes. The abbreviations are defined in 5.3.1 and the corresponding paragraphs of the encoding modes. For better readability of the table, the values used by JAB-Code for encoding are yellow, the characters to be encoded are green.

	Upper		Lower		Numeric		Punct		Mixed		Alphanumeric					
Value	Char	ISO	Char	ISO	Char	ISO	Char	ISO	Char	ISO	Value	Char	ISO	Value	Char	ISO
0	SP	32	SP	32	SP	32	!	33	#	35	0	SP	32	32	V	86
1	A	65	a	97	0	48	“	34	*	42	1	0	48	33	W	87
2	B	66	b	98	1	49	$	36	+	43	2	1	49	34	X	88
3	C	67	c	99	2	50	%	37	<	60	3	2	50	35	Y	89
4	D	68	d	100	3	51	&	38	=	61	4	3	51	36	Z	90
5	E	69	e	101	4	52	'	39	>	62	5	4	52	37	a	97
6	F	70	f	102	5	53	(	40	[	91	6	5	53	38	b	98
7	G	71	g	103	6	54	)	41	\	92	7	6	54	39	c	99
8	H	72	h	104	7	55	,	44	]	93	8	7	55	40	d	100

Table 13 *(continued)*

	Upper		Lower		Numeric		Punct		Mixed		Alphanumeric					
Value	Char	ISO	Char	ISO	Char	ISO	Char	ISO	Char	ISO	Value	Char	ISO	Value	Char	ISO
9	I	73	i	105	8	56	-	45	^	94	9	8	56	41	e	101
10	J	74	j	106	9	57	.	46	_	95	10	9	57	42	f	102
11	K	75	k	107	,	44	/	47	`	96	11	A	65	43	g	103
12	L	76	l	108	.	46	:	58	{	123	12	B	66	44	h	104
13	M	77	m	109	P/S		;	59	|	124	13	C	67	45	i	105
14	N	78	n	110	U/L		?	63	}	125	14	D	68	46	j	106
15	O	79	o	111	MS		@	64	~	126	15	E	69	47	k	107
16	P	80	p	112					HT	9	16	F	70	48	l	108
17	Q	81	q	113					LF	10	17	G	71	49	m	109
18	R	82	r	114					CR	13	18	H	72	50	n	110
19	S	83	s	115					CR LF	10, 13	19	I	73	51	o	111
20	T	84	t	116					, SP	44, 32	20	J	74	52	p	112
21	U	85	u	117					. SP	46, 32	21	K	75	53	q	113
22	V	86	v	118					: SP	58, 32	22	L	76	54	r	114
23	W	87	w	119					€	226, 130, 172	23	M	77	55	s	115
24	X	88	x	120					§	194, 167	24	N	78	56	t	116
25	Y	89	y	121					Ä	195, 132	25	O	79	57	u	117
26	Z	90	z	122					Ö	195, 150	26	P	80	58	v	118
27	P/S		P/S						Ü	195, 156	27	Q	81	59	w	119
28	L/L		U/S						ß	195, 159	28	R	82	60	x	120
29	N/L		N/L						ä	195, 164	29	S	83	61	y	121
30	A/L		A/L						ö	195, 182	30	T	84	62	z	122
31	MS		MS						ü	195, 188	31	U	85	63	MS	

5.3.2 Uppercase mode

The uppercase mode encodes 27 characters, including 26 capital letters (A-Z) and the SPACE character, at 5 bits per character. Each character is assigned a character value from 0 to 26 according to Table 13.

The remaining five values are used for a mode switch, as defined in Table 14. The first four values, from 27 to 30, define a direct switch to the punctuation, lowercase, numeric and alphanumeric modes. The last value 31 defines an extension of more switches. Four more mode switches are indicated by appending two bits at the end of $(11111)_{BIN}$. For example, $(1111100)_{BIN}$ indicates shifting into the byte mode and $(1111111)_{BIN}$ indicates eight additional switches by appending three bits at the end of $(1111111)_{BIN}$. For example, $(1111111001)_{BIN}$ encodes the message sequence 'https://', which is common in the URL encoding, and $(1111111000)_{BIN}$ indicates the encoding according to ISO/IEC 15434.

Data encoding starts by default in uppercase mode.

Table 14 — Mode switch in the uppercase mode

	Value	Char	Mode Switch		
Direct	27	P/S	shift to the punctuation mode for the next character		
	28	L/L	latch to the lowercase mode for the following characters		
	29	N/L	latch to the numeric mode for the following characters		
	30	A/L	latch to the alphanumeric mode for the following characters		
Extended	31	MS	more switches by appending bits	00	shift to the byte mode
				01	shift to the mixed mode
				10	latch to the ECI mode
				11	additional switches see Table 15

Table 15 — Additional switches in the uppercase mode

	Value	Char
Additional switches	000	[) > $^{R}_{S}$ shift to encode according to ISO/IEC 15434 until $^{E}_{oT}$, which indicates the end of this mode
	001	https://
	010	http://
	011	www.
	100	FNC1 represented by decimal value 29 or Hexadecimal value 1D.
	101	$^{E}_{oT}$ represented by decimal value 4 or Hexadecimal value 04.
	110	Reserved
	111	Reserved

Table 16 — Mode switch in the lowercase mode

	Value	Char	Mode Switch		
Direct	27	P/S	shift to the punctuation mode for the next character		
	28	U/S	shift to the uppercase mode for the next character		
	29	N/L	latch to the numeric mode for the following characters		
	30	A/L	latch to the alphanumeric mode for the following characters		
Extended	31	MS	more switches by appending bits	00	shift to the byte mode
				01	shift to the mixed mode
				10	latch to the uppercase mode
				11	shift to the numeric mode

5.3.3 Lowercase mode

The lowercase mode encodes 27 characters, including 26 small letters (a-z) and the SPACE character, at 5 bits per character. Each character is assigned a character value from 0 to 26 according to Table 13.

The remaining five values are used for a mode switch, as defined in Table 16. The first four values from 27 to 30, define direct switch to the punctuation, uppercase, numeric and alphanumeric mode. The last value 31 defines an extension of more switches. Four more mode switches are indicated by appending two bits at the end of $(11111)_{BIN}$. For example, $(1111100)_{BIN}$ indicates shifting into the byte mode and $(1111101)_{BIN}$ indicates shifting into the mixed mode.

5.3.4 Numeric mode

The numeric mode encodes 13 characters, including 10 digits (0-9), the SPACE character and two punctuation marks, at 4 bits per character. Each character is assigned a character value from 0 to 12 according to Table 13.

The remaining three values are used for the mode switch, as defined in Table 17. The first two values, 13 to 14, define a direct switch to the punctuation and the uppercase modes. The last value 15 defines an extension of more switches, which indicates four more mode switches by appending two bits at the end of $(1111)_{BIN}$, for example, $(111100)_{BIN}$ indicates shifting into the byte mode and $(111101)_{BIN}$ indicates shifting into the mixed mode.

Table 17 — Mode switch in the numeric mode

	Value	Char	Mode Switch		
Direct	13	P/S	shift to the punctuation mode for the next character		
	14	U/L	latch to the uppercase mode for the following characters		
Extended	15	MS	more switches by appending bits	00	shift to the byte mode
				01	shift to the mixed mode
				10	shift to the uppercase mode
				11	latch to lowercase mode

5.3.5 Punctuation mode

The punctuation mode encodes 16 commonly used punctuation characters, at 4 bits per character. Each character is assigned a character value from 0 to 15, according to Table 13.

The punctuation mode has a fixed run-length of one character, after which encoding reverts to the mode from which the punctuation mode was invoked.

5.3.6 Mixed mode

The mixed mode encodes Germanic umlauts, more punctuation characters and other marks, control characters and combinations, at 5 bits per character. Each character is assigned a character value from 0 to 31, according to Table 13.

Like punctuation mode, the mixed mode also has a fixed run-length of one character, after which encoding reverts to the mode from which the mixed mode was invoked.

5.3.7 Alphanumeric mode

The alphanumeric mode encodes 63 characters, including 26 capital letters (A-Z), 26 small letters (a-z), 10 digits (0-9) and the SPACE character, at 6 bits per character. Each character is assigned a character value from 0 to 62, according to Table 13.

The remaining value 63 is used for a mode switch, as defined in Table 18, which defines an extension of four mode switches by appending two bits at the end of $(111111)_{BIN}$. For example, $(11111100)_{BIN}$ indicates shifting into the byte mode, and $(11111110)_{BIN}$ indicates shifting into the punctuation mode.

Table 18 — Mode switch in the alphanumeric mode

	Value	Char	Mode Switch		
Extended	63	MS	more switches by appending bits	00	shift to the byte mode
				01	shift to the mixed mode
				10	shift to the punctuation mode
				11	latch to the uppercase mode

5.3.8 Byte mode

The byte mode encodes any 8-bit characters at 8 bits per character. The yte mode starts with a 4-bit binary value, which, if non-zero, encodes the number of bytes (1-15) that follow, but if zero then the next 13 bits encode the number of bytes minus 16. For example, $(0000000000000)_{BIN}$ indicates that 16 bytes follow and $(0000000001111)_{BIN}$ indicates that there 31 bytes follow. Thus, the byte mode can encode any ASCII characters, as specified in ISO/IEC 646, that are not included in Table 13 and long strings of binary data, possibly filling the whole symbol. At the end of the byte string, encoding returns to the mode from which the byte mode was invoked.

Table 19 — Encoding the ECI assignment number

ECI Assignment Number	Encoded value
000000 to 000127	**0**bbbbbbb
000000 to 016383	**10**bbbbbb bbbbbbbb
000000 to 999999	**11**bbbbbb bbbbbbbb bbbbbb
where b...b is the binary value of the ECI assignment number	

5.3.9 Extended Channel Interpretation (ECI) mode

The ECI mode enables data interpretation different from the default character set. The ECI mode starts with a 6-digit ECI assignment number which is encoded as an 8-bit, 16-bit, or 22-bit binary string, as defined in Table 19. The preceding indicating bits determines the length of the binary string.

— If it begins with a 0 bit, it contains 8 bits.

— If it begins with “10”, it contains 16 bits.

— If it begins with “11”, it contains 22 bits.

In each case, the bits that follow the indicating bits are the binary representation of the ECI assignment number. After the ECI assignment number, the encoding returns to the mode from which ECI mode was invoked.

In the input data to be encoded, the ECI assignment number is represented as a backslash character, $(5C)_{HEX}$, followed by a 6-digit number, e.g., “\nnnnnn”. When the ECI protocol applies, if the input data contains a backslash character, it shall be doubled as two $(5C)_{HEX}$ characters. See 7.3.

Data in an ECI sequence shall be handled as 8-bit values, which can be encoded using any of the encoding modes, regardless of their significance. For example, a sequence of bytes in the range $(30)_{HEX}$ to $(57)_{HEX}$, according ISO/IEC 646, would be most efficiently encoded in the numeric mode, even if the sequence might not actually represent numeric data.

Any ECI invoked shall apply until the end of the encoded data, or until another ECI is encountered.

5.3.10 FNC1 mode

The FNC1 mode is used for messages containing specific data formats in accordance with GS1 General Specifications[4]. The encoding shall start with FNC1 and end with the control character $^{E}_{oT}$, after which, encoding reverts to the upper case mode from which the FNC1 mode was invoked. If the FNC1 occurs within the start and end indicator, it serves as a field separator.

5.4 Error correction

5.4.1 Error correction levels

The error correction coding is performed by the LDPC code and operates on binary data. After encoding the message data to a binary stream, as specified in 5.3, the LDPC code adds check bits to the binary

stream to enable the symbol to remain decodable in case of damage. The error correction level shall be selectable between 1 and 10, as listed in Table 20. The default error correction level shall be 3.

Table 20 shows the recovery capability of the bit errors in more than 95 % of cases. More errors can be detected by the error correction code, but with a probability less than 95 %. The error correction level shall be determined by the application requirements and expected symbol quality. Higher error recovery capacity is achieved at the cost of larger datastream size, leading to larger symbol size. The increase of the datastream size is indicated by the code rate R, which is defined as $R = P_n/P_g$.

Table 20 — The approximated amount of bit error recovery capacity in %

Level	Recovery capacity in %	w_c	w_r	Code rate R
1	4	3	8	0,63
2	5	3	7	0,57
3(default)	6	4	9	0,55
4	7	3	6	0,50
5	8	4	7	0,43
6	9	4	6	0,34
7	10	3	4	0,25
8	11	4	5	0,20
9	12	5	6	0,17
10	14	6	7	0,14

5.4.2 Error correction parameters

Based on the input data and the symbol capacity, the best combination of the two parameters w_c and w_r shall be determined. The value of w_c shall be an integer between 3 and 8, w_r an integer between 4 and 9.

Given the symbol capacity and the net payload, the relation $w_r/w_c = C/(C\text{-}P_n)$ holds. The parameters, w_c and w_r, shall be determined by the code rate $R = 1\text{-}w_c/w_r$ which is closest to but smaller than the corresponding code rate of the specified error correction level. For a given code rate the smallest numbers for w_c and w_r shall be used to achieve the best error correction performance e.g., for code rate R = 0.5, $w_c = 3$ and $w_r = 6$ shall be used.

With these two parameters w_c and w_r, the number of error correction bits and the gross payload are specified by:

$$K = \left\lfloor \frac{C \times w_c}{w_r} \right\rfloor$$

and

$$P_g = P_n + K$$

With P_g and K determined, the size of the parity check matrix H is determined, and the matrix H shall be generated as defined in 5.4.4.

5.4.3 Padding Bits

The number of padding bits S_{Bit} for the unused capacity is $S_{Bit} = C\text{-}P_g$, and for the unused net payload $S_{BitData} = P_n\text{-}P_e$.

The padding bits S_{Bit} are a sequence consisting of alternate '0s and '1's, starting with '0', filling up the unused capacity at the end of the error correction stream.

The padding bits $S_{BitData}$ are a sequence consisting of only '0', filling up the unused net payload.

5.4.4 Generating the error correction stream

The error correction stream c is generated by multiplying the message m with the generator matrix G in the GF(2) with c = m⊗G. There are four steps to obtain the generator matrix:

Step 1: Construct a matrix A_0 with K/w_c rows and P_g columns:

$$A_0 = \begin{bmatrix} \underbrace{1111}_{w_r} & \ldots 0 \ldots & \ldots 0 \ldots \\ \ldots 0 \ldots & \underbrace{1111}_{w_r} & \ldots 0 \ldots \\ \ldots 0 \ldots & \ldots 0 \ldots & \underbrace{1111}_{w_r} \end{bmatrix}$$

Step 2: Form the matrix A with K rows and P_g columns by stacking w_c permutations:

$$A = \begin{bmatrix} \pi_1(A_0) \\ \pi_2(A_0) \\ \pi_3(A_0) \\ . \\ . \\ . \end{bmatrix}$$

where π is the permutation. The permutation is performed by using the random number generator R(seed, A_0), as defined in Annex F, with the initial seed 785465 for the message data, and the seed 38545 for the metadata, with π_1 = R(seed, A_0), π_2 = R(seed, π_1), π_3 = R(seed, π_2), ...

Step 3: Generate the matrix H by using the Gauss-Jordan elimination for the matrix A to obtain $H(C^T \mid I) \in (K \times P_g)$.

Step 4: The generator matrix is created by $H(I \mid C) \in (P_n \times P_g)$.

An example is given in Annex B.

5.5 Data interleaving

The final sequence of encoded data shall be constructed following the steps below.

1) Calculate the remaining capacity in the selected symbol, which is equal to C-P_g.

2) Fill the unused capacity with padding bits to get the final sequence. The padding bits shall be a binary string containing alternating 0 and 1 and starting with 0, e.g., 0101010.

3) Interleave all bits in the final sequence using the random permutation algorithm with an initial seed of 226759, as defined in Annex F.

5.6 Metadata module reservation

In a JAB Code symbol, the metadata in primary symbols are optional and the metadata in secondary symbols are variable-length data. The actual metadata length and the number of modules required to accommodate the metadata are determined by the following five code parameters, which may be specified by user input: the number of module colours, symbol shape, symbol size, error correction level and code structure.

If none of these parameters are specified by user input, the default parameter shall be used according to the following list:

1) The number of module colours – colour mode 2 (8 module colours)

2) Error correction level – level 3

3) Symbol shape – square symbol

4) Symbol size – smallest square symbol

5) Code structure – single primary symbol

If one of the parameters is user selected, the number of required metadata modules shall be determined and reserved in each symbol, following the placement order defined in 4.4.4.

5.7 Data module encoding and placement

The modules for finder patterns, alignment patterns and colour palettes shall be first placed in the matrix. Then the modules for data message shall be placed.

The bits of encoded data message, including the error correction bits, are graphically encoded using each colour module in the colour palette to represent $\log_2(N_c)$ bits. The $\log_2(N_c)$ binary bits are encoded using the index value of module colour in the palette. For example, if there are eight module colours in a symbol, i.e., $\log_2(N_c) = 3$, the first module colour in the colour palette represents 000, the second one represents 001, and so forth, as defined in Table 21.

The interleaved final sequence of encoded data shall be placed in the remaining modules, starting from the upper left available module, running downwards from left to right, to the most lower right available module, skipping over the modules occupied by finder patterns, alignment patterns, metadata, and colour palettes, as shown in Figure 16. The data module placement in secondary symbols follows the same way as in primary symbols.

Table 21 — Bit encoding using module colours

Module colour mode	Module colour	Colour index	Binary bits
$N_c = 2$ (eight colours)	black	0	000
	blue	1	001
	green	2	010
	cyan	3	011
	red	4	100
	magenta	5	101
	yellow	6	110
	white	7	111
$N_c = 1$ (four colours)	black	0	00
	cyan	1	01
	magenta	2	10
	yellow	3	11

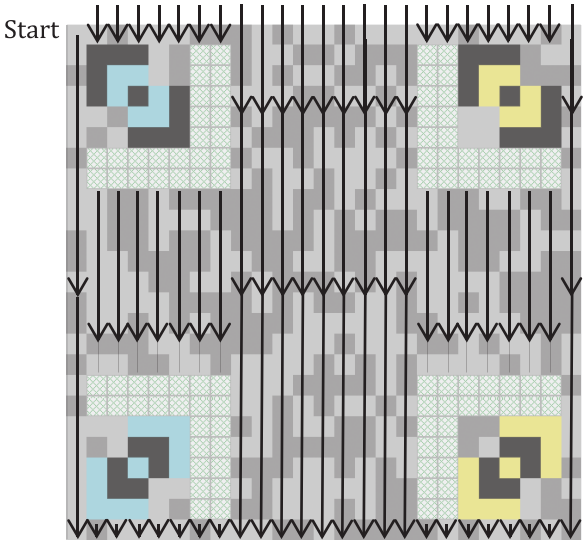


Figure 16 — Data module placement

5.8 Data masking

5.8.1 Data masking rules

For reliable JAB Code reading, the distribution of colour modules shall meet the following two conditions:

a) The colour modules shall be arranged in a well-balanced manner in the symbol.

b) The occurrences of patterns similar to finder patterns and alignment patterns in other regions of the symbol shall be avoided as much as possible.

To meet the above conditions, data masking shall be applied as follows.

a) Data masking is only applied to data modules, not to modules for finder pattern, alignment patterns, metadata, and colour palettes.

b) Apply each data mask pattern to data modules in turn. The masking result of each module is calculated by the XOR operation between the module colour and the mask pattern colour, as defined in 5.8.2.

c) Evaluate the masking results by charging penalties for undesirable features.

d) Select the data mask pattern with the lowest penalty point score.

5.8.2 Data mask patterns

JAB Code has eight data mask patterns, as listed in Table 22. The binary reference values are used in metadata to identify the masking type. Each data pattern covers only the modules in the data encoding region in a symbol, excluding modules for finder pattern, alignment patterns, metadata, and colour palettes. The colour of each module in a data mask pattern is determined by the pattern module colour generators, as defined in Table 22. The result of each generator indicates the index in the colour palette of the symbol. In the generators, x refers to the horizontal position of the module and y refers to its vertical position, with (x, y) = (0, 0) for the upper left module in the symbol.

The data masking is applied to a data module through the bitwise XOR operation between the colour index of the data module and the colour index of the corresponding module in the mask pattern. For example, in case of N_c = 2, if the module has the colour index of 5, $(101)_{BIN}$, and the corresponding

module in the mask pattern has the colour index 3, $(011)_{BIN}$, then the resulting module has the colour index 6, $(110)_{BIN}$.

Table 22 — Data mask pattern generation conditions

Data mask pattern reference	Pattern module colour generator
000	$(x+y) \bmod 2^{Nc+1}$
001	$x \bmod 2^{Nc+1}$
010	$y \bmod 2^{Nc+1}$
011	$((x \text{ div } 2) + (y \text{ div } 3)) \bmod 2^{Nc+1}$
100	$((x \text{ div } 3) + (y \text{ div } 2)) \bmod 2^{Nc+1}$
101	$((x+y) \text{ div } 2 + (x+y) \text{ div } 3) \bmod 2^{Nc+1}$
110	$(((x \times x \times y) \bmod 7) + ((2 \times x \times x + 2 \times y) \bmod 19)) \bmod 2^{Nc+1}$
111(default)	$(((x \times y \times y) \bmod 5) + ((2 \times x + y \times y) \bmod 13)) \bmod 2^{Nc+1}$

5.8.3 Evaluation of data masking results

The masking results using each data mask pattern listed in Table 22, shall be evaluated by scoring penalty points for each occurrence of the undesirable features listed in Table 23. The higher the penalty points, the less acceptable the masking result. In Table 23, each undesirable feature is scored by a weighting factor, which is defined as W1 = 100, W2 = 3 and W3 = 3.

In the evaluation, all the modules are considered, although the data masking shall only be applied to the data modules. Furthermore, all the symbols in a JAB Code, including the primary symbol and secondary symbols (if present), shall be evaluated together.

After evaluating the results according to Table 23, the data mask pattern that results in the lowest penalty score shall be selected for the code.

Table 23 — Scoring of data masking results

Feature	Scoring condition	Penalty points
Pattern with the same colour and structure as any finder pattern in row/column	Existence of the pattern	W1
Block of modules in same colour	block_size = m × n	W2 × (m – 1) × (n – 1)
Adjacent modules in row/column in same colour	Number of modules = (5+k), k > 0	W3 + k

5.9 Metadata generation and module placement

The final metadata information shall be constructed for each symbol as defined in 4.4.1 and 4.4.2. It shall be encoded using LDPC as specified in 4.4.3, including the number of module colours, symbol shape, symbol size, data mask pattern type, error correction parameters, and the code structure.

The bits of metadata, including the error correction bits, are encoded into the reserved modules. As defined in 4.4.1.2, all available module colours shall be used to encode metadata, except Part I of metadata in primary symbols, N_C, which shall be encoded with three colours. The other metadata shall be encoded using all available colours.

The encoded metadata bit stream shall be placed into the modules, which are reserved (see 5.6), following the placement order specified in 4.4.4. If the length of metadata are not an exact multiple of N, where N is the number of bits a metadata module represents, up to N-1 zeros shall be appended to the metadata bit stream to fill up the final used module.

6 Reference decode algorithm

6.1 Decoding procedure overview

This reference decode algorithm finds the symbols in an image and decodes them. Decoding a JAB Code from a captured image involves the following steps:

a) Preprocess the captured image and classify colours.

b) Locate the primary symbol within the image by finding the finder patterns.

c) Locate the alignment patterns, if they exist.

d) Establish the sampling grid using the found finder patterns and the alignment patterns and sample the symbol.

e) Decode the Part I metadata of the primary symbol and determine the module colour mode used. If Part I is invalid, skip decoding the metadata in the next steps and use the default metadata values to determine the code parameters.

f) Extract and construct the four colour palettes.

g) Decode Part II metadata of a primary symbol and determine the code parameters, including side-versions, error correction parameters, and mask pattern reference.

h) Decode the data modules by determining their colour index in the nearest colour palette.

i) Release the data masking using the mask pattern corresponding to the decoded mask pattern reference.

j) De-interleave the data stream.

k) Detect and correct errors in the data stream.

l) Decode the data stream into the original message in accordance with the encoding modes in use.

m) Decode the Part III metadata of the primary symbol in the data stream to determine the docking positions of secondary symbols. If docked secondary symbols exist, decode the Part I and Part II metadata of the docked secondary symbols in the following data stream.

n) If it exists, locate the secondary symbols docked to the primary symbol and decode the data stream in the secondary symbols.

o) If it exists, locate, and decode further docked secondary symbols recursively, according to the symbol decoding order.

p) Concatenate the decoded data out of all symbols.

6.2 Pre-processing image and classifying colours

When the tones and colour values of the captured image are limited in a narrow range, the pixel values in the image shall be adjusted by setting the white and black points according to Formula (1);

$$v^*_{x,y} = \begin{cases} 0 & if v_{x,y} = min \\ 255 & if v_{x,y} = max \\ \dfrac{v_{x,y} - min}{max - min} \times 255 & otherwise \end{cases} \quad (1)$$

where:

v is the original pixel value

*v** is the adjusted one

max is the maximal pixel value in the colour channel

min is the minimal value.

The pixel value adjustment shall be done separately in R, G and B channels.

Divide the adjusted image into four blocks. The block width and height is half of the image width and height. Calculate the averages of the pixel values in each block, in each colour channel, i.e., R, G and B.

Classify the colour of each pixel in every block according to the following steps:

a) If the R, G and B values of the pixel are all smaller than the block averages of the corresponding channel, set the RGB value to (0, 0, 0).

b) For each pixel, calculate the normalized standard deviation of the R, G and B values. If the standard deviation is smaller than 0.08 and the R, G and B values are all bigger than the block average, set the RGB value to (255, 255, 255).

c) If the standard deviation is equal to or bigger than 0.08, set the biggest one of the R, G and B values to 255 and the smallest one to 0. Calculate the ratio between the middle one and smallest one, denoted as *rs*. Calculate the ratio of the biggest one and the middle one, denoted as *rb*. If *rs* > *rb*, set the middle value to 255, otherwise set the middle value to 0.

6.3 Locating finder patterns

The four finder patterns in the JAB Code, i.e., Finder Pattern UL, Finder Pattern UR, Finder Pattern LR and Finder Pattern LL, consist of five layers with a core module, located at the four corners of the primary symbol as described in 4.3.7. All of them have a distinct characteristic that may make them easy to be identified in the image: layer widths in each finder pattern form a pC1-C2-C1-C2-qC1 sequence, where C1 and C2 represent the layer colours.

After the colour classification process, C1 and C2 shall have a value of either 0 or 255 in each colour channel, R, G, and B. Finder Pattern UL and Finder Pattern LL have values of alternate 0 and 255 in the G and B channels, and a constant value of 0 in the R channel. See Figure 17. Finder Pattern UR and Finder Pattern LR have values of alternate 0 and 255 in the R and G channels, and constant value of 0 in the B channel. The relative widths of each layer are p:1:1:1:q in all scanlines running through the centre of the core module. In this reference algorithm, the tolerance for each of these widths is 50 %, corresponding to a range of 0.5 to 1.5.

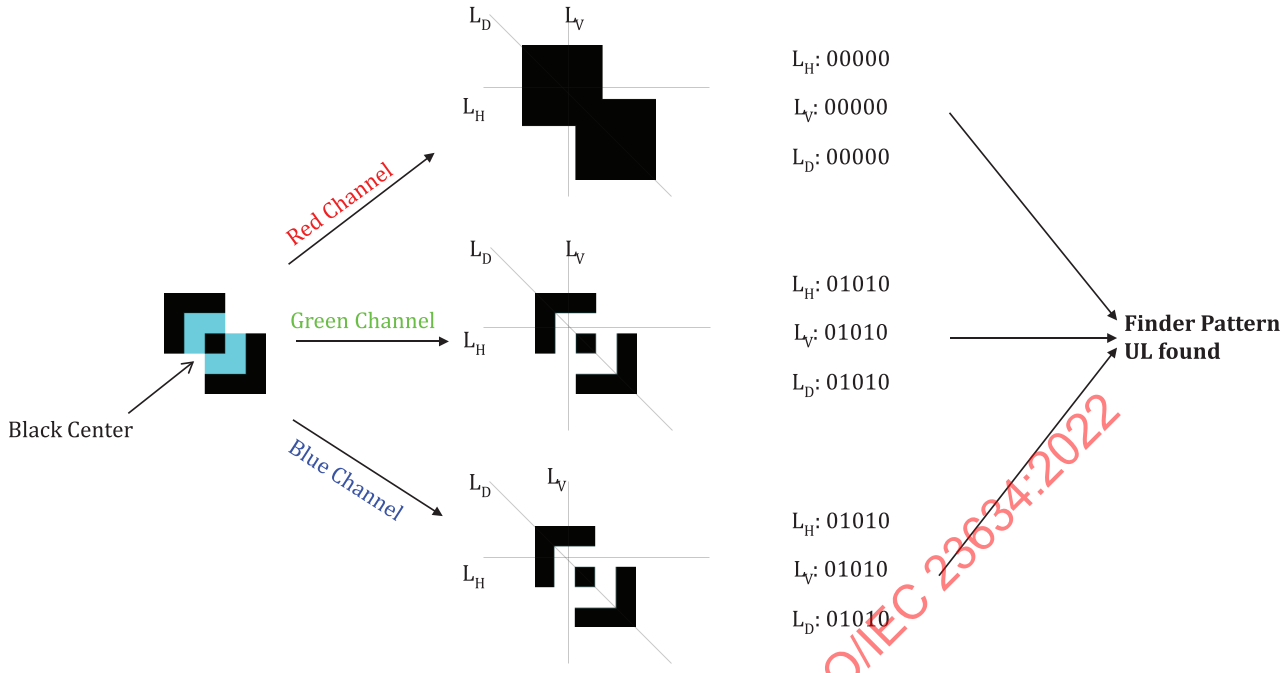


Figure 17 — Detect finder pattern using the example of the upper left finder pattern.

The finder pattern is separated in its colour channels. L_V, L_D, and L_H are the search patterns for the vertical, diagonal, and horizontal search direction. If, for each colour channel, the three scans (horizontal, vertical, and diagonal) match, a finder pattern is found.

Locate the finder patterns using the following steps:

a) Scan the G channel horizontally, where C1 and C2 have values of alternate 0 and 255 for all four types of finder patterns.

b) When a candidate scanline with the above-mentioned characteristic is detected, check whether the same pattern holds in the B channel at the same position. If not, go to Step d.

c) If the same pattern holds in the B channel, check whether the candidate scanline has 0 values in the R channel. If so, go to Step f.

d) Check whether the same pattern holds in the R channel at the same position.

e) If the same pattern holds in the R channel, check whether the candidate scanline has constant 0 values in the B channel.

f) Note the position A, B, C, and D where the scanline intersects the border between the layers, as shown in Figure 18. Check whether the layer sizes, AB, BC, and CD, are consistent. The tolerance for each layer size is 40 %. If not, revert to Step a and continue scanning.

g) Determine the finder pattern type by the detected core colour C1.

h) Determine the horizontal centre position x of the core module between B and C as shown in Figure 18.

i) Crosscheck the detected pattern at the same centre position in the R, G, and B channels in the horizontal, vertical, and both diagonal directions. The pattern shall be confirmed in a channel when it has been found in at least three directions. If, for each colour channel, the three scans match (horizontal, vertical, and at least one diagonal), save it as a candidate (see Figure 17). If not, revert to Step a and continue scanning.

j) Determine the vertical centre position y of the core module, between E and F, as shown in Figure 18, in the vertical and diagonal scan process.

k) Save the central coordinate (x, y) of the core module, and save the average layer size as the module size.

l) If the detected finder pattern candidate shares the same type, the same centre position and the same layer size with a previously detected candidate, join the two finder patterns together and increase the found-counter $FC_{x,y}$ for this position (x, y) by one. The tolerance for the position and the layer size is 100 % of the detected layer size.

m) Revert to Step a and continue scanning, repeating the above steps until all finder pattern candidates have been identified.

The candidates that have a found-counter less than 3 shall be abandoned. Choose the best finder pattern candidate for each type of UL, UR, LR and LL, which has the highest found-counter $FC_{x,y}$, as the final found finder pattern. If the found-counter of one finder pattern is less than half of the maximal found-counter of all the final found finder patterns, it shall be abandoned.

If no finder pattern is found, the decoding fails.

If one or two finder patterns are found, the decoding will proceed in step v. If only one finder pattern is missing, the central position of its core module shall be estimated by the central positions of the other three finder patterns as a missing corner point of a quadrangle. For instance, if Finder Pattern UL is missing, its central position shall be estimated using the following steps:

n) Calculate the average module size of Finder Pattern LL and LR according to Formula (2).

$$ave\_ms\_ll\_lr = \frac{ll\_module\_size + lr\_module\_size}{2} \tag{2}$$

o) Calculate the average module size of Finder Pattern UR and LL according to Formula (3).

$$ave\_ms\_ur\_ll = \frac{ur\_module\_size + ll\_module\_size}{2} \tag{3}$$

p) Estimate the central coordinates of Finder Pattern UL according to Formula (4). If the estimated central coordinates fall outside of the image in the first detection attempt, proceed to Step v. If it occurs in the second attempt, the decoding fails.

$$\begin{cases} x_{ul} = \dfrac{x_{ll} - x_{lr}}{ave\_ms\_ll\_lr} \times ave\_ms\_ur\_ll + x_{ur} \\ y_{ul} = \dfrac{y_{ll} - y_{lr}}{ave\_ms\_ll\_lr} \times ave\_ms\_ur\_ll + y_{ur} \end{cases} \tag{4}$$

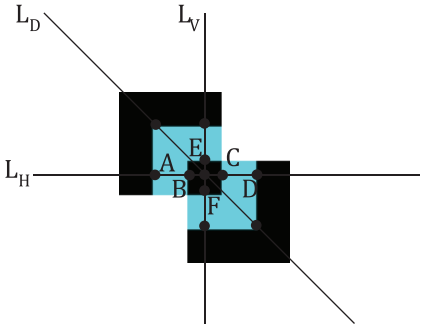


Figure 18 — Scanlines in finder pattern

Search for the missing finder pattern in a local area around the estimated central position using the following steps.

q) Determine the local search area by setting a search radius to five times of the average module size of the other three finder patterns, centred at the estimated position.

r) Classify all the pixels in the local area into three colours: black, cyan, and yellow as follows:

   1) Calculate the average pixel values in the area in the R, G and B channels separately.

   2) If the R, G and B values of a pixel are all smaller than the average in the corresponding channel, the pixel is classified as black, i.e., the RGB values are set to (0, 0, 0).

   3) Otherwise, if the R value is smaller than the B value, the pixel is classified as cyan, i.e., the RGB values are set to (0, 255, 255).

   4) Otherwise, if the R value is bigger than the B value, the pixel is classified as yellow, i.e., the RGB values are set to (255, 255, 0).

s) Scan the local area, following Step 1-13 for locating finder patterns, to search for finder patterns of the missing type.

t) Select the best finder pattern found, which has the highest found-counter $FC_{x,y}$. If several finder patterns are found equally often, then the first finder pattern found shall be selected.

u) If the missing finder pattern is not found, continue the decoding using the estimated values.

If one or two finder patterns are found in the first detection attempt, the following steps shall be performed to start the second attempt.

v) Calculate the average pixel values in R, G and B channels in a local square area, centred at the centre of each found finder pattern, respectively. The side size of the square area shall have a size of 5 times the module size of the corresponding finder pattern.

w) Calculate the average R, G and B values of the average pixel values around each found finder pattern.

x) Repeat steps 1 to 3 in 6.2 to reclassify the colours by replacing the average pixel values of each block with the calculated average R, G and B values in Step 23.

y) Repeat steps a to u in this subclause to locate finder patterns. If not, all finder patterns can be determined in the second attempt, the decoding fails.

After all finder patterns are located, estimate the module size by averaging the module sizes of the four finder patterns.

Subsequently, determine the horizontal and vertical side sizes by dividing the distance between finder patterns by their average module size. The horizontal side size is calculated at both the top and the bottom side, while the vertical side size is calculated at both the left and right side, respectively, as shown Formula (5) to Formula (8).

$$side\_size\_x\_top = floor\left(\frac{dist\_ul\_ur\_x}{(ul\_module\_size + ur\_module\_size)/2 \times cos\theta_1} + 7.5\right) \tag{5}$$

$$side\_size\_x\_bottom = floor\left(\frac{dist\_ll\_lr\_x}{(ll\_module\_size + lr\_module\_size)/2 \times cos\theta_2} + 7.5\right) \tag{6}$$

$$side\_size\_y\_left = floor\left(\frac{dist\_ul\_ll\_y}{(ul\_module\_size + ll\_module\_size)/2 \times cos\theta_3} + 7.5\right) \tag{7}$$

$$side\_size\_y\_right = floor\left(\frac{dist\_ur\_lr\_y}{(ur\_module\_size + lr\_module\_size)/2 \times cos\theta_4} + 7.5\right) \quad (8)$$

where

$dist\_ul\_ur\_x$ is the horizontal distance between Finder Pattern UL and UR

$dist\_ll\_lr\_x$ is the horizontal distance between Finder Pattern LL and LR

$dist\_ul\_ll\_y$ is the vertical distance between Finder Pattern UL and LL

$dist\_ur\_lr\_y$ is the vertical distance between Finder Pattern UR and LR

$ul\_module\_size, ur\_module\_size, ll\_module\_size$ and lr_module_size are the module sizes of Finder Pattern UL, UR, LL and LR, respectively.

And

$$cos\theta_1 = \frac{max(|ul\_x - ur\_x|, |ul\_y - ur\_y|)}{dist\_ul\_ur\_x}$$

$$cos\theta_2 = \frac{\max(|ll\_x - lr\_x|, |ll\_y - lr\_y|)}{dist\_ll\_lr\_x}$$

$$cos\theta_3 = \frac{max(|ll\_x - ul\_x|, |ll\_y - ul\_y|)}{dist\_ul\_ur\_x}$$

$$cos\theta_4 = \frac{\max(|lr\_x - ur\_x|, |lr\_y - ur\_y|)}{dist\_ll\_lr\_x}$$

converts the module size in the scanning direction to the module size along the connection between finder patterns.

The calculated horizontal and vertical side sizes shall be evaluated using the function as shown in Table 24. The side size with a bigger flag value shall be chosen to determine the horizontal and vertical side size. Subsequently, determine the horizontal and vertical side versions by the side sizes.

Table 24 — Determine the side version

```
Start
  Set flag = 1
  Set res = side_size MOD 4
  If res = 0:
        side_size = side_size + 1
  Else if res = 2:
        side_size = side_size – 1
  Else if res = 3:
        flag = 0
        side_size = side_size + 2
End
```

6.4 Locating alignment patterns

For Side-Version 6 or larger symbols, Alignment Patterns X0 and Alignment Pattern X1 are placed between the finder patterns, in horizontal and vertical directions, at intervals defined in Table 2.

Both Alignment Pattern X0 and Alignment Pattern X1 have values of alternate 0 and 255in the R and B channels, and a constant value of 255 in the G channel. The relative widths of each layer are p:1:q in all scanlines running through the centre of the core module. In this reference algorithm, the tolerance for each of these widths is 50 %, corresponding to a range of 0.5 to 1.5.

If the primary symbol is encoded using the default parameters, the first alignment pattern next to Finder Pattern UL or LL in the horizontal direction shall be first located to confirm the horizontal side version, and the first alignment pattern next to Finder Pattern UL or UR in the vertical direction shall be located to confirm the vertical side version.

a) As described in Table 25, estimate the provisional central coordinate and the module size of the first alignment pattern on the guideline AB and EF next to Finder Pattern UL as shown in Figure 20, respectively. The number of modules between Finder Pattern UL and the first alignment patterns shall be determined by the side versions obtained in 6.3 and the coordinates defined in Table 2.

b) Classify all the pixels in the local area with a radius of four times of the module size centred at the estimated central coordinate into the eight colours:

   1) Calculate the average pixel values in R, G, and B channels in a local square area centred at the centre of each found finder pattern, respectively. The side size of the square area shall have a size of 5 times the module size of the corresponding finder pattern.

   2) Calculate the average R, G, and B values of the average pixel values around each found finder pattern.

   3) Repeat Step 1 to 3 in 6.2 to reclassify the colours by replacing the average pixel values of each block with the calculated average R, G, and B values in Step b.2.

c) Scan the local area with a radius of four times of the module size centred at the estimated central coordinate.

   1) Start with horizontal scanning in the R channel.

   2) If an above-mentioned pattern with alternating 0 and 255 is found, crosscheck the same pattern in the R, G, and B channels, in horizontal, vertical and both diagonal directions.

   3) If, for each colour channel, the three test scans (horizontal, vertical, and at least one diagonal) match, save it as a candidate. Otherwise, revert to Step c.1. and continue scanning.

   4) Determine the horizontal centre position x, and the vertical centre position y of the core module.

   5) Determine the layer size of the core.

   6) Save the central coordinate (x, y) of the core module and save the core layer size as the module size.

   7) If a candidate shares the same central coordinate and the same module size with a previously detected candidate, join them together and increase the found-counter $FC_{x,y}$ for this position (x, y) by one. The tolerance for the position, and the layer size is 100 % of the detected layer size.

   8) Revert to Step c.1. and continue scanning until the complete symbol is scanned.

d) After the scanning is done, choose the candidate with the highest $FC_{x,y}$ as the found alignment pattern.

e) If no alignment pattern is found, go to Step g, otherwise, determine the position of the found alignment pattern by the number of modules between Finder Pattern UL and the found alignment pattern along the guideline AB or EF plus four, which is calculated by dividing the distance between them by their average module size. The calculated position shall be adjusted using Formula (9)

$$ap\_pos^{*} = \begin{cases} ap\_pos - 1 & if\ ap\_pos\ mod\ 3 = 0 \\ ap\_pos + 1 & if\ ap\_pos\ mod\ 3 = 1 \\ ap\_pos & if\ ap\_pos\ mod\ 3 = 2 \\ invalid\ value & if\ ap\_pos < 14\ or\ ap\_pos > 26 \end{cases} \quad (9)$$

f) If the position is invalid, go to Step g, otherwise, if the determined position is different from the value defined in Table 2 for the side version obtained in 6.3. The side version shall be updated to the nearest side version whose first alignment pattern has the determined position.

g) If the expected alignment pattern on the guideline AB or EF is not found or the determined position is invalid, update the corresponding side version to the last side version (side version minus one) or the next side version (side version plus one) alternatively and repeat Step a to f. If the expected alignment pattern cannot be identified after five attempts (using five different adjacent side versions), go to Step h.

h) If the first alignment pattern on the guideline AB cannot be identified, try to locate the first alignment pattern on the guideline CD next to Finder Pattern LL and repeat Step a to g. If the first alignment pattern on the guideline EF cannot be identified, try to locate the first alignment pattern on the guideline GH next to Finder Pattern UR and repeat Step a to g. If no valid position of the first alignment pattern in the horizontal or the vertical direction can be determined, the decoding fails.

Locate other alignment patterns using the following steps:

i) Estimate the provisional central coordinate and the module size of Alignment Pattern X0 and X1, as described in Table 25. The coordinate estimation is based on the central coordinates of finder patterns and alignment pattern in the neighbourhood and the lines parallel to the guidelines AB, CD, EF and GH. For example, the lines A1B1, A2B2, C1D1, E1F1, E2F2 and G1H1 as shown in Figure 20. The number of modules between alignment patterns/finder patters shall be determined according to the coordinates defined in Table 2.

j) Use Step b to d to locate each alignment pattern.

k) If the expected alignment pattern is not found, continue decoding using the estimated values.

Table 25 — Estimate the central coordinates of Alignment Patterns X0 and X1

If AP is on the guideline AB: //AP is the alignment pattern to be determined

DistanceX = FP_UR.x – AP_Left.x; //AP_Left is the alignment/finder pattern left to AP

DistanceY = FP_UR.y – AP_Left.y

Alpha = arctan(DistanceY / DistanceX)

Distance = AP_Left.module_size × (the number of Modules between AP_Left and AP)

AP.x = AP_Left.x + Distance × cos(Alpha)

AP.y = AP_Left.x + Distance × sin(Alpha)

AP.module_size = AP_Left.module_size

Else if AP on the guideline EF:

DistanceX = FP_LL.x – AP_Upper.x; //AP_Left is the alignment/finder pattern upper to AP

DistanceY = FP_LL.y – AP_ Upper.y

Alpha = arctan(DistanceY / DistanceX)

Distance = AP_Upper.module_size × (the number of Modules between AP_Upper and AP)

AP.x = AP_Upper.x + Distance × cos(Alpha)

AP.y = AP_Upper.x + Distance × sin(Alpha)

AP.module_size = AP_Upper.module_size

Else:

Ave_Size_ULU = (AP_Upper_Left.module_size + AP_Upper.module_size) / 2

Ave_Size_LU = (AP_Left.module_size + AP_Upper.module_size) / 2

AP.x = (AP_Upper.x – AP_Upper_Left.x) / Ave_Size_ULU × Ave_Size_LU + AP_Left.x

AP.y = (AP_Upper.y – AP_Upper_Left.y) / Ave_Size_ULU × Ave_Size_LU + AP_Left.y

AP.module_size = Ave_Size_LU

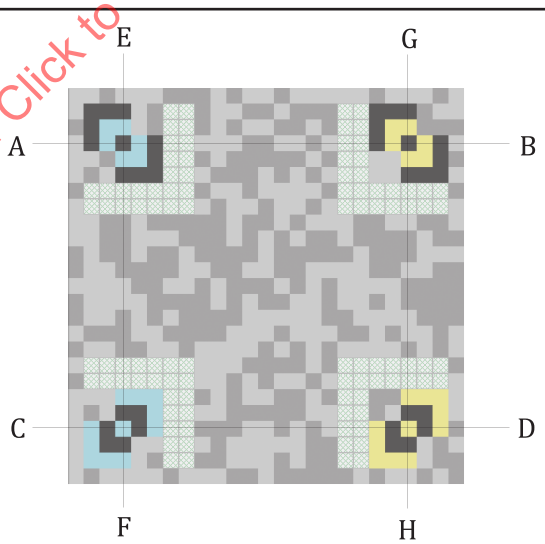


Figure 19 — Finder patterns and guidelines

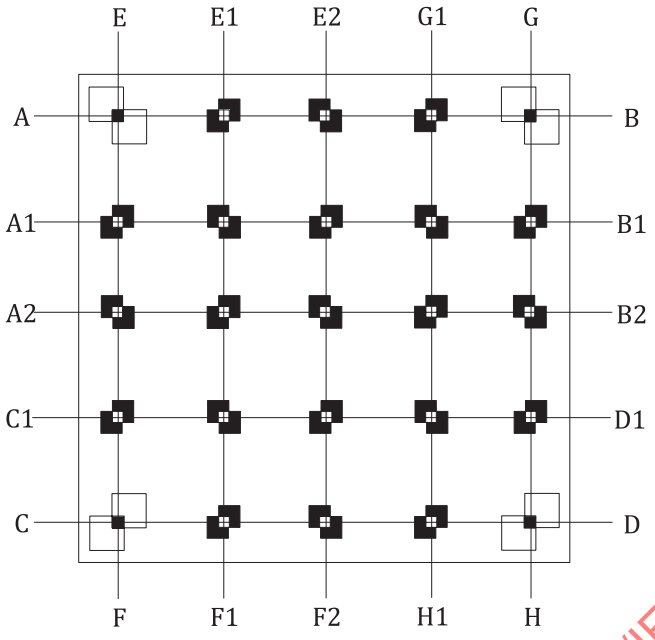


Figure 20 — Finder patterns and alignment patterns with guidelines

6.5 Establishing sampling grid and sampling symbol

The sampling grid shall be established using the following rules:

a) For the symbols with side-version less than 6, in which there exists no alignment pattern, the sampling grid shall be established solely based on the central coordinates of the four finder patterns as shown in Figure 19.

b) For the symbols with alignment patterns, a sampling grid shall be established in each sub-block, divided by finder/alignment patterns, as illustrated in Figure 20. In the case an alignment pattern is not detected, the sampling grid for the corresponding sub-block shall be established using a minimal rectangle containing the sub-block, whose vertices consist of detected alignment/finder patterns. To determine the sampling grid in the sub-blocks at the four corners of the symbol, one alignment pattern shall be replaced by the finder pattern at the corresponding coordinate.

c) Calculate the perspective transformation matrix using the central coordinates of the finder/alignment patterns located at the vertices of a sampling rectangle.

d) Sample each module at the intersection of the grid lines in the sampling grid using the calculated perspective transformation matrix. The pixel values of the R, G, and B channel are sampled respectively using the following Formula (10):

$$M_c(x,y) = \frac{1}{N} \sum_{i,j \in A} P_c(i,j) \tag{10}$$

where

$M_c(x,y)$ is the module colour value in the colour channel c at coordinate (x, y)

$P_c(i,j)$ is the pixel value in channel c at coordinate (i, j) in the image

A defines a local area centred at (x, y) which contains N pixels. The size of A is defined as a 3x3 block in this reference algorithm.

6.6 Decoding metadata and constructing colour palettes

After the primary symbol is sampled, decode the metadata and construct the colour palettes according to the following steps:

a) Read Part I of the metadata at the coordinates defined in Figure 9. If the corresponding modules for Part I contains invalid colour combinations that do not exist in Table 7, continue decoding using the default parameters and skip the metadata decoding in the next steps.

b) Decode Part I and correct errors. If there are too many errors in Part I to correct, the decoding fails.

c) To construct the colour palettes, read the module colours according to the place order and the coordinates as defined in 4.4.4. For each colour palette, two colours shall be read from the corresponding finder pattern modules.

d) Calculate the black thresholds of each colour channel for each colour palette as shown in Table 26.

e) Normalize each colour palette as shown in Table 27.

f) Read the colour of the modules for Part II of the metadata at the coordinates defined in Figure 9.

g) Decode the module data for Part II by determining their colour index in the nearest colour palette, as shown in Table 28.

h) Decode Part II and correct errors to determine the code parameters, including the side-versions, the error correction parameters, and the masking pattern reference.

   1) If there are too many errors to correct, the decoding fails.

   2) If the decoded horizontal or vertical side-version in Part II differs from the detected one, the decoding fails. Reconstruct the sampling grid and resample the symbol using the decoded side version.

   3) If the decoded error correction parameter is invalid, the decoding fails.

Table 26 — Calculate the black threshold for each colour palette

```
Set THS as the black threshold
Set CP as the colour palette   //CP[i][j] is the value of the jth channel of the ith colour in CP
Set CN as the number of module colours
If CN = = 4:
    cpr0 = MAX(CP[c0][red], CP[c1][red])
    cpr1 = MAX(CP[c2][red], CP[c3][red])
    cpg0 = MAX(CP[c0][green], CP[c2][green])
    cpg1 = MAX(CP[c1][green], CP[c3][green])
    cpb0 = MAX(CP[c0][blue], CP[c3][blue])
    cpb1 = MAX(CP[c1][blue], CP[c2][blue])
If CN = = 8:
    cpr0 = MAX(CP[c0][red], CP[c1][red], CP[c2][red], CP[c3][red])
    cpr1 = MAX(CP[c4][red], CP[c5][red], CP[c6][red], CP[c7][red])
    cpg0 = MAX(CP[c0][green], CP[c1][green], CP[c4][green], CP[c5][green])
    cpg1 = MAX(CP[c2][green], CP[c3][green], CP[c6][green], CP[c7][green])
    cpb0 = MAX(CP[c0][blue], CP[c2][blue], CP[c4][blue], CP[c6][blue])
    cpb1 = MAX(CP[c1][blue], CP[c3][blue], CP[c5][blue], CP[c7][blue])
THS[red] = (cpr0 + cpr1) / 2.0
THS[green] = (cpg0 + cpg1) / 2.0
THS[blue] = (cpb0 + cpb1) / 2.0
```

Table 27 — Normalize the colour palettes

```
Set CP as the colour palette
Set CP_Norm as the normalized colour palette
For each colour in CP:
    Set Max as the maximal value in all channels of CP[colour]
    For each colour channel:
        CP_Norm[colour][channel] = CP[colour][channel] / Max
```

Table 28 — Decode module data

```
Set C as the colour of the module to be decoded
Set CP as the nearest colour palette to the module to be decoded
Set CP_Norm as the normalized CP
Set THS as the black thresholds of CP
If C[red] < THS[red] AND C[green] < THS[green] AND C[blue] < THS[blue]:
    Decoded_Colour_Index = 0
Else:
    Set Max as the maximal value in all channels of C
    For each colour channel:
        C_Norm[channel] = C[channel] / Max
    For each colour in CP:
        Diff[colour] = L2(C_Norm – CP_Norm[colour])      //L2 is the Euclidean distance
    Decoded_Colour_Index = Get_Colour_Index_in_CP_with_smallest_Diff(Diff)
    If Decoded_Colour_Index = = 0 OR Decoded_Colour_Index = = 7:
        C_Sum = C[red] + C[green] + C[blue]
        CP_Sum0 = CP[c0][red] + CP[c0][green] + CP[c0][blue]
        CP_Sum7 = CP[c7][red] + CP[c7][green] + CP[c7][blue]
        If C_Sum < (CP_Sum0 + CP_Sum7) / 2:
            Decoded_Colour_Index = 0
        Else:
            Decoded_Colour_Index = 7
```

6.7 Decoding the data stream

Excluding the modules for the finder patterns, alignment patterns, colour palettes and metadata, the data stream is encoded in the rest of the modules. Determine the colour of the module based on the nearest colour palette in the symbol. Decoding the data stream shall proceed using the following steps:

a) Read the colour of the rest of the modules that encodes the data stream.

b) Decode each data module by mapping the module colour into the corresponding bit string, according to its index in the nearest colour palette as showed in Table 28.

c) Based on the mask reference in metadata, apply the used masking pattern to the decoded data modules to release the data masking and restore the original encoded bit strings.

d) Assemble the bit string of each data module to get the data stream.

e) De-interleave the data stream to obtain the encoded data in the original sequence.

f) Based on the decoded error correction parameters from the metadata, detect errors in the data stream and correct them using the LDPC decoder. If there are too many errors to correct, the decoding fails.

g) Decode the data message into the original message in accordance with the encoding mode in use, as described in 5.3.

h) To determine the docking positions of secondary symbols, decode the Part III metadata of a primary symbol, that directly follows the padding bits, if any, and as defined by 3.1.6.

i) If docked secondary symbols exist, decode the Part I and Part II metadata of each docked secondary symbol to determine their parameters, including the side-version of the non-docking side and the error correction parameters.

6.8 Locating and decoding secondary symbols

Based on the decoded Part III metadata, if secondary symbols are docked to the primary symbol, locate and decode them according to the symbol decoding order defined in 4.5.2, and the following steps.

The two Alignment Patterns, U or L, are located beside the two finder patterns at the docking side. The distance between the two finder patterns and the two adjacent alignment patterns in the secondary symbol is fixed at seven modules.

a) Determine the provisional central coordinates of the two Alignment Patterns U or L close to the docking side, based on the central coordinates of the two finder patterns at the docking side and the corresponding guidelines.

b) For horizontally docked secondary symbols, the two Alignment Patterns U or L are located on the extension lines of AB and CD. For vertically docked secondary symbols, they are located on the extension lines of EF and GH.

c) Scan a local area around the provisional central coordinate to locate the Alignment Patterns U and L; the same as locating the alignment pattern X0 and X1 in 6.4.

d) Calculate the distance between the two Alignment Patterns U and L at the docking side, and the other two Alignment Patterns U and L far from the docking side using the decoded side-version of the non-docking side of the secondary symbol.

e) Determine the provisional central coordinates of the other two Alignment Patterns U or L far from the docking side, based on the two detected Alignment Patterns U or L along the extension lines of the guidelines connecting the finder patterns and the Alignment Patterns U or L at the docking side.

f) Scan a local area around the provisional central coordinate to locate the other Alignment Patterns U and L in the same way as in the above step.

g) Locate the other alignment patterns, i.e., Alignment Patterns X0 and X1, if they exist, in the same way as the primary symbol.

h) Establish the sampling grid and sample the symbol in the same way as the primary symbol.

i) Construct the colour palettes at the reserved positions, as defined in 4.4.4. For each colour palette, two colours shall be read from the nearest Alignment Patterns U or L.

j) Decode the data stream in the same way as in the primary symbol.

k) Decode Part III metadata of the secondary symbol following the padding bits $S_{BitData}$ according to 3.1.6 to determine the docking positions of further secondary symbols. If further docked secondary symbols exist, decode Part I and Part II metadata of each further docked secondary symbol in the remaining data stream to determine their parameters.

Repeat the above steps until all the secondary symbols docked to the primary symbol are decoded. If additional docked symbols to the decoded secondary symbols exist, locate and decode all the symbols recursively, according to the decoding order defined in 4.5.2.

7 Transmitted Data

7.1 General principles

All encoded data characters in the primary and secondary symbols shall be included in the data transmission. Symbology control characters and error correction characters are not transmitted.

More complex interpretations are addressed below.

7.2 Protocol for FNC1

When FNC1 precedes the first message character, it signals that the encoded message conforms to the GS1 Applications Identifier standard format. Transmission of symbology identifiers shall be enabled, and this FNC1 shall not be represented in the transmitted data, although its presence shall be indicated by the use of an appropriate option value in the symbology identifier as given in Annex H.

When FNC1 immediately follows a single upper or lower case letter or two digits at the beginning of the message, it signals that the encoded message conforms to a particular industry standard format. Transmission of symbology identifiers shall be enabled, and this FNC1 shall not be represented in the transmitted data, although its presence shall be indicated by the use of an appropriate option value in the symbology identifier. The leading message character(s) shall be transmitted with the encoded message.

7.3 Protocol for ECIs

In systems where ECIs are supported, the use of a symbology identifier prefix is required with every transmission. Whenever an ECI character is encountered, it shall be transmitted as the escape character 92_{DEC} (or $5C_{HEX}$), which represents the character "\" (backslash or reverse solidus) in the default interpretation.

The next codeword(s) are converted into a 6-digit value, inverting the rules defined in Table 19. The 6-digit value is transmitted as the appropriate ASCII values (48-57). Application software recognizing \ nnnnnn shall interpret all subsequent characters as being from the ECI defined by the 6-digit sequence. This interpretation remains in effect until the end of the encoded data or until another ECI sequence is encountered.

If the backslash (byte $5C_{HEX}$) needs to be used as encoded data, two bytes of that value shall be transmitted. Thus, a single occurrence is always an escape character and a double occurrence indicates true data.

EXAMPLE

Encoded data: A\\B\C

Transmission: A\\\\B\\C

Use of the appropriate symbology identifier ensures that the application can correctly interpret the escape character.

7.4 Symbology identifier

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

Once the structure of the data (including the use of any ECI) has been identified, the appropriate symbology identifier should be added by the decoder as a preamble to the transmitted data. The symbology identifier is required if ECIs appear anywhere in the symbol, or if FNC1 is used as defined in 7.2. See Annex H for the symbology identifier and option values, which apply to JAB Code.

8 JAB-Code symbol quality

8.1 Symbol quality evaluation

To evaluate the JAB Code symbol quality, follow the method defined in ISO/IEC 15415 for grading 2D bar code symbols; some supplements are required and described in detail in 8.2.

To verify a JAB Code symbol, first run the reference decoding algorithm specified in 6. The scan grade shall be the lowest of the grades for decode, unused error correction, grid non-uniformity, fixed pattern damage, colour palette accuracy, and colour variation in data modules.

8.2 JAB-Code verification parameter according to ISO/IEC 15415

8.2.1 Decode

Decode shall be graded with 4 if the symbol can be decoded successfully by applying the reference decoding algorithm in 6, otherwise it shall be graded 0.

8.2.2 Unused Error Correction

The Unused Error Correction, UEC, shall be graded using the following formulae:

UEC < = 0.125 Grade = 0.0

0.125 < UEC < 0.625 Grade = round(80 * UEC) –1

UEC > = 0.625 Grade = 4.0

where: round(x) rounds down to the next 0,1 of x.

The amount of the UEC is calculated as

$UEC = 1 - (e + 2 * t) / (P_g * E_{cap})$,

where:

e is the number of erasures;

t is the number of errors;

E_{cap} is the error correction capacity listed in Table 29;

P_g is the gross message length.

Table 29 — Error recovery capacity in % for LDPC Code

Level	E_{cap} in %
1	8
2	9
3	10
4	11
5	14
6	17
7	22

Table 29 *(continued)*

Level	E_{cap} in %
8	24
9	26
10	29

8.2.3 Grid non-uniformity

The ideal grid shall be established by using the symbols edges as reference points and subdivide the symbol in both axes equally. To grade the deviation from the ideal grid, the boundary from each module to its neighbouring module shall be determined. For this purpose, two neighbouring grid intersections, as defined in 6.5, are considered. On their connecting line, the colour transition in one of the three colour channels, that results, according to 6.2, is determined. The distance from the theoretical to the actual grid position shall be determined by the absolute distances of both values. The largest of all distances, GN, shall be used in the following formulas. If two neighbouring modules have the same colour, then no colour transition is present and is not included in the evaluation.

GNU > = 0.875 Grade = 0.0

0.875 > GNU > 0.375 Grade = 7-round (80*GNU)

GNU < = 0.375 Grade = 4.0

where: round(x) rounds down to the next 0,1 of x.

8.2.4 Fixed Pattern Damage

The patterns to be assessed are the four finder patterns at the corner, and the alignment patterns (where present, in Version 6 or larger). The four finder patterns shall be assessed as four segments, A1, A2, A3, A4. All of the alignment patterns together shall be assessed as a single segment, B.

Damage to each segment shall be graded according to the individual modules from which it is caused.

The procedure described below shall be applied to the fixed pattern in turn.

Based on the steps described in 6.2, each finder pattern is evaluated in each colour channel, separately, as gray-scale image. The scan profiles shall be measured in three directions, horizontal, vertical, and diagonal, relative to the symbol's orientation through the centre of the finder patterns, as shown below.

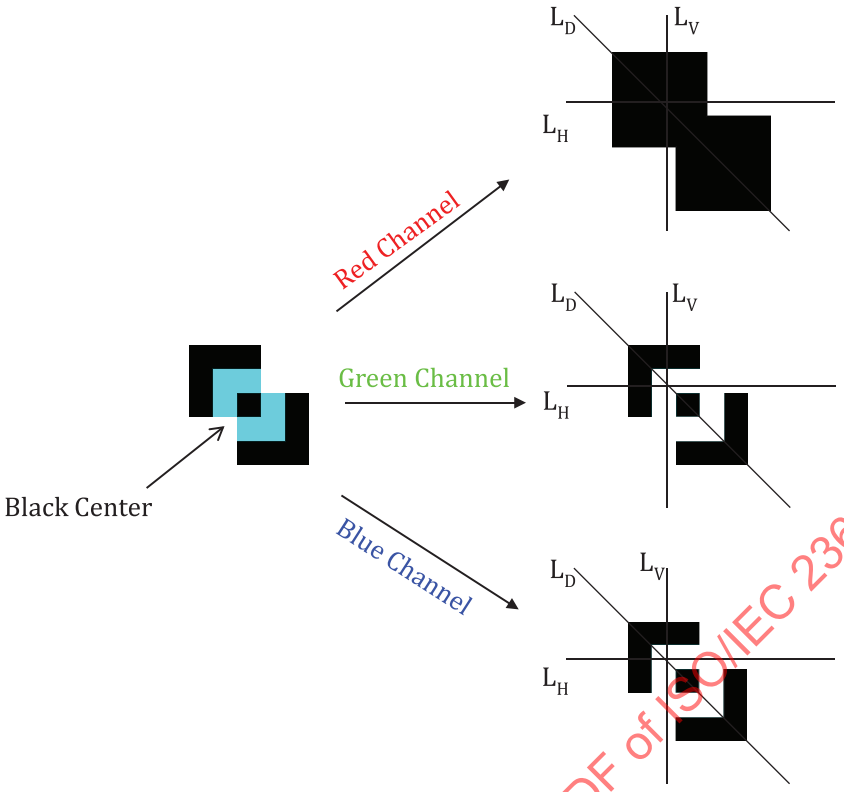


Figure 21 — Upper left finder pattern in its three colour channels

For each colour channel, the relation between the width of the centre and its surrounding layer, CSL, shall be calculated for the three directions - horizontal, vertical, and diagonal, according to the following formula. If the regions have one gray-scale colour, as in the red channel of Figure 21, the calculations shall be skipped:

$$abs(CSL) = (Avg(R_{SL})-R_C)/ (Avg(R_{SL})+R_C),$$

where:

$Avg(R_{SL})$	is the average width of the surrounding layer
R_C	is the width of the centre
abs()	gives the absolute value.

The highest absolute value shall be graded according to the following formula:

abs(CSL) < 0.1 Grade = 4.0

0.1 < = abs(CSL) < = 0.20 Grade = 8 – round(40 * abs(CSL))

abs(CSL) > 0.20 Grade = 0.0,

where: round(x) rounds down to the next 0,1 of x.

Since the intended colour of each module is known in the finder pattern and alignment pattern, any module colour with a distance greater than the global threshold shall be counted as a module error. The global threshold $T_g = 100$ shall be used.

The Euclidean distance is used as metric in the RGB colour space:

$$T_g = \text{sqrt}((R-R_{int})^2 + (G-G_{int})^2 + (B-B_{int})^2),$$

where:

sqrt() means the square root and

R_{int}, G_{int}, B_{int} are the intended colour of fixed pattern modules

T_g is calculated for all pixels of the three directions horizontal, vertical, and diagonal relative to the symbol's orientation.

1) For each Segment A1, A2, A3, and A4, count the number of module errors, separately, and grade according to Table 30.

Table 30 — Grade thresholds for Segments A1 to A4

Finder Pattern Segments A1, A2, A3, A4	Grade
Number of module errors	
0	4
1	3
2	2
3	1
> 3	0

2) For Segment B, count the number of alignment patterns containing a module error. Express this number as a percentage of the number of alignment patterns in the symbol and grade according to the following paragraph.

The grade for segment B shall be computed as a linearly interpolated value, rounded down to the next 0,1 of a grade level. The continuous grade for the fixed pattern damage of segment B FPDB shall be evaluated by the following formula:

FPDB = 0 % Grade = 4.0

0 % < FPDB < 40 % Grade = round(4.0-FPDB)

FPDB > 40 % Grade = 0.0,

where round(x) rounds down to the next 0,1 of x.

For example, an alignment pattern damage of 6 % gets a grade of 3,4 and an alignment pattern damage of 22 % gets a grade of 1,8.

The resulting Fixed Pattern Damage grade for the symbol shall be the lowest of the segment grades.

8.2.5 Symbol contrast, modulation and reflectance margin

This document deviates from ISO/IEC 15415 in grading symbol contrast, modulation and reflectance margin and shall not be graded according to it.

Symbol contrast is specified in 8.3.1. Reflectance margin is specified in 8.3.2 and 8.2.3.

8.3 JAB-Code colour verification

8.3.1 Colour Palette Accuracy

Since the colour placement is known in the colour palette, the colour accuracy of the palette (CPA) shall be measured by calculating the Euclidean distance between the intended colours and the actual colours, with each channel being normalized depending on the number of module colours:

$$CPA = sqrt(((R_{Col} - R)/d_R)^2 + ((G_{Col} - G)/d_G)^2 + ((B_{Col} - B)/d_B)^2),$$

where:

R_{Col}, G_{col} and B_{col} are the three channels of each module in the colour palette

d_R, d_G, d_B is half the distance to the next colour in this colour channel (see Annex G.1).

Grade the colour palette accuracy according to the following formula. The lowest grade shall be used as the result.

$CAP < 0.2$ Grade = 4.0

$0.2 <= CAP < 0.75$ Grade = 5,6 – round(80 * CAP)

$CAP >= 0.75$ Grade = 0.0,

where: round(x) rounds down to the next 0,1 of x.

8.3.2 Colour Variation in Data Modules

The colour of the data modules shall be clustered depending on the number of module colours in the colour palette. The following steps shall be utilized:

1) In a first step, the colour of the data modules shall be clustered depending on the number of module colours in the colour palette.

2) Measure the Euclidean distance (CVDM) of each module to its reference colour, which is the corresponding colour of the colour palette:

$$CVDM_{Mod} = sqrt(((R_{Col} - R_{Mod})/d_R)^2 + ((G_{Col} - G_{Mod})/d_G)^2 + ((B_{Col} - B_{mod})/d_B)^2),$$

where:

R_{Col}, G_{Col}, B_{Col} is the colour of the reference module in the colour palette

R_{Mod}, G_{Mod}, B_{Mod} is the module colour of each module

d_R, d_G, d_B is half the distance to the next colour in this colour channel, see Annex G.1.

3) $CVDM_{Var}$ is calculated by

$$CVDM_{Var} = 1 / n_{DataMod} \sum_{j \in J} (CVDM_{Mod}^{j} - CVDM_{Avg})^2,$$

where: $n_{DataMod}$ is the number of data modules.

Calculate the variance $CVDM_{\mathbf{Var}}$ for each cluster and grade each cluster separate by means of the formula:

$CVDM_{Var} < 0.02$ Grade = 4.0

$0.02 <= CVDM_{Var} < 0.1$ Grade = 5 - round(500 * $CVDM_{\mathbf{Var}}$)

$CVDM_{Var} >= 0.1$ Grade = 0.0,

where: round(x) rounds down to the next 0,1 of x.

To grade the colour variation of the data modules, the lowest cluster grade shall be used.

Annex A
(informative)

User guidelines

A.1 User selection of module colour

In the JAB Code, two module colour modes are specified, which allow a symbol to contain up to 8 different module colours. Using more module colours in a symbol allows higher data capacity, but it also puts higher requirements upon the technologies used to produce and read the symbol. The selection of module colours should be determined in relation to:

— the required data payload, according to the application requirements;

— the expected symbol size, according to the application requirements;

— the capability of the technologies used to produce and scan the symbol.

See Annex G for guidelines on module colour specifications.

A.2 User selection of error correction level

The user should define an appropriate error correction level according to the application requirements. As specified in 4.4.1.4, JAB Code allows customizable error correction levels. For a given message length, a higher level of error correction will lead to some increase in symbol size. The recommended error correction level for normal use should be set as the default level in the encoder and decoder.

If the symbol size is fixed in the application, regardless of the message length, the highest possible error correction level should be used that achieve the best robustness.

A.3 User selection of symbol and code shape

JAB Code allows square and rectangle symbols, and arbitrary code shapes by symbol cascading. The user should define the appropriate symbol shape and code structure to suit the application requirements.

The symbol shape should be determined by the shape of the space in which the symbol is to be applied. In case of non-square placing space, rectangle symbols can be used to accommodate more data than square symbols by making the most of the available space.

With the same symbol size, secondary symbols may accommodate more data than primary symbols, as they have lower overhead, thanks to absence of finder patterns, and shorter metadata. However, symbol cascading increases the reading complexity of JAB Code, and may consequently decrease decoding reliability. Therefore, symbol cascading should only be used in the following cases:

— the data message cannot be accommodated by a single primary symbol;

— the available space to place the code has an irregular shape, which cannot be fully utilized by a single square or rectangle symbol,

— small symbols (small side-version) are preferred due to the application requirements.

A.4 Guidelines for symbol print and scan

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

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

— Select a print density, which will yield tolerance values that can be achieved by the marking or printing technology being used. Ensure that the module dimension is an integer multiple of the print head pixel dimension.

— Choose a reader with a resolution compatible with the symbol density and quality produced by the printing technology.

— Ensure that the optical properties of the printed symbol are compatible with the wavelength of the scanner light source or sensor. Colours as used here are defined in hue and intensity in[5].

— Ensure that the lighting condition is consistent over the whole symbol when scanning the printed symbol. A colour temperature of 6500k for the lighting is recommended.

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

A.5 Autodiscrimination capability

JAB Code primary symbols may be quickly discriminated from other symbologies. When a JAB Code is read by suitably programmed decoders which have been designed to autodiscriminate different symbologies, it signifies that a symbology other than JAB Code is present if less than three out of the four types of finder patterns are detected.

Annex B
(informative)

Error detection and correction

B.1 Error correction encoding

The Low-Density Parity Check Code (LDPC) shall be used for error correction. The parity check matrix $H_{P_n}\left(C^T|I\right)\in\left(K\times P_g\right)$ in systematic form for each length of the message length P_n, is defined by the three parameter w_c and w_r and K. The parameter w_c describes the number of '1's in each column and w_r the number of '1's in each rows. With w_c and w_r, the rate R of the code is determinable with $R = 1\text{-}w_c/w_r$.

One way to generate the matrix H, is to randomly fill the columns with w_c '1's, such that w_r comprises the identity matrix I. It is recommended to select $w_c \geq 3$ and $w_r \geq w_c+1$.

The generator matrix $G_{P_n}\left(I|C\right)\in\left(P_n\times P_g\right)$ is obtained by the matrix H and only shown for the first two metadata lengths. The codeword c is obtained by matrix multiplication in the GF(2), $c = m\otimes G$.

EXAMPLE Given the parameters $w_c = 3$, $w_r = 6$, K = 5 and the message m = [1 0 1 0 1], the generated matrix H and G are:

$$H_5=\begin{bmatrix}1&0&1&1&0&1&0&0&0&0\\1&1&0&0&1&0&1&0&0&0\\0&1&1&1&0&0&0&1&0&0\\0&1&1&0&1&0&0&0&1&0\\1&0&0&1&1&0&0&0&0&1\end{bmatrix}\rightarrow G_5=\begin{bmatrix}1&0&0&0&0&1&1&0&1&0\\0&1&0&0&0&1&0&1&0&1\\0&0&1&0&0&0&1&1&0&1\\0&0&0&1&0&0&1&1&1&0\\0&0&0&0&1&1&0&0&1&1\end{bmatrix}$$

The final codeword is obtained by $c = m\otimes G = [1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0]$.

B.2 Error correction decoding

B.2.1 Error detection and correction with soft decision

The error correction for the metadata in JAB Code shall be computed using an iterative Log Likelihood decoding algorithm for binary LDPC codes. After releasing the masking, the error correction shall be performed.

The Log Likelihood decoding requires the matrix $H_{P_n}\left(C^T|I\right)\in\left(K\times P_g\right)$ used for encoding, the received codeword r (after releasing the masking) and the maximum number of iterations L.

The minimal Hamming distance d_{min} induced by the used matrix H gives the number of detectable errors and the number of correctable errors $\lfloor(d_{min}-1)/2\rfloor$.