
**Information technology — Automatic
identification and data capture
techniques — Han Xin Code bar code
symbology specification**

*Technologies de l'information — Techniques d'identification et de
capture de données automatiques — Spécification des symboles du
code à barres de Han Xin*



Reference number
ISO/IEC 20830:2021(E)

Documents
Document Sharing Hub
© ISO/IEC 2021



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2021

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

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	vi
Introduction	vii
1 Scope	1
2 Normative references	1
3 Terms, definitions, and symbols	1
3.1 Terms and definitions	1
3.2 Mathematical and logical symbols	3
4 Symbology description	4
4.1 Symbology characteristics	4
4.1.1 Basic characteristics	4
4.1.2 Summary of additional features	5
4.2 Symbol structure	5
4.2.1 General	5
4.2.2 Symbol Versions and Sizes	6
4.2.3 Finder Pattern	8
4.2.4 Position Detection Pattern separator	10
4.2.5 Alignment Pattern	10
4.2.6 Assistant Alignment Pattern	13
4.2.7 Structural Information Region	13
4.2.8 Data Region	14
4.2.9 Quiet Zone	14
5 Requirements	15
5.1 Encode procedure overview	15
5.2 Data analysis	16
5.3 Mode	17
5.3.1 General	17
5.3.2 Numeric mode	19
5.3.3 Text mode	19
5.3.4 Binary byte mode	19
5.3.5 Common Chinese Characters in Region One mode	19
5.3.6 Common Chinese Characters in Region Two mode	20
5.3.7 GB18030 2-byte Region mode	20
5.3.8 GB18030 4-byte Region mode	20
5.3.9 Extended Channel Interpretation (ECI) mode	20
5.3.10 Unicode mode	20
5.3.11 GS1 mode	20
5.3.12 URI mode	21
5.4 Data encoding	21
5.4.1 General	21
5.4.2 Constructing the information bit stream	21
5.4.3 Constructing information codewords sequence	21
5.4.4 Numeric mode encoding	21
5.4.5 Text mode encoding	23
5.4.6 Binary mode encoding	24
5.4.7 Common Chinese Characters in Region One mode encoding	25
5.4.8 Common Chinese Characters in Region Two mode encoding	26
5.4.9 GB18030 2-byte Region mode encoding	27
5.4.10 GB18030 4-byte Region mode encoding	28
5.4.11 ECI mode encoding	29
5.4.12 Unicode mode	30
5.4.13 GS1 mode encoding	33
5.4.14 URI mode	36

5.4.15	Mixed modes encoding.....	41
5.5	Error detection and correction.....	42
5.5.1	General.....	42
5.5.2	Generating the error correction codewords.....	42
5.5.3	Error correction capacity.....	44
5.6	User considerations for encoding data in a Han Xin Code symbol.....	44
5.6.1	General.....	44
5.6.2	User selection of error correction level.....	44
5.6.3	User selection of mode.....	44
5.6.4	User selection of Extended Channel Interpretation.....	45
5.6.5	User selection of symbol size.....	45
5.7	Construction of final data bit stream.....	45
5.8	Symbol construction.....	45
5.8.1	General.....	45
5.8.2	Fixed Pattern placement.....	45
5.8.3	Data placement.....	46
5.8.4	Masking.....	48
5.8.5	Structural Information placement.....	49
6	Symbol dimensions.....	50
6.1	Dimensions.....	50
6.2	Quiet zone.....	50
7	User guidelines.....	50
7.1	Human readable interpretation.....	50
7.2	Autodiscrimination capability.....	51
7.3	Principle of Han Xin Code symbol printing and scanning.....	51
8	Symbol quality.....	51
8.1	General.....	51
8.2	Symbol quality parameters.....	51
8.2.1	General.....	51
8.2.2	Fixed Pattern damage.....	51
8.2.3	Symbol grade.....	51
8.2.4	Grid non-uniformity.....	51
8.3	Process control measurements.....	51
9	Decoding procedure overview.....	52
10	Reference decode algorithm for Han Xin Code.....	53
10.1	General.....	53
10.2	Image preprocessing.....	53
10.3	Locate Finder Pattern and determine the orientation.....	53
10.4	Structural Information decoding.....	55
10.5	Establish the sampling grid.....	56
10.6	Sampling.....	63
10.7	Masking releasing.....	63
10.8	Restore data codewords.....	64
10.9	Error correction decoding.....	64
10.10	Data codeword decoding.....	64
11	Transmitted data.....	64
11.1	General.....	64
11.2	Basic interpretation.....	64
11.3	Protocol for Extended Channel Interpretation.....	64
11.4	Protocol for GS1 data transmission.....	65
	Annex A (normative) Alignment Pattern parameters of symbol of different versions.....	66
	Annex B (normative) Data capacity and error correction characteristics of Han Xin Code.....	69
	Annex C (informative) Information capacity of Han Xin Code.....	84

Annex D (normative) Error correction codeword generator polynomials	93
Annex E (normative) Structural Information	95
Annex F (informative) Autodiscrimination compatibility	98
Annex G (informative) Error correction decoding algorithm	99
Annex H (informative) User guidance for Han Xin Code printing and scanning	101
Annex I (normative) Print quality of Han Xin Code — Symbology-specific aspects	103
Annex J (informative) Useful process control techniques	107
Annex K (informative) Han Xin Code encoding examples	108
Annex L (informative) Symbology identifier	129
Annex M (normative) Charsets of URI mode	130
Annex N (normative) Source codes for Unicode mode in C programming	137
Annex O (informative) Implement Source code for URI mode in C programming	161
Bibliography	196

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

Han Xin Code is a two-dimensional matrix symbology which is made up of an array of nominally square modules arranged in an overall square pattern, including a Finder Pattern located at four corners of the symbol that are intended to assist in easy locating of its position, size and inclination. Alignment Patterns and Assistant Alignment Patterns are also used in Versions 4 to 84 symbols. A wide range of size of symbols is provided together with four error correction levels. Module dimension is user-specified to produce symbols by a wide variety of techniques.

Manufacturers of bar code equipment and users of the technology require publicly available standard symbology specifications to which they can refer when developing equipment and application standards. This document is published to meet this request.

Information technology — Automatic identification and data capture techniques — Han Xin Code bar code symbology specification

1 Scope

This document defines the requirements for the symbology known as Han Xin Code. It specifies the Han Xin Code symbology characteristics, data encoding process, symbol structure, dimensions and print quality requirements, error correction rules, reference decoding algorithm, and user-selectable application parameters.

2 Normative references

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

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

ISO/IEC 15415:2011, *Information technology — Automatic identification and data capture techniques — Bar code symbol print quality test specification — Two-dimensional symbols*

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

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

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

GS1 General Specifications

3 Terms, definitions, and symbols

3.1 Terms and definitions

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

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

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <https://www.electropedia.org/>

3.1.1

assistant alignment pattern

non-data pattern located in the perimeter of the Han Xin Code symbol and intended to provide additional reference points to synchronize modules for reading

3.1.2

data bit stream

binary sequence that comprises the *information bit stream* ([3.1.9](#)) and the *error correction bit stream* ([3.1.4](#))

3.1.3

data codeword

codeword that is used to encode information codewords and error correction codewords.

3.1.4

error correction bit stream

binary sequence used to correct errors, made by error correction encoding from the *information bit stream* ([3.1.9](#))

3.1.5

Unicode

international encoding standard for use with different languages and scripts, by which each letter, digit, or symbol applies across different platforms and programs.

Note 1 to entry: The repertoire of Unicode is synchronized with ISO/IEC 10646, and both are code-for-code identical.

3.1.6

GS1 mode

encoding *mode* ([3.1.10](#)) used for representing GS1 data in Han Xin Code

3.1.7

URI mode

encoding *mode* ([3.1.11](#)) used for representing uniform resource identifier (URI) described in RFC 3986 in Han Xin Code

3.1.8

Unicode mode

encoding *mode* ([3.1.11](#)) used for representing text data in *Unicode* ([3.1.5](#)) encoding/charset in Han Xin Code.

3.1.9

information bit stream

binary sequence made up of *mode* ([3.1.11](#)) encodings from the original input data

3.1.10

masking

XOR processing of the bit pattern in the information encoding region of the symbol with an algorithmically determined pattern to provide a symbol with more evenly balanced numbers of dark and light modules and to reduce the occurrence of patterns which would interfere with fast processing of the image

3.1.11

mode

method of representing a specific character set as a binary bit stream

3.1.12

mode indicator

bit sequence indicating in which *mode* ([3.1.11](#)) the following data sequence is encoded

3.1.13

mode terminator

bit sequence used to terminate the bit sequence representing an encoding *mode* ([3.1.11](#))

3.1.14**padding bit**

bit "0", appended to the *information bit stream* (3.1.9) to meet the requirements of the error correction algorithm

3.1.15**position detection pattern**

one of the four pattern components of the Finder Pattern in Han Xin Code symbols

3.1.16**position detection center**

center of the 3×3 dark modules in the *position detection pattern* (3.1.15)

3.1.17**position detection pattern separator**

one-module wide non-data pattern, made up of all light modules, used to separate the *position detection pattern* (3.1.15) from the *structural information* (3.1.18) region

3.1.18**structural information**

bit stream of data used to record *version* (3.1.20), error correction level and *masking* (3.1.10) solution

3.1.19**symbol padding bit**

bit "0", not representing data, used to fill the empty positions of the symbol when the information encoding region cannot be fully filled with 8-bit codewords

3.1.20**version**

size of the symbol represented in terms of its position in the sequence of permissible sizes for Han Xin Code symbols, from 23×23 modules (Version 1) to 189×189 modules (Version 84)

3.2 Mathematical and logical symbols

d	number of error correction codewords
e	number of erasures
k	total number of information codewords
n	total number of data codewords
t	number of errors
X	horizontal width of a module
Y	vertical distance from the center line of modules in one row to the center line of modules in an adjacent row
$(\dots)_{\text{bin}}$	data in () is figured in binary system
$\dots_{\text{HEX}}$	data is figured in hexadecimal
$(\dots)_{\text{hex}}$	data in () is figured in hexadecimal

div	is the integer division operator
mod	is the integer remainder after division
XOR	is the exclusive-or logic function whose output is one only when its two inputs are not equivalent

NOTE Without any specific statement, a byte usually comprises 8 binary bits and the byte's contents are represented in hexadecimal.

4 Symbology description

4.1 Symbology characteristics

4.1.1 Basic characteristics

4.1.1.1 General

Han Xin Code is a two-dimensional matrix symbology with the following basic characteristics:

4.1.1.2 Encodable characters

- (1) Numeric characters (digits 0~9)
- (2) ASCII characters (refer to ISO/IEC 646)
- (3) Chinese characters (refer to GB18030)
- (4) Octet bytes such as graphic and audio information, etc.
- (5) GS1 data used in GS1 system
- (6) Uniform Resource Identifier (URI)
- (7) Any text data reference to a encoding/charset (such as Unicode, JIS, etc.)

4.1.1.3 Representation of data

A dark module is a binary one and a light module is a zero, However, dark module is zero and a light module is one for the reflectance reversal symbols. See [4.1.2](#) for details of reflectance reversal.

4.1.1.4 Symbol size in modules

23 modules × 23 modules to 189 modules × 189 modules (Version 1 to 84, increasing in steps of two modules per side).

4.1.1.5 Maximum data capacity

- (1) Numeric data: 7827 characters
- (2) ASCII characters: 4350 characters
- (3) Common Chinese Characters in Regions One and Two of GB18030: 2174 characters
- (4) 2-byte Chinese characters data: 1739 characters
- (5) 4-byte Chinese characters data: 1044 characters
- (6) Binary byte data: 3261 bytes

4.1.1.6 Selectable error correction

Four levels of Reed-Solomon error correction and their error correction abilities of symbol codewords are shown as follows:

L1	8 %
L2	15 %
L3	23 %
L4	30 %

4.1.2 Summary of additional features

The following are additional inherent and optional features of Han Xin Code:

- Masking: (Inherent) Masking pattern is used to the proportions of dark modules and light modules in the symbols and decrease the occurrences of images preventing fast processing.
- Reflectance reversal: (Inherent) Symbols are intended to be read when marked so that the image is either dark on light or light on dark (see [Figure 1](#)).
- Extended Channel Interpretations: (Optional) This mechanism enables characters from other character sets (e.g. Arabic, Cyrillic, Greek, Hebrew) and other data interpretations or industry-specific requirements to be represented.

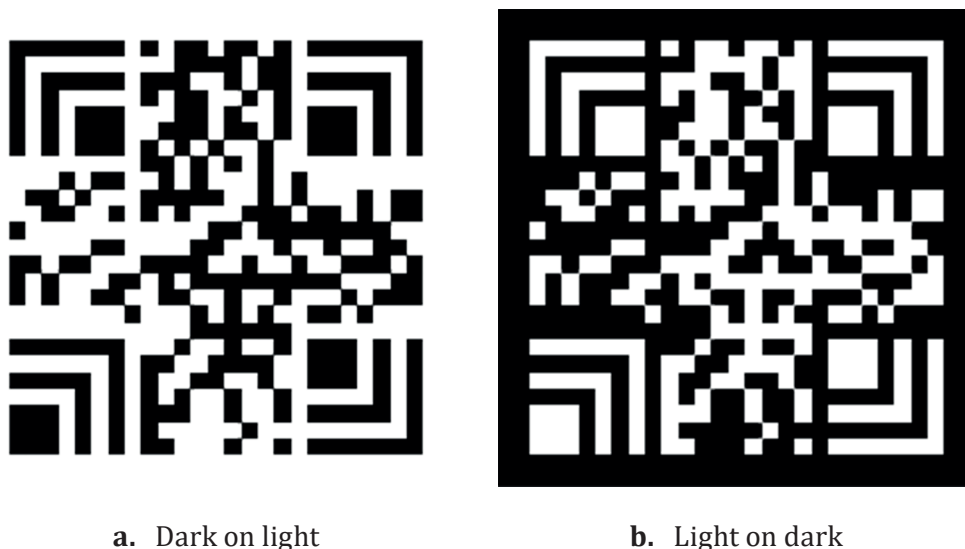


Figure 1 — Examples of Hanxin Code symbol encoding the text "Han Xin Code Symbol"

4.2 Symbol structure

4.2.1 General

Each Han Xin Code symbol shall be constructed of $n \times n$ nominally square modules set out in a regular square array and shall consist of an information encoding region, Structural Information regions and Fixed Pattern region. Fixed Pattern region include Finder Pattern, Position Detection Pattern separators, Alignment Pattern and Assistant Alignment Patterns. The symbol shall be surrounded by a quiet zone. [Figure 2](#) illustrates a Version 24 Han Xin Code symbol. [Figure 3](#) illustrates the structure of a Version 24 symbol.



Figure 2 — Han Xin Code symbol (Version 24)

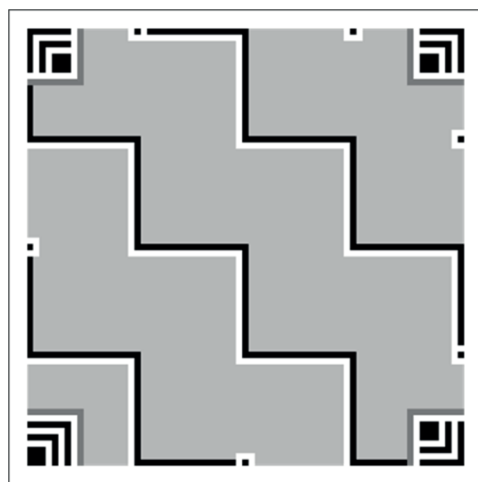


Figure 3 — Structure of a Version 24 Han Xin Code symbol

4.2.2 Symbol Versions and Sizes

There are eighty-four sizes of Han Xin Code symbol referred to as Version 1, Version 2 ... Version 84 respectively. Version 1 measures 23 modules $\times$ 23 modules, Version 2 measures 25 modules $\times$ 25 modules and so on, increasing in steps of 2 modules per side up to Version 84 which measures 189 modules $\times$ 189 modules. [Figure 4](#) to [Figure 9](#) illustrate the symbols of Versions 1, 4, 24, 40, 62 and 84.



Figure 4 — Version 1 symbol



Figure 5 — Version 4 symbol



Figure 6 — Version 24 symbol

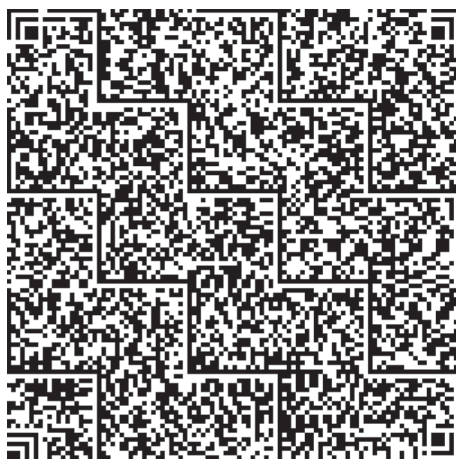


Figure 7 — Version 40 symbol

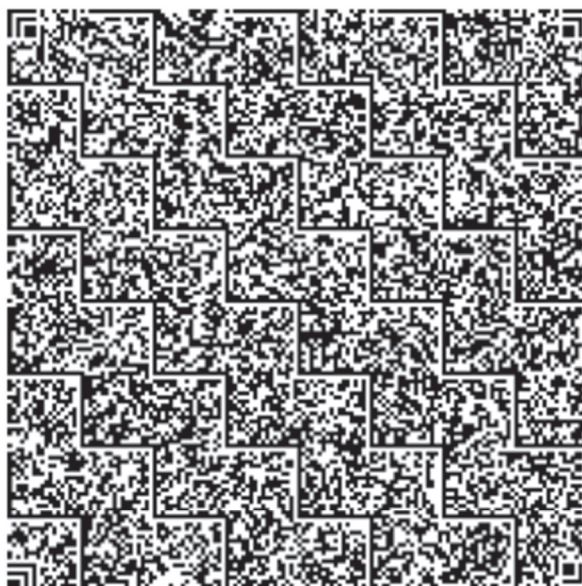


Figure 8 — Version 62 symbol

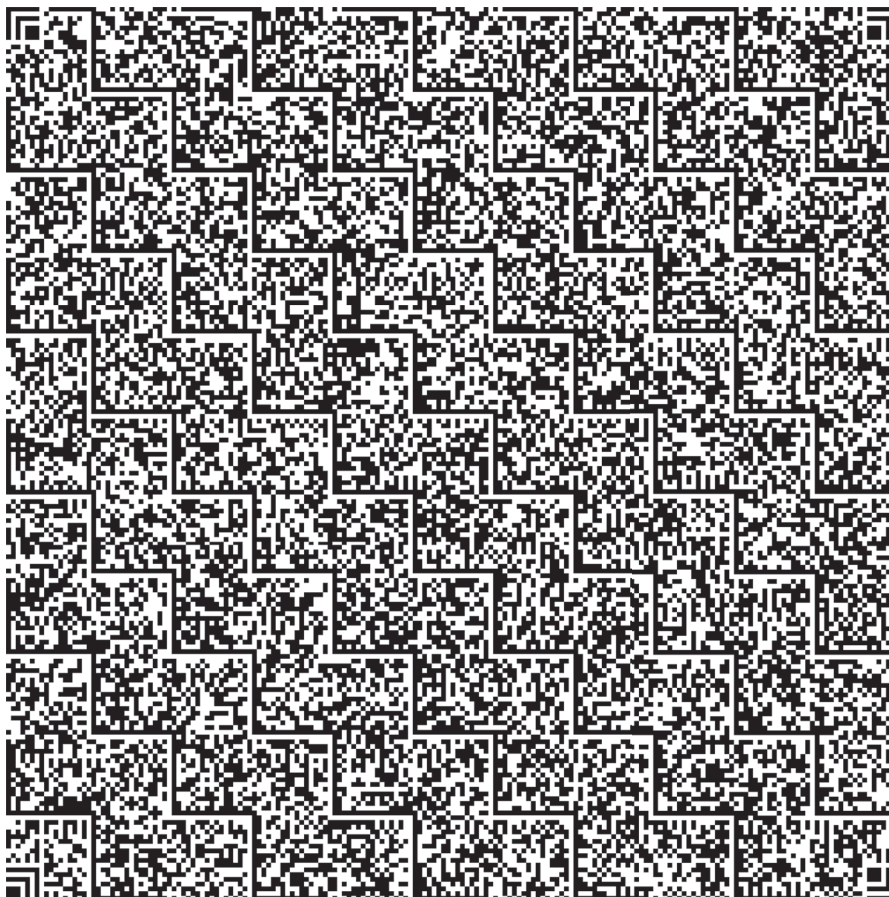


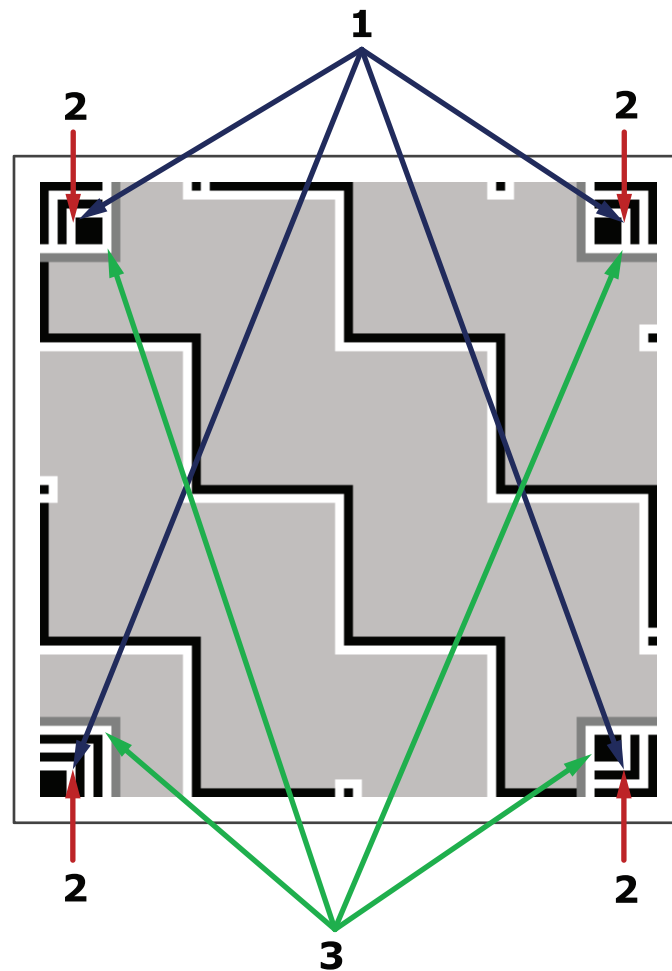
Figure 9 — Version 84 symbol

4.2.3 Finder Pattern

The Finder Pattern is made up of four Position Detection Patterns located at the four corners of the symbol respectively, which is illustrated in [Figure 10](#).

The size of Position Detection Pattern is 7×7 modules. The Position Detection Pattern at the upper left corner is constructed by five aligned squares, whose lower right corners are superposed. The squares are constructed of dark 7×7 modules, light 6×6 modules, dark 5×5 modules, light 4×4 modules, dark 3×3 modules respectively. The other Position Detection Patterns are obtained by rotating the upper left one, as illustrated in [Figure 10](#). The scanning ratio of each Position Detection Pattern is 1:1:1:1:3 or 3:1:1:1:1 (scanning along different directions) as illustrated in [Figure 11](#). The center of dark 3×3 modules in Position Detection Pattern is called the Position Detection Center. Similar patterns have a low probability of being encountered elsewhere in the symbol, so that identification of the four Position Detection Patterns can define unambiguously the location and rotational orientation of the symbol in the field of view.

The shapes of all Position Detection Patterns are the same except their placement directions. [Figure 11](#) is the Position Detection Pattern of the upper left corner. The lower left corner Position Detection Pattern for the upper left corner Position Detection Pattern by clockwise rotation of 90 degrees, the lower right corner Position Detection Pattern for the lower left corner Position Detection Pattern by clockwise rotation of 90 degrees, the upper right corner Position Detection Pattern for the lower right corner Position Detection Pattern by counterclockwise rotation of 90 degrees.

**Key**

- 1 Finder Pattern
- 2 Position Detection Pattern
- 3 Position Detection Pattern separator

Figure 10 — Finder Pattern of Han Xin Code symbol**Figure 11 — Structure of Position Detection Pattern**

4.2.4 Position Detection Pattern separator

A one-module wide separator, constructed of all light modules, is placed between each Position Detection Pattern and the Structural Information region, as illustrated in [Figure 10](#).

4.2.5 Alignment Pattern

The Alignment Pattern is a set of step-wise alignment lines in the Han Xin Code symbol, as illustrated in [Figure 12](#). There is no Alignment Pattern in the symbols of Version 1, Version 2, and Version 3.

The alignment line is made up of a dark line and an adjacent light line which are one module wide and follow the rule that the dark line is on the upside or right of the light line. On the left borderline and the bottom borderline of the symbol, the alignment line is a dark line in one module width.

There are two alignment lines in the Version 4 to 10 symbols with the length of k modules. When the version is bigger than 10, there are two kinds of Alignment Pattern layouts, the length of the two alignment lines on the lower left corner in the symbol is a special value r modules, the length of the rest regions is k modules except the alignment line on the upper left corner and the alignment line on the lower right corner is a special value $(k-9)$ modules.. The relation of r , k and m in different versions is given by the following formula:

$$n = r + m \times k$$

where

- n : is the total of modules in each side of the symbol;
- r : is the selected length of the two specific alignment lines which locate near by the lower left corner of the Version 4 to 84 Han Xin code in module. r is the length of the dark line(including the turning module);
- k : is the selected length of the normal alignment lines in module, which equals to the length of the light line(not including one of the turning modules)and the length of the dark line(not including one of the turning modules);
- m : is a number obtained by dividing n minus r by k .

The relationship of r and k grids is shown in [Figure 12](#).

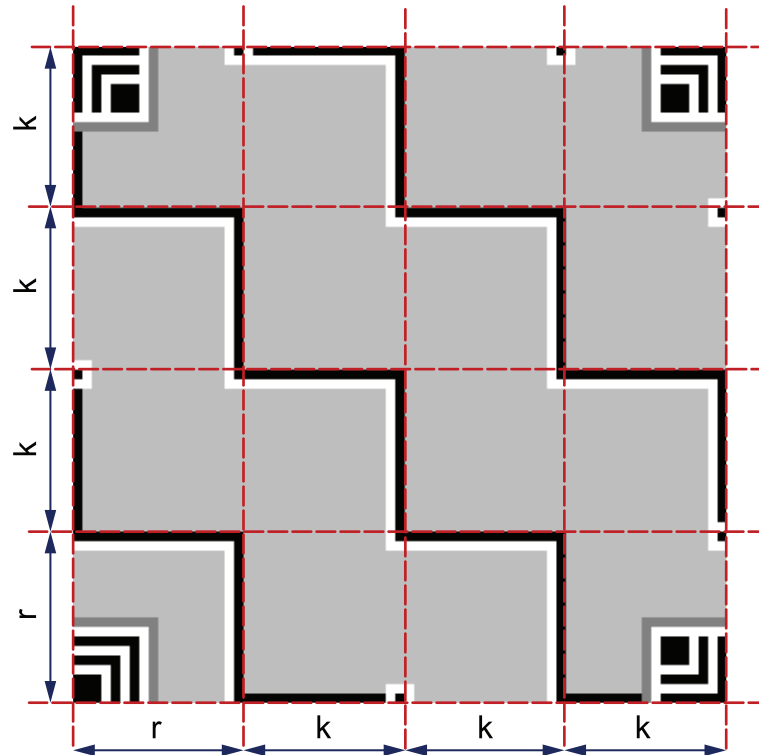
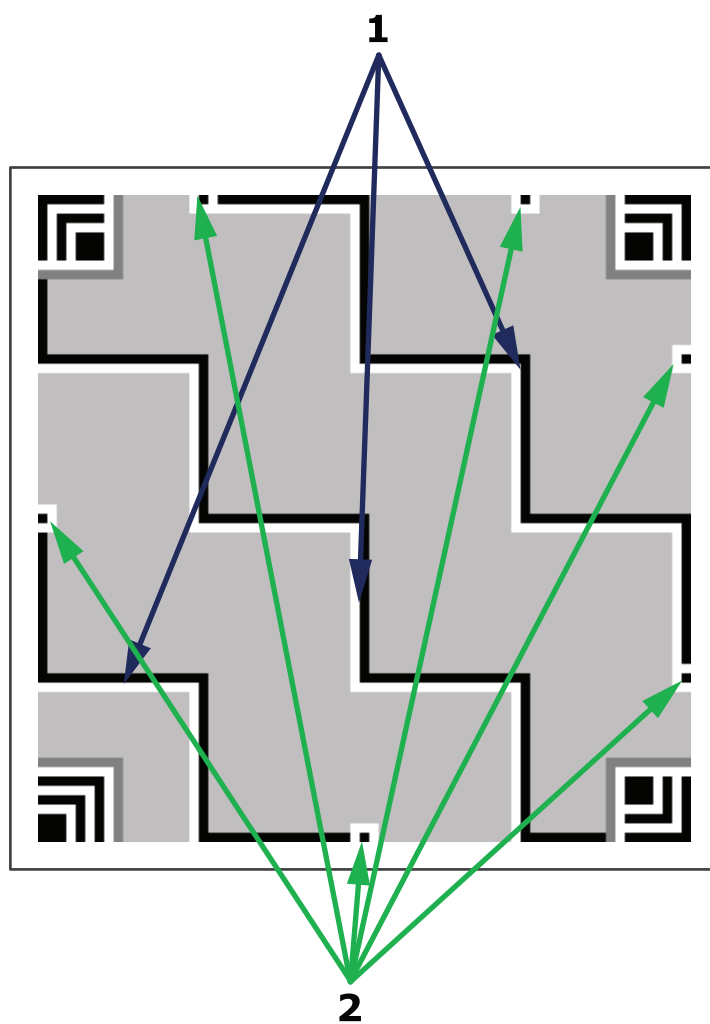


Figure 12 — Alignment Pattern of Version 24 Han Xin Code symbol

The parameters n , k , m , r in different versions of Han Xin Code shall conform to [Table A.1](#) in [Annex A](#).



Key

- 1 Alignment Pattern
- 2 Assistant Alignment Pattern

Figure 13 — Alignment Pattern and Assistant Alignment Pattern of Version 24 Han Xin Code symbol

4.2.6 Assistant Alignment Pattern

Assistant Alignment Pattern is made up of patterns comprised by 6 modules, including 5 light modules and a dark module, on the four sides of the symbol, as illustrated in [Figure 13](#) and [Figure 14](#).

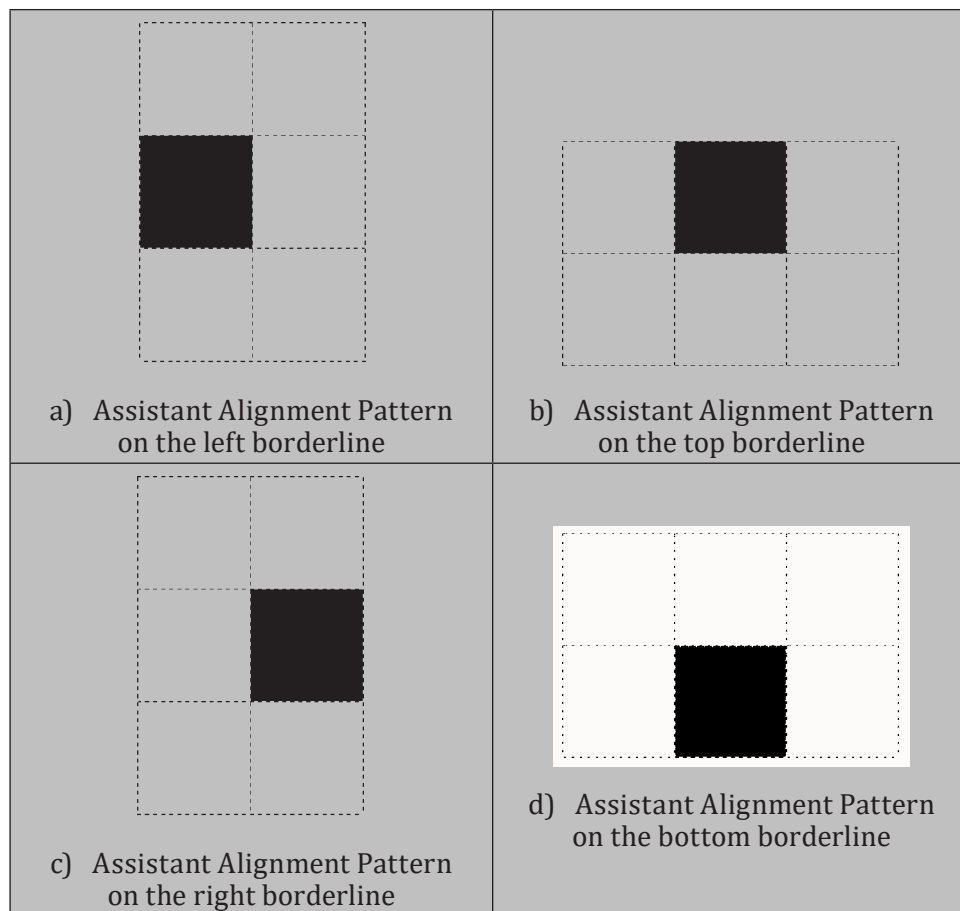
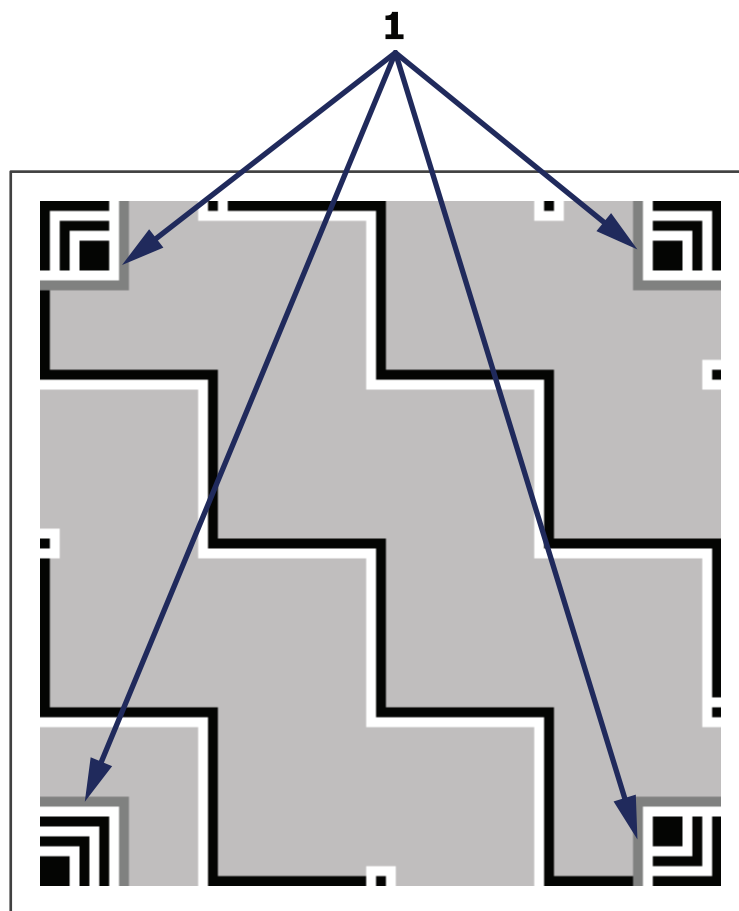


Figure 14 — Assistant Alignment Patterns

Every dark Alignment Pattern segment that points through data to a margin, without hitting another Fixed Pattern, aligns with the dark element of an Assistant Alignment Pattern, as illustrated in [Figure 14](#).

4.2.7 Structural Information Region

Structural Information Region is a one module wide region surrounding the four Position Detection Pattern separator regions, beginning and ending on a margin, as illustrated in [Figure 15](#). Every Structural Information region is made up of 17 modules, totaling 68 modules, which is used to encode Structural Information. See [5.8.3](#) for details.



Key

- 1 Structural Information Region

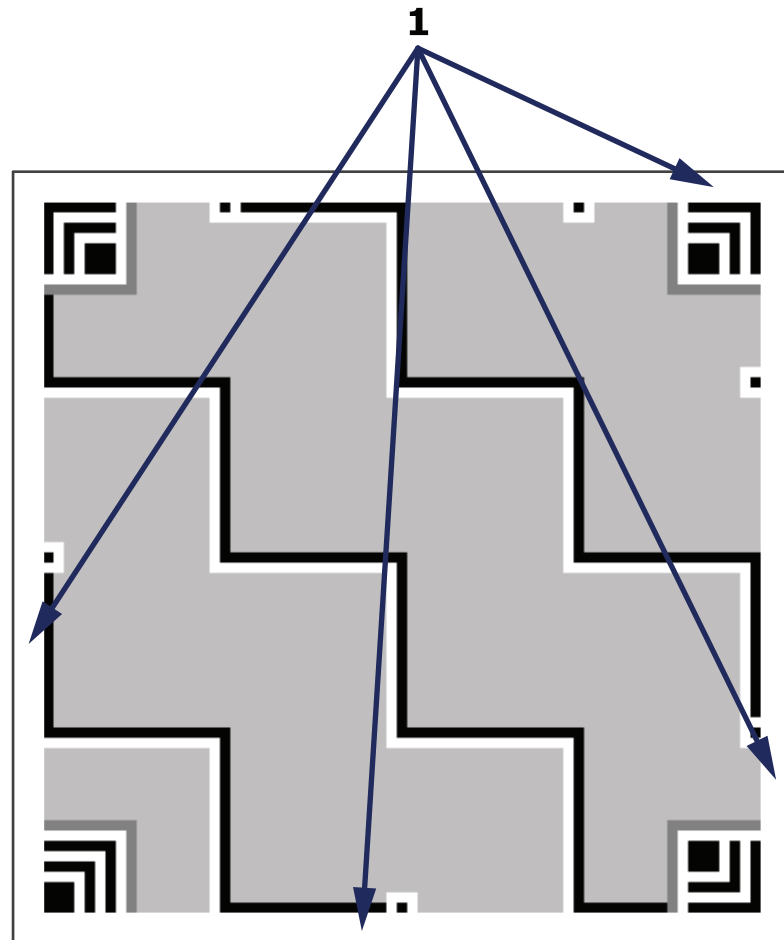
Figure 15 — Function information region of Han Xin Code symbol

4.2.8 Data Region

The data region encodes information codewords, padding bits, error correction codewords and symbol padding bits. It excludes the Fixed Patterns and the Structural Information region in the symbol.

4.2.9 Quiet Zone

This is a region which shall be at least three modules surrounding the symbol on all four sides. Nominal reflectance value of the Quiet Zone shall be equal to that of the light modules, as illustrated in [Figure 16](#).

**Key**

1 Quiet Zone

Figure 16 — Quiet Zone**5 Requirements****5.1 Encode procedure overview**

This subclause provides an overview of the encoding procedure.

The following steps are required to convert user data to a Han Xin Code symbol.

Step 1: Data analysis

The user data is analyzed to identify the variety of different characters to be encoded. Han Xin Code includes several modes (see [5.3](#)) to allow different sub-sets of characters to be converted into symbol characters in efficient ways. Some additional data, such as mode indicators and mode terminators, are generated during mode transforming. These additional data should be fully considered when choosing modes in the process of input data analysis. the versions and data capacities of Han Xin Code symbols shall be identical with [Annex B](#). In order to improve the compaction efficiency, the lowest level encoding scheme (the least bits per character) capable of encoding the data should be selected. When the encoding mode is decided, calculate the length of the data bit stream according to the length of information bit stream and error correction level. If the user does not specify the symbol version, then choose the smallest version that could accommodates the data.

Step 2: Data encoding

Convert the data characters into a bit stream in accordance with the rules for the mode in force, as defined in 5.3 and 5.4. Insert mode indicators as necessary to change modes at the beginning of each new mode segment, and mode terminators at the end of the data sequence. Split the resulting information bit stream into 8-bit codewords. Add pad bits as necessary to fill the number of data codewords required for the version and the error correction level.

Step 3: Error correction encoding

Divide the codewords sequence into the required number of blocks to enable the error correction algorithms to be processed. In each block, generate the error correction codewords, and append them at the end of the data codewords sequence.

Step 4: Constructing final data bit stream

In each error correction block, combine the information codewords sequence and error correction codewords sequence into a data codewords sequence. Construct the final data sequence by connecting all data codewords sequences in order.

Step 5: Module placement in matrix

Place the data modules in the matrix together with the final data bit stream, Finder Pattern, Position Detection Pattern separators, Alignment Pattern (if required), Assistant Alignment Pattern (if required) and symbol padding bits (if required).

Step 6: Data masking

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

Step 7: Structural Information placement in matrix

Generate the Structural Information according to the version, error correction level and masking solution. Place the Structural Information in the matrix and complete the symbol.

5.2 Data analysis

If the user input data does not follow any Extended Channel Interpretation (ECI) protocol or use default ECI of Han Xin Code (\000003), analyze the user data to determine its content. Select the appropriate mode and divide the user data into sub-series of one or more encoding modes to encode each sequence as described in 5.3.

When the input data is GS1 data or URI, use GS1 mode or URI mode described in 5.3 to encode user data.

When the input data is numeric data, ASCII data and/or GB18030 characters etc., use numeric mode, Text mode, Chinese character modes and Binary Byte mode to encode the input data.

If the input data is in UTF8, and the input data cannot be efficiently encoded by other modes, the use Unicode mode to encode this data.

If user has not known or does not specify the encoding/charset/mode of input data, then analysis the input data, determine the lowest level encoding scheme (the least bits per character) capable of encoding the data (as described in 5.1 Step 2 and 5.4.15).

If the input data is known to be encoded differently than the default ECI of Han Xin Code (ECI 3), select the appropriate ECI mode to encode ECI data. The Extended Channel Interpretation (ECI) protocol allows the output data stream to have different interpretations from that of the default character set. The ECI protocol is defined consistently across a number of symbologies. Three broad types of interpretations are supported in Han Xin Code:

- a. international character sets (or code pages)

- b. general purpose interpretations such as encryption and compaction
- c. user defined interpretations for closed systems

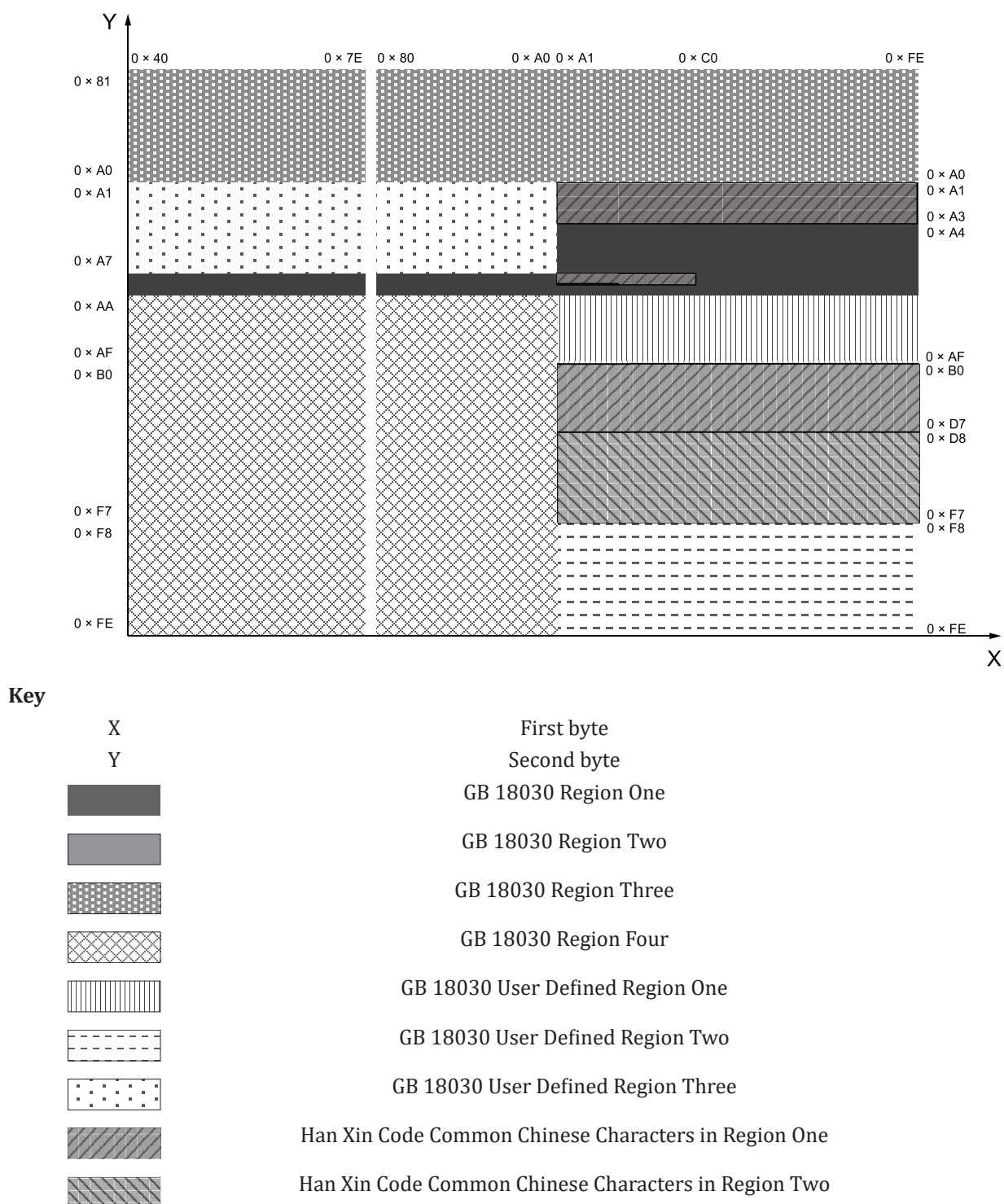
The Extended Channel Interpretation protocol is fully specified in the Extended Channel Interpretation (ECI) document. The protocol provides a consistent method to specify particular interpretations on byte values before printing and after decoding.

A specific Extended Channel Interpretation may be invoked anywhere in the encoded message. The default interpretation for Han Xin Code is ECI 000003 which shall represent the ISO/IEC 646 character set.

5.3 Mode

5.3.1 General

Han Xin Code has mode indicators which are used to define categories of the encoding data. The modes of Han Xin Code defined below are based on the character values and assignments associated with the default ECI. The data encoding modes for Han Xin Code are Numeric, Text mode, Binary Byte, Common Chinese Characters in Region One, Common Chinese Characters in Region Two, GB18030 2-byte region, GB18030 4-byte Region, ECI, URI, GS1, Unicode, etc. The mapping of Common Chinese Characters Mode One and Two are shown in [Figure 17](#).



NOTE In GB18030, all the characters and symbols are designed to be encoded in one-byte, two-byte and four-byte, and the Chinese characters are encoded mainly in two bytes mode, which are separated to Region One and Region Two. The Chinese characters Mode One and Two are not equivalent to the Region One and Region Two defined in GB18030.

Figure 17 — Mapping of the Common Chinese Character Mode One and Two in GB18030

Each mode is indicated by specific mode indicator, as shown in [Table 1](#).

Table 1 — Mode characteristics

Mode	Mode indicators	Bits per character
Reserved	(0000) _{bin}	
Numeric	(0001) _{bin}	3.3 (10 bits for three digits)
Text	(0010) _{bin}	6
Binary Byte	(0011) _{bin}	8
Common Chinese Characters in Region One	(0100) _{bin}	12
Common Chinese Characters in Region Two	(0101) _{bin}	12
GB18030 2-byte Region	(0110) _{bin}	15
GB18030 4-byte Region	(0111) _{bin}	21
ECI	(1000) _{bin}	N/A
Unicode	(1001) _{bin}	Adaptive
Reserved	(1010) _{bin}	
Reserved	(1011) _{bin}	
Reserved	(1100) _{bin}	
Reserved	(1101) _{bin}	
GS1	(1110 0001) _{bin}	N/A
URI	(1110 0010) _{bin}	N/A
Reserved	(1111) _{bin}	

See [Annex C](#) for Han Xin Code symbols of different versions and data capacities of different encoding modes.

5.3.2 Numeric mode

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

((0000000000)_{bin} to (1111100111)_{bin}). The Numeric mode indicator is (0001)_{bin}.

5.3.3 Text mode

Text mode encodes data from common symbols defined in ISO/IEC 646, i.e. byte values 00_{HEX} to 1B_{HEX} and 20_{HEX} to 7F_{HEX}. The Text mode is divided into two sub-modes, i.e. Text1 and Text2. The encoding range of Text1 is capital letters (A-Z), lowercase letters (a-z), and 10 numeric digits (0-9). The encoding range of text2 is the rest of the characters except ASCII 28 to 31 decimal. Every data character in Text mode is represented by 6 bits. The Text mode indicator is (0010)_{bin}.

5.3.4 Binary byte mode

Binary byte mode encodes binary data in any form and encodes them in their binary byte. Every byte in Binary Byte mode is represented by 8 bits. The Binary Byte mode indicator is (0011)_{bin}.

5.3.5 Common Chinese Characters in Region One mode

Every common Chinese Character in Region One is represented by 12 bits. Common Chinese Characters in Region one includes characters whose first byte value is in the range of B0_{HEX} to D7_{HEX} and second byte value is in the range of A1_{HEX} to FE_{HEX} (3760 characters), and characters whose first byte value is in the range of A1_{HEX} to A3_{HEX}, and second byte value is in the range of A1_{HEX} to FE_{HEX} (282 characters), and characters whose byte values are in the range of A8A1_{HEX} to A8C0_{HEX} (32 characters). Common Chinese Characters in Region one are 4074 in total. The Common Chinese Characters in Region One mode indicator is (0100)_{bin}.

5.3.6 Common Chinese Characters in Region Two mode

Every common Chinese Character in Region Two is represented by 12 bits. Common Chinese Characters in Region Two which includes character whose first byte value is in the range of $D8_{\text{HEX}}$ to $F7_{\text{HEX}}$, and second byte value is in the range of $A1_{\text{HEX}}$ to FE_{HEX} (i.e., Chinese characters from 𠄎(chu) to 楚(zha)). Common Chinese Characters in Region Two are 3008 in total. The common Chinese Characters in Region Two mode indicator is $(0101)_{\text{bin}}$.

5.3.7 GB18030 2-byte Region mode

Every GB18030 2-byte Region character is represented by 15 bits. The GB18030 2-byte Region encodes data from all characters (including the common Chinese Characters in Regions One and Two) in GB18030 double-byte region, (i.e., Chinese characters whose first byte value is in the range of 81_{HEX} to FE_{HEX} and second byte value is in the range of 40_{HEX} to $7E_{\text{HEX}}$ or 80_{HEX} to FE_{HEX}). GB18030 2-byte Region has 23940 Chinese characters in total. The GB18030 2-byte Region mode indicator is $(0110)_{\text{bin}}$.

5.3.8 GB18030 4-byte Region mode

Every GB18030 4-byte Region character is represented by 21 bits. GB18030 4-byte Region encodes data from all characters in GB18030 four-byte region (i.e., characters whose first byte value is in the range of 81_{HEX} to FE_{HEX} , and second byte value is in the range of 30_{HEX} to 39_{HEX} , and third byte value is in the range of 81_{HEX} to FE_{HEX} , and fourth byte value is in the range of 30_{HEX} to 39_{HEX}). The GB18030 4-byte Region has 1587600 characters in total. The GB18030 4-byte Region mode indicator is $(0111)_{\text{bin}}$.

5.3.9 Extended Channel Interpretation (ECI) mode

The Extended Channel Interpretation (ECI) mode indicates the encoded data is interpreted according to the ECI protocol defined by the AIM ECI Specifications. The ECI mode indicator is $(1000)_{\text{bin}}$. If all the data is in the default ECI 000003 there is no need to explicitly encode the ECI mode indicator $(1000)_{\text{bin}}$. Mode indicator $(1000)_{\text{bin}}$ is used to introduce ECI mode.

The Extended Channel Interpretation can only be used with readers enabled to transmit symbology identifiers. Readers that are not enabled to transmit the symbology identifier should not transmit the data from any symbol containing an ECI.

Apply one or more modes, such as numeric, Text mode, Common Chinese Character in Region One, etc., to encode its byte in maximum efficiency without considering its actual meaning. For example, a sequence of values in the range of 30_{HEX} to 39_{HEX} would be considered as a sequence of numbers (0 - 9) and encoded in numeric mode (see [5.3.2](#)) even if they are not to be interpreted as numbers.

5.3.10 Unicode mode

Unicode mode designs a way to represent any text data reference to UTF8 encoding/charset in Han Xin Code. It is the intention of Han Xin Code to have Unicode-Mode-only symbols. The mode indicator of Unicode mode is $(1001)_{\text{bin}}$, the data analysis algorithm and encoding process are described in [5.4.12](#).

5.3.11 GS1 mode

GS1 mode indicates the data represented in Han Xin Code is GS1 data of GS1 system defined by GS1 General Specification. If Han Xin Code encodes GS1 data, the GS1 mode shall be entered as the first mode in the symbol. Other modes may be entered for data encodation efficiency but the transmission of the Symbology Identifier does not change, i.e., all the symbol data is GS1 data. GS1 Mode is not allowed to be used for non-GS1 data. If GS1 Mode is not the first mode and is entered somewhere other than the beginning of the symbol, the symbol is invalid. The mode indicator of GS1 mode is $(1110\ 0001)_{\text{bin}}$, the GS1 mode encoding algorithm is described in [5.4.13](#).

5.3.12 URI mode

Uniform Resource Identifier (URI) is widely used in business and daily life of every person. URI mode indicates the data represented in Han Xin Code is Uniform Resource Identifier (URI) reference to RFC 3986. For most web based applications, the data encoded in a 2D barcode is treated as a URI, and there is no need to terminate the URI mode to start a new mode. It is the intention of this mode for a symbol to be used only for URI data. The mode indicator of URI mode is $(1110\ 0010)_{\text{bin}}$, the URI mode encoding algorithm is described in [5.4.14](#).

5.4 Data encoding

5.4.1 General

The encoding process consists of constructing information bit stream and constructing information codewords sequence.

5.4.2 Constructing the information bit stream

Select the mode required to encode any given data according to Numeric mode, Text mode, Common Chinese Characters in Region One mode, etc. (see [5.3.2](#) to [5.3.12](#)).

The bit stream made by mode encoding should include:

- a. Mode indicator (4 bits)
- b. Character count indicator (for Binary Byte mode)
- c. Mode bit stream after encoding
- d. Mode terminator (for Numeric, Text, Common Chinese Characters in Region One, Chinese Characters in Region Two, GB18030 2-byte region, GB18030 4-byte region, Unicode mode, GS1 mode and URI mode).

Each mode segment shall begin with the first bit of the mode indicator and end with the final bit of the mode terminator. Construct the information bit stream by connecting all mode segments in order.

5.4.3 Constructing information codewords sequence

Determine the number of information codewords according to the error correction level and symbol version (see [Annex B](#)). Split the information bit stream into 8-bit codewords, and then add padding 0_{bin} -bits as necessary to fill the number of codewords required for the version and the error correction level.

5.4.4 Numeric mode encoding

In Numeric mode, the input data string is divided into groups of three digits (the number of digits in last group can be less than three), and each group is converted to its 10-bit binary equivalent (i.e. $(0000000000)_{\text{bin}} - (1111100111)_{\text{bin}}$). The binary data is then prefixed with the mode indicator and appended the mode terminator after the binary data sequence.

[Table 2](#) specifies the mapping relationship about number of numeric characters in last group and mode terminators.

Table 2 — Mode terminators of Numeric mode

Number of numeric characters in last group	Mode terminator
1	(111111101) _{bin}
2	(111111110) _{bin}
3	(111111111) _{bin}

EXAMPLE 1

Input data: 84613168549316542

1. Divide into groups of three digits: 846 131 685 493 165 42
2. Choose the Terminator:111111110
3. Convert each group to its binary equivalent:

846→1101001110

131→0010000011

685→1010101101

493→0111101101

165→0010100101

42→0000101010

4. Connect the binary data in sequence:

1101001110 0010000011 1010101101 0111101101 0010100101 0000101010

5. Add the mode indicator and terminator to binary data:

0001 1101001110 0010000011 1010101101 0111101101 0010100101 0000101010 111111110

EXAMPLE 2

Input data: 0019536472255

1. Divide into groups of three digits: 001 953 647 225 5
2. Choose the Terminator:111111101
3. Convert each group to its binary equivalent:

001→0000000001

953→1110111001

647→1010000111

225→0011100001

5→0000000101

4. Connect the binary data in sequence:

0000000001 1110111001 1010000111 0011100001 0000000101

5. Add the mode indicator and terminator to binary data:

0001 0000000001 1110111001 1010000111 0011100001 0000000101 111111101

5.4.5 Text mode encoding

Text mode encodes data from a subset of the characters in ISO/IEC 646 (i.e., byte values $00_{\text{HEX}} - 1B_{\text{HEX}}$ and $20_{\text{HEX}} - 7F_{\text{HEX}}$). Each character is represented by 6 bits. The Text mode is divided into two sub-modes (i.e., Text1 and Text2). The encoding range of Text1 sub-mode is capital letters (A-Z), lowercase letters (a-z), and 10 numeric digits (0-9), as shown in Table 3. The encoding range of Text2 sub-mode is the rest of the symbols, as shown in Table 4. In addition, $(11110)_{\text{bin}}$ is defined as a transition between Text1 sub-mode and Text2 sub-mode. The mode terminator is $(11111)_{\text{bin}}$.

The default encoding sub-mode for Text mode is Text1 sub-mode. Characters to be encoded are read in order. If they are in Text1 sub-mode, they will be encoded according to the encoding values in Table 3; if they are in Text2 sub-mode, $(11110)_{\text{bin}}$ is first used to transfer mode from Text1 sub-mode to Text2 sub-mode, then the characters are encoded according to the encoding values in Table 4 until another sub-mode character is encountered or the character sequence is ended. $(11110)_{\text{bin}}$ is used to change back to Text1 sub-mode. The Text mode terminator $(11111)_{\text{bin}}$ is added followed by the generated encoding binary bit stream to end the mode's encoding.

Table 3 — Encoding/decoding mapping table of Text1 sub-mode

Character	ASCII value	Encoding value	Character	ASCII value	Encoding value
0	48	$(000000)_{\text{bin}}$	V	86	$(011111)_{\text{bin}}$
1	49	$(000001)_{\text{bin}}$	W	87	$(100000)_{\text{bin}}$
2	50	$(000010)_{\text{bin}}$	X	88	$(100001)_{\text{bin}}$
3	51	$(000011)_{\text{bin}}$	Y	89	$(100010)_{\text{bin}}$
4	52	$(000100)_{\text{bin}}$	Z	90	$(100011)_{\text{bin}}$
5	53	$(000101)_{\text{bin}}$	a	97	$(100100)_{\text{bin}}$
6	54	$(000110)_{\text{bin}}$	b	98	$(100101)_{\text{bin}}$
7	55	$(000111)_{\text{bin}}$	c	99	$(100110)_{\text{bin}}$
8	56	$(001000)_{\text{bin}}$	d	100	$(100111)_{\text{bin}}$
9	57	$(001001)_{\text{bin}}$	e	101	$(101000)_{\text{bin}}$
A	65	$(001010)_{\text{bin}}$	f	102	$(101001)_{\text{bin}}$
B	66	$(001011)_{\text{bin}}$	g	103	$(101010)_{\text{bin}}$
C	67	$(001100)_{\text{bin}}$	h	104	$(101011)_{\text{bin}}$
D	68	$(001101)_{\text{bin}}$	i	105	$(101100)_{\text{bin}}$
E	69	$(001110)_{\text{bin}}$	j	106	$(101101)_{\text{bin}}$
F	70	$(001111)_{\text{bin}}$	k	107	$(101110)_{\text{bin}}$
G	71	$(010000)_{\text{bin}}$	l	108	$(101111)_{\text{bin}}$
H	72	$(010001)_{\text{bin}}$	m	109	$(110000)_{\text{bin}}$
I	73	$(010010)_{\text{bin}}$	n	110	$(110001)_{\text{bin}}$
J	74	$(010011)_{\text{bin}}$	o	111	$(110010)_{\text{bin}}$
K	75	$(010100)_{\text{bin}}$	p	112	$(110011)_{\text{bin}}$
L	76	$(010101)_{\text{bin}}$	q	113	$(110100)_{\text{bin}}$
M	77	$(010110)_{\text{bin}}$	r	114	$(110101)_{\text{bin}}$
N	78	$(010111)_{\text{bin}}$	s	115	$(110110)_{\text{bin}}$
O	79	$(011000)_{\text{bin}}$	t	116	$(110111)_{\text{bin}}$
P	80	$(011001)_{\text{bin}}$	u	117	$(111000)_{\text{bin}}$
Q	81	$(011010)_{\text{bin}}$	v	118	$(111001)_{\text{bin}}$
R	82	$(011011)_{\text{bin}}$	w	119	$(111010)_{\text{bin}}$
S	83	$(011100)_{\text{bin}}$	x	120	$(111011)_{\text{bin}}$
T	84	$(011101)_{\text{bin}}$	y	121	$(111100)_{\text{bin}}$

Table 3 (continued)

Character	ASCII value	Encoding value	Character	ASCII value	Encoding value
U	85	(011110) _{bin}	z	122	(111101) _{bin}

Table 4 — Encoding/decoding mapping table of Text2 sub-mode

Character	ASCII value	Encoding value	Character	ASCII value	Encoding value
NUL	0	(000000) _{bin}	#	35	(011111) _{bin}
SOH	1	(000001) _{bin}	\$	36	(100000) _{bin}
STX	2	(000010) _{bin}	%	37	(100001) _{bin}
ETX	3	(000011) _{bin}	&	38	(100010) _{bin}
EOT	4	(000100) _{bin}	'	39	(100011) _{bin}
ENQ	5	(000101) _{bin}	(	40	(100100) _{bin}
ACK	6	(000110) _{bin}	)	41	(100101) _{bin}
BEL	7	(000111) _{bin}	*	42	(100110) _{bin}
BS	8	(001000) _{bin}	+	43	(100111) _{bin}
HT	9	(001001) _{bin}	,	44	(101000) _{bin}
LF	10	(001010) _{bin}	-	45	(101001) _{bin}
VT	11	(001011) _{bin}	.	46	(101010) _{bin}
FF	12	(001100) _{bin}	/	47	(101011) _{bin}
CR	13	(001101) _{bin}	:	58	(101100) _{bin}
SO	14	(001110) _{bin}	;	59	(101101) _{bin}
SI	15	(001111) _{bin}	<	60	(101110) _{bin}
DLE	16	(010000) _{bin}	=	61	(101111) _{bin}
DC1	17	(010001) _{bin}	>	62	(110000) _{bin}
DC2	18	(010010) _{bin}	?	63	(110001) _{bin}
DC3	19	(010011) _{bin}	@	64	(110010) _{bin}
DC4	20	(010100) _{bin}	[	91	(110011) _{bin}
NAK	21	(010101) _{bin}	\	92	(110100) _{bin}
SYN	22	(010110) _{bin}	]	93	(110101) _{bin}
ETB	23	(010111) _{bin}	^	94	(110110) _{bin}
CAN	24	(011000) _{bin}	_	95	(110111) _{bin}
EM	25	(011001) _{bin}	`	96	(111000) _{bin}
SUB	26	(011010) _{bin}	{	123	(111001) _{bin}
ESC	27	(011011) _{bin}		124	(111010) _{bin}
SP	32	(011100) _{bin}	}	125	(111011) _{bin}
!	33	(011101) _{bin}	~	126	(111100) _{bin}
"	34	(011110) _{bin}	DEL	127	(111101) _{bin}

5.4.6 Binary mode encoding

Binary Byte mode is used to encode binary data in any form. The linking of Binary Byte mode indicator, count indicator and binary byte sequence form an information bit stream of binary byte encoding mode. The count indicator is a 13-bit binary sequence and is represented as the byte number of input binary data.

5.4.7 Common Chinese Characters in Region One mode encoding

Common Chinese Characters in Region One mode encodes common characters and symbols in double-byte region one and two of GB18030, 4074 characters in total. Each character or symbol is represented by 12-bit binary sequence according to the following method. The encoding value of the character in this mode is in the range of $(000000000000)_{\text{bin}}$ to $(11111101001)_{\text{bin}}$.

- a. For characters and symbols with GB18030 values whose first byte value is in the range of $B0_{\text{HEX}}$ to $D7_{\text{HEX}}$ and second byte value is in the range of $A1_{\text{HEX}}$ to FE_{HEX} :
 - (1) Subtract $A1_{\text{HEX}}$ from the second byte value.
 - (2) Subtract $B0_{\text{HEX}}$ from the first byte value.
 - (3) Multiply the result of (2) by $5E_{\text{HEX}}$.
 - (4) Add the result of (2) to the product from (3).
 - (5) Convert the result to a 12-bit binary string.
- b. For characters and symbols with GB18030 values whose first byte value is in the range of $A1_{\text{HEX}}$ to $A3_{\text{HEX}}$ and second byte value is in the range of $A1_{\text{HEX}}$ to FE_{HEX} :
 - (1) Subtract $A1_{\text{HEX}}$ from the second byte value.
 - (2) Subtract $A1_{\text{HEX}}$ from the first byte value.
 - (3) Multiply the result of (2) by $5E_{\text{HEX}}$.
 - (4) Add the result of (1) and $EB0_{\text{HEX}}$ to the product from (3).
 - (5) Convert the result to a 12-bit binary string.
- c. For characters and symbols with GB18030 values from $A8A1_{\text{HEX}}$ to $A8C0_{\text{HEX}}$, subtract $A1_{\text{HEX}}$ from the second byte value then add FCA_{HEX} to the result, then convert the result into a 12-bit binary string.

The information bit stream is generated in the following manner. Begin with mode indicator $(0100)_{\text{bin}}$. Encode characters in order into 12-bit binary and connect them into a sequence. Add the mode terminator $(11111111111)_{\text{bin}}$. Additionally, $(11111111110)_{\text{bin}}$ is considered as transition from Common Character In Region One to Common Character In Region Two.

EXAMPLE 1

Input character :



Character value:	$C8AB_{\text{HEX}}$
(1) Subtract $B0_{\text{HEX}}$ from its first byte value	$C8_{\text{HEX}} - B0_{\text{HEX}} = 18_{\text{HEX}}$
(2) Multiply the result of (1) by $5E_{\text{HEX}}$	$18_{\text{HEX}} \times 5E_{\text{HEX}} = 8D0_{\text{HEX}}$
(3) Subtract $A1_{\text{HEX}}$ from its second byte value	$AB_{\text{HEX}} - A1_{\text{HEX}} = 0A_{\text{HEX}}$
(4) Add the result of (2) to result of (3)	$8D0_{\text{HEX}} + 0A_{\text{HEX}} = 8DA_{\text{HEX}}$
(5) Convert the result to a 12-bit binary string	$(100011011010)_{\text{bin}}$

EXAMPLE 2

Input character:

•
;

Character value:

A3BB_{HEX}

- (1) Subtract A1_{HEX} from its first byte value $A3_{HEX} - A1_{HEX} = 02_{HEX}$
- (2) Multiply the result of (1) by 5E_{HEX} $02_{HEX} \times 5E_{HEX} = BC_{HEX}$
- (3) Subtract A1_{HEX} from its second byte $BB_{HEX} - A1_{HEX} = 1A_{HEX}$
- (4) Add the result of (2), result of (3) and EB0_{HEX} $BC_{HEX} + 1A_{HEX} + EB0_{HEX} = F86_{HEX}$
- (5) Convert the result to a 12-bit binary string $(1111\ 1000\ 0110)_{bin}$

EXAMPLE 3

Input character:

ň

Character value:

A8BE_{HEX}

- (1) Subtract A1_{HEX} from its second byte value $BE_{HEX} - A1_{HEX} = 1D_{HEX}$
- (2) Add the result of (1) and FCA_{HEX} $1D_{HEX} + FCA_{HEX} = FE7_{HEX}$
- (3) Convert the result to a 12-bit binary string $(111111100111)_{bin}$

5.4.8 Common Chinese Characters in Region Two mode encoding

Common Chinese Characters in Region Two mode encodes common characters and symbols in double-byte region two of GB18030, 3008 characters in total. Each character or symbol is represented by 12-bit binary sequence according to following method. The encoding value of the character in this mode is in the range of $(000000000000)_{bin}$ to $(101110111111)_{bin}$.

For characters and symbols with GB18030 values whose first byte value is in the range of D8_{HEX} to F7_{HEX} and second byte value is in the range of A1_{HEX} to FE_{HEX}:

- (1) Subtract A1_{HEX} from the second byte value.
- (2) Subtract D8_{HEX} from the first byte value.
- (3) Multiply the result of (2) by 5E_{HEX}.
- (4) Add the result of (1) to the product from (3).
- (5) Convert the result to a 12-bit binary string.

The information bit stream is generated in the following manner. Begin with mode indicator (0101)_{bin}. Encode characters in order into 12-bit binary and connect them into a sequence. Add the mode terminator $(111111111111)_{bin}$. Additionally, $(111111111110)_{bin}$ is considered as transition from Common Character in Region Two to Common Character in Region One.

EXAMPLE

Input character :



Character value:	$F3A3_{\text{HEX}}$
(1) Subtract $D8_{\text{HEX}}$ from its first byte value	$F3_{\text{HEX}} - D8_{\text{HEX}} = 1B_{\text{HEX}}$
(2) Multiply the result of (1) by $5E_{\text{HEX}}$	$1B_{\text{HEX}} \times 5E_{\text{HEX}} = 9EA_{\text{HEX}}$
(3) Subtract $A1_{\text{HEX}}$ from its second byte value	$A3_{\text{HEX}} - A1_{\text{HEX}} = 02_{\text{HEX}}$
(4) Add the result of (3) to the product from (2)	$9EA_{\text{HEX}} + 02_{\text{HEX}} = 9EC_{\text{HEX}}$
(5) Convert the result to a 12-bit binary string	$(100111101100)_{\text{bin}}$

5.4.9 GB18030 2-byte Region mode encoding

GB18030 2-byte Region mode encodes all characters and symbols in double-byte region of GB18030, i.e. Chinese characters whose first byte value is in the range of 81_{HEX} to FE_{HEX} and second byte value is in the range of 40_{HEX} to $7E_{\text{HEX}}$ or in the range of 80_{HEX} to FE_{HEX} , 23940 Chinese characters in total. Characters and symbols in this mode are represented by 15-bit binary sequence according to following method. The encoding value of the character in this mode is in the range of $(0000000000000000)_{\text{bin}}$ to $(1011101100000011)_{\text{bin}}$.

The following steps are the complete 2-bit Byte Chinese Characters Mode Encoding process:

- (1) For characters whose second byte value is in the range of 40_{HEX} to $7E_{\text{HEX}}$, subtract 40_{HEX} from its second byte value. For characters whose second byte value is in the range of 80_{HEX} to FE_{HEX} , subtract 41_{HEX} from its second byte value.
- (2) Subtract 81_{HEX} from the first byte value.
- (3) Multiply the result of (2) by BE_{HEX} .
- (4) Add the result of (1) to the product from (3).
- (5) Convert the result to a 15-bit binary string.

The information bit stream is generated in the following manner. Begin with mode indicator $(0110)_{\text{bin}}$. Encode characters in order into 15-bit binary and connect them into a sequence. Add the mode terminator $(111111111111111)_{\text{bin}}$.

EXAMPLE 1

Input character :



Character value:	$9D51_{\text{HEX}}$
(1) Subtract 81_{HEX} from its first byte value	$9D_{\text{HEX}} - 81_{\text{HEX}} = 1C_{\text{HEX}}$
(2) Multiply the result of (1) by BE_{HEX}	$1C_{\text{HEX}} \times BE_{\text{HEX}} = 14C8_{\text{HEX}}$
(3) Subtract 40_{HEX} from its second byte value	$51_{\text{HEX}} - 40_{\text{HEX}} = 11_{\text{HEX}}$
(4) Add the result of (3) to the product from (2)	$14C8_{\text{HEX}} + 11_{\text{HEX}} = 14D9_{\text{HEX}}$
(5) Convert the result to a 15-bit binary sequence	$(001010011011001)_{\text{bin}}$

EXAMPLE 2

Input character :

瀾

Character value:

9EAF_{HEX}

- (1) Subtract 81_{HEX} from its first byte value $9E_{HEX} - 81_{HEX} = 1D_{HEX}$
- (2) Multiply the result of (1) by BE_{HEX} $1D_{HEX} \times BE_{HEX} = 1586_{HEX}$
- (3) Subtract 41_{HEX} from its second byte $AF_{HEX} - 41_{HEX} = 6E_{HEX}$
- (4) Add the result of (3) to the product from (2) $1586_{HEX} + 6E_{HEX} = 15F4_{HEX}$
- (5) Convert the result to a 15-bit binary sequence (001010111110100)_{bin}

5.4.10 GB18030 4-byte Region mode encoding

GB18030 4-byte Region mode encodes all characters and symbols in four-byte region of GB18030, i.e. Chinese characters and symbols whose first byte value is in the range of 81_{HEX} to FE_{HEX}, second byte value is in the range of 30_{HEX} to 39_{HEX}, third byte value is in the range of 81_{HEX} to FE_{HEX} and fourth byte value is in the range of 30_{HEX} to 39_{HEX}, 1,587,600 Chinese characters and symbols in total. This mode does not have a mode terminator. By default, this encoding mode will be ended automatically after encoding a character. Each of the characters and symbols are represented by a 21-bit binary sequence with a leading four-bit mode indicator (0111)_{bin} according to the following method.

The encoding value of the character in this mode is in the range of (00000000000000000000)_{bin} to (11000001110011001111)_{bin}. A four-bit mode indicator is added in front of the sequence.

The following steps are encoding method:

- (1) Subtract 30_{HEX} from the fourth byte value.
- (2) Subtract 81_{HEX} from the third byte value.
- (3) Subtract 30_{HEX} from the second byte value.
- (4) Subtract 81_{HEX} from the first byte value.
- (5) Multiply the result of (2) by 0A_{HEX}.
- (6) Multiply the result of (3) by 04EC_{HEX}.
- (7) Multiply the result of (4) by 3138_{HEX}.
- (8) Add the result of (1), the product from (5), the product from (6) and the product from (7) together.
- (9) convert the result to a 21-bit binary sequence.

EXAMPLE

Input character :

丙

Character value:

8139EF30_{HEX}

- (1) Subtract 81_{HEX} from its first byte value $81_{HEX} - 81_{HEX} = 00_{HEX}$
- (2) Multiply the result of (1) by 3138_{HEX} $00_{HEX} \times 3138_{HEX} = 00_{HEX}$
- (3) Subtract 30_{HEX} from its second byte value $39_{HEX} - 30_{HEX} = 09_{HEX}$

- (4) Multiply the result of (3) by $04E_{\text{C}_{\text{HEX}}}$ $09_{\text{HEX}} \times 04E_{\text{C}_{\text{HEX}}} = 2C4C_{\text{HEX}}$
- (5) Subtract 81_{HEX} from its third byte value $EF_{\text{HEX}} - 81_{\text{HEX}} = 6E_{\text{HEX}}$
- (6) Multiply the result of (5) by $0A_{\text{HEX}}$ $6E_{\text{HEX}} \times 0A_{\text{HEX}} = 44C_{\text{HEX}}$
- (7) Subtract 30_{HEX} from the fourth byte value $30_{\text{HEX}} - 30_{\text{HEX}} = 00_{\text{HEX}}$
- (8) Add the results of (2), (4), (6) and (7) together
- $$00_{\text{HEX}} + 2C4C_{\text{HEX}} + 44C_{\text{HEX}} + 00_{\text{HEX}} = 3098_{\text{HEX}}$$
- (9) Convert the result to a 21-bit binary sequence $(000000011000010011000)_{\text{bin}}$

5.4.11 ECI mode encoding

5.4.11.1 General

ECI mode is invoked by the use of mode indicator $(1000)_{\text{bin}}$. The data sequence is encoded according to the rules of Numeric mode, Text mode, etc. See encoding rules described in 5.4. Each mode segment begins with the highest bit of mode indicator and ends with the lowest bit of data bit stream (for Binary Byte mode) or mode terminator (for Numeric mode, Text mode, Common Chinese Characters in Region One mode, Common Chinese Characters in Region Two mode, GB18030 2-byte Region mode, GB18030 4-byte Region mode). The length of each mode segment is determined definitely by the encoding mode rules; there is no separator between segments. The ECI encoding sequence is generated by connecting the ECI designator, ECI assignment numbers and one or more binary sequences encoded by related ECI mode.

5.4.11.2 ECI designator

The ECI designator appears in the data to be encoded as character $5C_{\text{HEX}}$ (“\” or the backslash in ISO/IEC 646) followed by the six digit assignment number. Where $5C_{\text{HEX}}$ appears as true data it shall be doubled in the data string before encoding in symbols to which the ECI protocol applies. Each ECI is designated by a six-digit assignment number which is encoded in the Han Xin Code symbol as the one, two or three bytes following the ECI mode indicator. The encoding rules are defined in Table 5. The lower numbered ECI assignments may be encoded in multiple ways, but the shortest way is preferred.

Table 5 — Encoding ECI assignment numbers

ECI assignment value	Number of encoding bit	Codeword values
000000 to 000127	8	0bbbbbbb
000000 to 016383	16	10bbbbbb bbbbbbbb
000000 to 811799	24	110bbbbbb bbbbbbbb bbbbbbbb
NOTE b.....b is the binary value of ECI assignment number.		

EXAMPLE

Assume data to be encoded are Greek letters “ΑΒΓΔΕ”, using character set ISO/IEC 8859-7 (ECI000009).

Data to be encoded: \000009ΑΒΓΔΕ (character values are $A1_{\text{HEX}}$ $A2_{\text{HEX}}$ $A3_{\text{HEX}}$ $A4_{\text{HEX}}$ $A5_{\text{HEX}}$)

Bit sequence in symbol:

ECI mode indicator:	1000
ECI Assignment number (000009):	00001001
Mode indicator (byte):	0011
Character count indicator:	0000000000101
Data:	10100001 10100010 10100011 10100100 10100101
Final information bit string:	1000 00001001 0011 0000000000101 10100001 10100010 10100011 10100100 10100101

5.4.11.3 Multi-ECI

Refer to the AIM ECI specification for the rules defining the effect of subsequent ECI designator in an ECI data segment. (ECI assignment numbers are 000000 to 000898.) There can be multiple ECI modes to encode data. For example, data to which a character set ECI has been applied may be subject to encryption or compaction using a transformation ECI which will co-exist with the initial ECI (ECI assignment number more than 000898), or a second character set ECI will have the effect of terminating the first ECI and starting a new ECI segment. Where any ECI designator appears in the data, it shall be encoded in the Han Xin Code symbol in accordance with [5.4.11.2](#) and shall commence a new mode segment.

5.4.12 Unicode mode

5.4.12.1 General

Unicode mode represents the UTF 8 message in Han Xin Code.

For the input data, it is composed of characters which could be represented by 1 byte, 2 bytes, 3 bytes or 4 bytes. In the Unicode mode, the input data is analysed by using the following self-adaptive data analysis algorithm. Firstly, divide and combine the input data into the 1, 2, 3, or 4 byte pattern pre-encoding sub-sequences, and secondly apply a run-length data compression algorithm to encode each sub-sequences of the input data. The self-adaptive analysis and run-length encoding process is described in [5.4.12.3](#). The source code of Unicode mode encoding shall be as listed in the [Annex N](#).

5.4.12.2 Mode indicator and terminator

The mode indicator of Unicode mode is $(1001)_{\text{bin}}$ and the mode terminator of Unicode mode is $(1111)_{\text{bin}}$.

5.4.12.3 Data analysis and encoding algorithm

- a) The Mode Indicator of Han Xin Code Unicode mode is $(1001)_{\text{bin}}$.
- b) Use following steps to analysis encoded data:
 - 1) Initially analysis the input data in accordance with the encoding algorithm described in c), and get the initial byte pattern analysis result:
 - I. Read the first 12 bytes of data to analysis:
 - i) If the whole data length is less than 12 bytes, go to iii); otherwise, go to next step.
 - ii) Use 1 byte pattern, 2 bytes pattern, 3 bytes pattern and 4 byte pattern to encode the first 12 bytes data, choose the byte pattern which has the lowest pre-encoding bit rate for single byte (encoding bits / bytes counter) as the initial byte pattern of first 12 bytes. If this byte pattern is 4 bytes pattern, then go to viii); otherwise, go to next step.

- iii) Read the first 9 bytes of data, if the whole data length is less than 9 bytes, go to v); otherwise, go to next step.
 - iv) Use 1 byte pattern, 2 bytes pattern (only analysis 8 bytes) and 3 bytes pattern to pre-encode the first 9 bytes data, choose the byte pattern which has the lowest encoding bit rate for single byte as the initial byte pattern of first 9 bytes. If this byte pattern is 3 bytes pattern, then go to viii); otherwise, go to next step.
 - v) Read the first 6 bytes of data, if the whole data length is less than 6 bytes, go to vii); otherwise, go to next step.
 - vi) Use 1 byte pattern and 2 bytes pattern to pre-encode the first 6 bytes data, choose the byte pattern which has the lowest encoding bit rate for single byte as the initial byte pattern of first 6 bytes. If this byte pattern is 2 bytes pattern, then go to viii); otherwise, go to next step.
 - vii) Use 1 byte pattern as the initial byte pattern of first byte.
 - viii) Utilise the initial byte pattern in the initial data byte series, set the next analysis position of data to the next byte of the initial byte series.
- II. For next analysis position of data, if reach the end of data, the initially analysis is over; otherwise, read the next 12 bytes of data, to analysis:
- i) If the whole data length from next analysis position is less than 12 bytes, go to iii); otherwise, go to next step.
 - ii) Use 1 byte pattern, 2 bytes pattern, 3 bytes pattern and 4 byte pattern to pre-encode the next 12 bytes data, choose the byte pattern which has the lowest encoding bit rate for single byte (encoding bits / bytes counter) as the initial byte pattern of next 12 bytes. If this byte pattern is 4 bytes pattern, then go to viii); otherwise, go to next step.
 - iii) Read the next 9 bytes of data, if the whole data length from next analysis position is less than 9 bytes, go to v); otherwise, go to next step.
 - iv) Use 1 byte pattern, 2 bytes pattern (only analysis 8 bytes) and 3 bytes pattern to encode the next 9 bytes data, choose the byte pattern which has the lowest encoding bit rate for single byte as the initial byte pattern of the 9 bytes. If this byte pattern is 3 bytes pattern, then go to viii); otherwise, go to next step.
 - v) Read the next 6 bytes of data, if the whole data length from next analysis position is less than 6 bytes, go to vii); otherwise, go to next step.
 - vi) Use 1 byte pattern and 2 bytes pattern to encode the next 6 bytes data, choose the byte pattern which has the lowest encoding bit rate for single byte as the initial byte pattern of next 6 bytes. If this byte pattern is 2 bytes pattern, then go to viii); otherwise, go to next step.
 - vii) Use 1 byte pattern as the initial byte pattern of the next byte.
 - viii) If the initial byte pattern of the data series is different than the previous initial byte pattern of the previous data byte series, set the initial byte pattern as data pattern of the data series, move the next analysis position of data by adding the length of the byte series, go back to II; otherwise, if the initial byte pattern of the data byte series is same as the previous initial byte pattern of the previous data, analyze and decide whether to combine the two data byte series into one or not. The data combination analysis is to calculate and compare the encoding bits for using two separate byte patterns and the encoding bits for integrating to one single byte series:
 - If the encoding bits for integrating to one single byte series is not greater than the encoding bits using two separate byte patterns, then do not change the previous byte

pattern to encode, but notice to change the length, difference and minimum values of each byte for the byte pattern;

- If the encoding bits for integrating to one single byte pattern is greater than the encoding bits for using two separate byte patterns, then start a new byte pattern to encode;
- For both of the above two situation, move the next analysis position by adding the length of the new analyzed byte pattern.

2) Optimize the initial byte pattern analysis result by following steps:

I. If the data sequence and the previous data sequence use the same byte pattern:

Calculate the encoding bits of the two adjacent data sequence with two separate byte patterns (different length, difference and minimum values)and the encoding bits of the two data sequence with single byte pattern:

- If the encoding bits for integrating to one single byte pattern is greater than the encoding bits for using two separate byte patterns, then do not need to change the byte pattern;
- If the encoding bits for integrating to one single byte pattern is not greater than the encoding bits for using two separate byte patterns, then integrate the two separate byte patterns to one single byte pattern, but notice to change the length, difference and minimum values of each byte for the byte pattern.

II. If total byte length of two byte patterns is less than 16:

Calculate the encoding bits of the two adjacent data sequence with two separate byte patterns (different length, difference and minimum values)and the encoding bits of the two data sequence with one local single byte pattern(1 byte mode):

- If the encoding bits for integrating to one local single byte pattern is not less than the encoding bits for using two separate byte patterns, then do not need to change the byte pattern;
- If the encoding bits for integrating to one local single byte pattern is less than the encoding bits for using two separate byte patterns, then integrate the two separate byte patterns to one local single byte pattern

III. Repeat above steps described in this step until there is no more optimization can be made.

c) Utilise the analysis result by Step 2, use following steps to encode the input data:

- 1) For each data sequence of 1, 2, 3, or 4 byte pattern, choose the relevant byte pattern indicator as the start bits of the encoding (see [Table 6](#)).

Table 6 — Byte pattern indicators in Unicode mode

Byte pattern	Byte pattern indicator
1 byte pattern	(0001) _{bin}
2 bytes pattern	(0010) _{bin}
3 bytes pattern	(0011) _{bin}
4 bytes pattern	(0100) _{bin}
The terminator of Unicode mode	(1111) _{bin}

- 2) Utilise the encoding format in [Table 7](#) to encode the byte pattern counter of data sequence of this byte pattern (the byte pattern counter is based on byte pattern, not the byte length, for example 3 bytes pattern with counter 2 has 6 bytes):

Table 7 — Encoding format of the byte pattern counter in Unicode mode

Counter range	Encoding bits	Encoding format
0 to 7	4	0XXX
8 to 63	8	10XX XXXX
64 to 511	12	110X XXXX XXXX
512 to 4095	16	1110 XXXX XXXX XXXX
4096 to 32767	20	1111 0XXX XXXX XXXX XXXX

- 3) Find the minimum of each 1, 2, 3, or 4 bytes groups, calculate and recode all the differences of each byte positions of the 1, 2, 3, or 4 byte groups compared to the minimum. Add 1, 2, 3, or 4 4-bit difference length identifier to the encoding bit stream to encode how many bits will be enough to encode the differences of each byte position compared to the minimum values for each byte positions of each 1, 2, 3 or 4 bytes patterns.

For example:

4 bytes pattern, the each byte group needs 2 bits, 3 bits, 6 bits and 8 bits to encode the difference in order, the encoding here is:

0010 0011 0110 1000

- 4) Add 1, 2, 3 or 4 8-bit encoding to encode the minimum values for each byte groups. if all data of one byte position is all same (0 bit differences), then this minimum value for this byte position is the value for all data of this byte position.
- 5) Add differences encoding bits calculated by c)3) to encode difference value of each 1, 2, 3, or 4 byte group of for each byte position from beginning of this byte pattern to the end. If all data of one byte position is all same (0 bit differences), then do not need to encode the difference of this byte position.
- d) Add the mode terminator of Unicode mode of Han Xin Code - $(1111)_{bin}$ at the end of the encoding of input data.

5.4.13 GS1 mode encoding

5.4.13.1 General

GS1 mode encode GS1 data of GS1 system represented by following structure:

AI_1Data_1 $AI_iData_i<FNC1>$ AI_nData_n

AI means Application Identifier which shall conform to the GS1 General Specification.

Data means the Data of specific Application Identifier AI defined in GS1 General Specification.

5.4.13.2 Data representation rule

When representing the above GS1 data in Han Xin Code, split the GS1 data into several GS1 data segments:

AI_1Data_1 $AI_iData_i<FNC1>$ AI_nData_n
 GS1 data segment 1 GS1 data segment i GS1 data segment n

where

$Segment_i = AI_iData_i, i = 1, 2, \dots, m$

And each GS1 data segment has a AI_i key, and $Data_i$ is the value associated with the key AI_i , with the $Data_i$ value of this AI_i key. According to the GS1 specifications, all predefined length GS1 data strings should be preceded, followed by all non-predefined length GS1 data strings. If more than two non-predefined length strings are used at the end of the whole string, then <FNC1> should be added in between. The separator character <FNC1> is placed immediately after a non-predefined length data segment and is followed by the GS1 Application Identifier of the next data segment. However, if a non-predefined length data segment is the last data segment to be encoded, the <FNC1> should not be used.

5.4.13.3 Mode indicator and terminator

Mode indicator of GS1 mode is $(1110\ 0001)_{bin}$, and mode terminator of GS1 mode is $(1111\ 1111)_{bin}$.

5.4.13.4 Encoding algorithm

- a) Use $(1110\ 0001)_{bin}$ to indicate the start of GS1 mode, goto b), etc.
- b) Combine the GS1 data segments into data segments groups separated by <FNC1>.
- c) Apply the standard Han Xin Code encoding algorithm to the first segments group or the whole message if there is no need to add FNC1.
- d) If there are two or more groups, select the first two adjacent GS1 data segment groups separate by <FNC1>. If the leading encoding scheme before the <FNC1> are numeric mode, treat the <FNC1> as a numeric extension which is encoded as "1111101000", then add the <FNC1> encoding into the encoding bits by padding "1111101000" into the encoding bit string, and continue Numeric encoding till the end of the following group or the first character other than 0~9. If the leading encoding scheme is Text mode, first end the Text mode by using Text mode terminator, then add the numeric mode indicator "0001" to start the numeric mode before the <FNC1> encoding, pad the "1111101000" encoding, and utilise the Han Xin Code encoding algorithm till the end of the following group. Then go to step 5
- e) Continue the Step 3 till all the groups are encoded, and add the GS1 mode terminator.

EXAMPLE 1

For the GS1 data (01)03453120000011 (17)191125 (10)ABCD1234

- a) Use $(1110\ 0001)_{bin}$ to indicate the start of GS1 mode;
- b) combine the data segments in to groups. There is no need to add the <FNC1> between the data element strings, so there is only one group:

01034531200000111719112510ABCD1234

- c) encode the group with the standard Han Xin Code algorithm.

For 01034531200000111719112510ABCD1234, it could be divided into two string, and encoded with Numeric and Text mode separately.

- I. For "01034531200000111719112510", the encode process is:

- i) Separate the string into groups of three digits: 010 345 312 000 001 117 191 125 10
- ii) Start with change to numeric mode (0001) and choose the Terminator: 111111110
- iii) Convert each group into its binary equivalent:

010→0000001010

345→0101011001

312→0100111000

000→0000000000

001→0000000001

117→0001110101

191→0010111111

125→0001111101

10→0000001010, then append the binary terminator (111111110)

- iv) Then the encoding binary sequence for “01034531200000111719112510” is **0001** 0000001010 0101011001 0100111000 0000000000 0000000001 0001110101 0010111111 0001111101 0000001010 111111110

- II. For “ABCD1234”, change to text mode (0010), then:

- i) Encode the characters in the Text1 sub-mode:

A→001010

B→001011

C→001100

D→001101

1→000001

2→000010

3→000011

4→000100

Then append text mode terminator (111111)

- ii) The encoding binary sequence for “ABCD1234” is **0010** 001010 001011 001100 001101 000001 000010 000011 000100 111111.

- iii) The final encoding binary sequence is

1110 0001 0001 0000001010 0101011001 0100111000 0000000000 0000000001 0001110101 0010111111 0001111101 0000001010 111111110 **0010** 001010 001011 001100 001101 000001 000010 000011 000100 111111 11111111.

EXAMPLE 2

For the GS1 data(01)03453120000011 (17)191125 (10)ABCD1234 (21)10,

- a) Use (**1110 0001**)bin to indicate the start of GS1 mode;
 b) Combine the data segments into groups. <FNC1> should be added between the AI (10) and AI(21) data element strings, so there are two groups:

01034531200000111719112510ABCD1234<FNC1>2110

- c) Encode the first group with the standard Han Xin Code algorithm (same as last example).

For “01034531200000111719112510ABCD1234”, it could be encoded as “**0001** 0000001010 0101011001 0100111000 0000000000 0000000001 0001110101 0010111111 0001111101 0000001010 111111110 **0010** 001010 001011 001100 001101 000001 000010 000011 000100 111111”.

- d) For <FNC1>2110, as for the leading encoding scheme is Text mode and the characters following <FNC1> are 0~9, it should be encoded as:

- I. Choose the Numeric mode indicator “**0001**”, and the Terminator “1111111101”

- II. <FNC1>should be encoded as “1111101000”

III. 2110 should continue the Numeric encoding, and the binary encoding is “0011010011 0000000000”.

IV. The final binary encoding sequence is

1110 0001 0001 0000001010 0101011001 0100111000 0000000000 0000000001 0001110101
0010111111 0001111101 0000001010 1111111110 **0010** 001010 001011 001100 001101 000001
000010 000011 000100 111111 **0001** 1111101000 0011010011 0000000000 1111111101 11111111

5.4.14 URI mode

5.4.14.1 General

Nowadays, Uniform Resource Identifier (URI) is widely used in business and daily life of every person. URI mode indicate the data represented in Han Xin Code is Uniform Resource Identifier (URI) reference to RFC 3986.

URI Mode is a highly efficient encoding scheme to encode URI string reference to RFC 3986. By recognizing the characteristics of RFC 3986, URI Mode use 3 charsets (URI-A, URI-B and URI-C), which should conform to [Annex M](#), and a Percent-Encoding charset to represent URI. In this scheme, URI-A charset includes the most common used characters or URI fragments (such as www., .com, http://, etc.) of URI; URI-B includes other characters defined in RFC 3986 and less common keywords; URI-C includes all characters and URI fragments defined in URI-A and URI-B. The analysis and encoding process is described in [5.4.14.3](#).

5.4.14.2 Mode indicator and terminator

The mode indicator of URI Mode is $(1110\ 0010)_{\text{bin}}$ and the mode terminator of URI Mode is $(111)_{\text{bin}}$.

5.4.14.3 Data analysis and encoding algorithm

- a) The mode indicator of URI mode of Han Xin Code is $(1110\ 0010)_{\text{bin}}$
- b) Using following methods and steps to analysis the input URI string:
 - I. Initially analysis the input URI string by following the rules described as follow in accordance with the charsets defined in c), find and record the initial encodings for each characters or character sequence of the input URI string:
 - i) If there are two character behind the character “%”, and these “%XX” character sequence meet the requirements of Percent-Encoding (RFC 3986 Section 2.1 Page 12) defined in RFC 3986, utilise Percent-Encoding charset to these “%XX” character sequence.
 - ii) If the character or character sequence can be encoded by using URI-A charset and URI-C charset, utilise URI-A charset as first choice.
 - iii) If the character or character sequence can be encoded by using URI-B charset and URI-C charset, utilise URI-B charset as first choice.
 - iv) If there are two ways by used URI-A charset to encode a character or a charset sequence at the same position(s), then utilise the way has the bigger encoding value as first choice.
 - v) If there are two ways by used URI-C charset to encode a character or a charset sequence at the same position(s), then utilise the way has the bigger encoding value as first choice.
 - II. After initially analysis described in I, optimize the data analysis result by using the following steps:
 - i) If the initial encoding for a character or character sequence utilise URI-A charset and URI-B charset is chosen to encode the next character or character sequence, then calculate how many encoding bits will be used to encode these two adjacent character sequence in URI-A and URI-B separately and how many encoding bits will be used by utilising URI-C

charset for the character sequence composed by these two next character sequence. Only if the “combined sequence” encoding bits number be less than or equal to the “separate sequences”, then utilise URI-C charset to encode the character sequence composed by these two adjacent character sequence.

- ii) If the initial encoding for a character or character sequence utilise URI-B charset to encode the character sequence and the encoding for the next character or character sequence is URI-A charset, then calculate how many encoding bits will be used to encode these two adjacent character sequence separately and how many encoding bits will be used by utilising URI-C charset for the character sequence composed by these two next character sequence. Only if the “combined sequence” encoding bits number less than or equal to the “separate sequences”, then utilise URI-C charset to encode the character sequence composed by these two adjacent character sequence.
 - iii) If the initial encoding for a character or character sequence utilise URI-A charset to encode the character sequence and utilise URI-C charset to the next character sequence, then calculate how many encoding bits will be used to encode these two adjacent character sequence separately and how many encoding bits will be used by utilising URI-C charset for the character sequence composed by these two next character sequence. Only if the “combined sequence” encoding bits number less than will be less than or equal to the “separate sequences”, then utilise URI-C charset to encode the character sequence composed by these two next character sequence.
 - iv) If the initial encoding for a character or character sequence utilise URI-B charset to encode the character sequence and utilise URI-C charset to the next character sequence, then calculate how many encoding bits will be used to encode these two adjacent character sequences separately and how many encoding bits will be used by utilising URI-C charset for the character sequence composed by these two next character sequence. Only if the “combined sequence” encoding bits number less than or equal to the “separate sequences”, then utilise URI-C charset to encode the character sequence composed by these two next character sequence.
 - v) If the initial encoding for a character or character sequence utilise URI-C charset to encode the character sequence and utilise URI-A charset to the next character sequence, then calculate how many encoding bits will be used to encode these two next character sequence separately and how many encoding bits will be used by utilising URI-C charset for the character sequence composed by these two next character sequence. Only if the “combined sequence” encoding bits number less than or equal to the “separate sequences”, then utilise URI-C charset to encode the character sequence composed by these two next character sequence.
 - vi) If the initial encoding for a character or character sequence utilise URI-C charset to encode the character sequence and utilise URI-B charset to the next character sequence, then calculate how many encoding bits will be used to encode these two next character sequences and how many encoding bits will be used by utilising URI-C charset for the character sequence composed by these two next character sequence. Only if the “combined sequence” encoding bits number less than or equal to the “separate sequences”, then utilise URI-C charset to encode the character sequence composed by these two next character sequence.
 - vii) Repeat above steps described in this step until there is no more optimization can be made.
- c) Utilise the analysis result by b), use the indicators in [Table 8](#) and the corresponding charsets to encode the input URI string:

Table 8 — Encoding indicators of each charsets in URI mode

Charset	Charset indicator
URI-A	(001) _{bin}
URI-B	(010) _{bin}
URI-C	(011) _{bin}
Percent-Encoding	(100) _{bin}
URI Mode Terminator	(111) _{bin}
NOTE 1 URI-A charset is defined in Table M.1. NOTE 2 URI-B charset is defined in Table M.2. NOTE 3 URI-C charset is defined in Table M.3. NOTE 4 Percent-Encoding: For the two characters “XX” followed to the character “%”, and the “XX” meet the requirements of hex “00” to “FF” or “00” to “ff”, use Percent-Encoding charset to encode the “%XX” character sequence. In the encoding process of Percent-Encoding, after the charset indicator (100)_{bin} add a 8-bit counter to encode the length of the number of how many Percent-Encoding sequences are, after the counter use 8-bit binary string to encode each XX.	

d) The mode terminator of URI mode of Han Xin Code is (111)_{bin}.

The C implementation source codes for encoding and decoding are listed in [Annex O](#).

EXAMPLE 1

For the URI <http://www.example.com>

a) The URI mode indicator is (1110 0010)_{bin}

b) Analysis of the input URI string:

<http://www.example.com>

I. Initially analyse the input URI strings:

According to the algorithm, the input URI string could be encoded by URI-A charset(see [Table 9](#)), which is started with the URI- A mode indicator (001).

Table 9 — URI mode “<http://www.example.com>” encoding

Character/URI fragment	Encoding Value	Encoding (bits)
http://	49	110001
www.	56	111000
e	4	000100
x	23	010111
a	0	000000
m	12	001100
p	15	001111
l	11	001011
e	4	000100
.com	57	111001
Terminator of URI-A	63	111111

II. Since only URI-A charset is used to encode the whole URI, there is no need to optimize it with other URI character sets.

- c) Add the URI mode Terminator (111), then the final encoding binary sequence is (1110 0010 001 110001 11100 0 000100 010111 000000 001100 001111 001011 000100 111001 111111 111)_{bin}

EXAMPLE 2

For the URI: dictionary.cambridge.org/zhs/词典/英语/soil?q=soil

The URL encoding is:
["https://dictionary.cambridge.org/zhs/%E8%AF%8D%E5%85%B8/%E8%8B%B1%E8%AF%AD/soil?q=soil"](https://dictionary.cambridge.org/zhs/%E8%AF%8D%E5%85%B8/%E8%8B%B1%E8%AF%AD/soil?q=soil)

The steps for encoding follow.

- a) The mode indicator is (1110 0010)_{bin}

- b) Analysis of the input URI string:

<https://dictionary.cambridge.org/zhs/%E8%AF%8D%E5%85%B8/%E8%8B%B1%E8%AF%AD%E9%98%BF%E6%8B%89%E4%BC%AF%E8%AF%AD/soil?q=soil>

- I. Initial analysis of the input URI strings:

According to the algorithm, the input URI string could be encoded by using a combination of the URI-A charset and the Percent-Encoding mode.

- II. The string "<https://dictionary.cambridge.org/zhs/>" could be encoded in accordance with [Table 10](#).

Table 10 — URI mode "<https://dictionary.cambridge.org/zhs/>" encoding

Character/URI fragment	Encoding value	Encoding (bits)
http://	49	110001
d	3	000011
i	8	001000
c	2	000010
t	19	010011
i	8	001000
o	14	001110
n	13	001101
a	0	000000
r	17	010001
y	24	011000
.	36	100100
c	2	000010
a	0	000000
m	12	001100
b	1	000001
r	17	010001
i	8	001000
d	3	000011
g	6	000110

Table 10 (continued)

Character/URI fragment	Encoding value	Encoding (bits)
e	4	000100
.org	60	111100
/	37	100101
z	25	011001
h	7	000111
s	18	010010
/	37	100101
Terminator of URI-A	63	111111

III. The Percent-Encodings

The "%" sequence "%E8%AF%8D%E5%85%B8" is encoded by the Percent-Encoding charset (**100**). There are 6 "%HH" formatted bytes, so set the counter to be "06"_{HEX}, and the encoding binary sequence for "%E8%AF%8D%E5%85%B8" is

"100 00000110 11101000 10101111 10001101 11100101 10000101 10111000"

The "/" is encoded in URI-A charset, and the encoding binary sequence for "/" is

"001 100101 111111";

The "%" sequence "%E8%8B%B1%E8%AF%AD" is encoded by returning to Percent-Encoding charset (**100**). There are 6 "%HH" formatted bytes, so set the counter to be "06"_{HEX}, and the encoding binary sequence for "%E8%8B%B1%E8%AF%AD" is

"100 00000110 11101000 10001011 10110001 11101000 10101111 10101101"

IV. The rest of the URI "/soil?q=soil" could also be encoded by URI-A charset.(see [Table 11](#))

Table 11 — URI mode "/soil?q=soil" encoding

Character/URI fragment	Encoding value	Encoding (bits)
/	37	100101
s	18	010010
o	14	001110
i	8	001000
l	11	001011
?	43	101011
q	16	010000
=	45	101101
s	18	010010
o	14	001110
i	8	001000
l	11	001011
Terminator of URI-A	63	111111

- V. As for only URI-A charset and Percent charset is chosen to encode the whole URI, there is no need to optimize it.
- c) Add all the indicator before the encoding binary sequence.
- d) Add the URI mode Terminator , and the final encoding binary sequence is **(1110 0010 001 000011 001000 000010 010011 001000 001110 001101 000000 010001 011000 100100 000010 000000 001100 000001 010001 001000 000011 000110 000100 111100 100101 011001 000111 010010 100101 111111 100 00000110 11101000 10101111 10001101 11100101 10000101 10111000 001 100101 111111 100 00000110 11101000 10001011 10110001 11101000 10101111 10101101 001 100101 010010 001110 001000 001011 101011 010000 101101 010010 001110 001000 001011 111111 111)**_{bin}

5.4.15 Mixed modes encoding

Han Xin Code supports any mixing of data sequence of the Numeric, Text, ECI, Binary Bytes and 4 GB18030 modes. The definition of encoding rules used in mixed modes is as follows.

There is the option for a symbol to contain sequences of data in one mode and then to change modes if the data content requires it. Table illustrates the structure of data containing n segments.

Applied to the original data, hybrid encoding rules are designed to transfer information encoding from one mode to another. The basic hybrid encoding structure is as [Table 12](#).

Table 12 — Format of mixed mode data

Segment 1 (mode except Binary Byte mode)	mode 1 indicator	data 1	mode terminator 1
Segment 2 (Binary Byte mode)	mode indicator 2	data count indicator 2	data 2
Segment 3 (mode except Binary Byte mode)	mode indicator 3	data 3	mode terminator 3
...	...		
Segment n	mode indicator n	data n	mode terminator n

NOTE Only binary mode applies count indicator while other encoding modes apply mode terminator to denote the mode's end.

Numbers and common Chinese characters may be encoded in several modes. The methods of suitable encoding mode selection are:

- For numeric data, if the preceding mode of numeric data to be encoded is in any other mode except Text mode, Numeric mode will be selected. If the preceding mode is Text mode, the numeric data will be encoded in following steps:
 - If in sub-mode Text1 and its length is more than 11 digits, Text encoding mode will be ended and encoding switched to Numeric mode. Otherwise, Text mode will remain in effect.
 - If in Text2 sub-mode of Text mode and its length is more than 8 digits, Text encoding mode will be ended and encoding switched to Numeric mode. Otherwise, Text mode will remain in effect.
- For common Chinese characters in GB18030 2-byte region, its hybrid encoding rules are as follows:
 - For Common Chinese Characters in Region One, if the preceding encoding mode is any one except GB18030 2-byte Region mode, Common Character In Region One encoding mode will be applied. However, if the previous mode is GB18030 2-byte Region and its length is more than 11 characters, encoding will be switched to Common Chinese Character in Region One encoding mode. Otherwise, GB18030 2-byte encoding mode will remain in effect.

- b. For Common Chinese Characters in Region Two, if the preceding encoding mode is any one except GB18030 2-byte Region, Common Character In Region Two encoding mode will be applied. However, if the preceding mode is GB18030 2-byte Region and its length is more than 11 characters, it will shift to Common Chinese Character in Region Two encoding mode. Otherwise, GB18030 2-byte encoding mode will remain in effect.

5.5 Error detection and correction

5.5.1 General

Han Xin Code symbols employ Reed-Solomon error correction.

According to the symbol version and error correction level, divide information codewords sequence into blocks (see [Annex B](#)) to generate an error correction codewords sequence per block.

The polynomial arithmetic for Han Xin Code shall be calculated using bit-wise modulo 2 arithmetic and byte-wise modulo $(101100011)_{\text{bin}}$ (decimal 355) arithmetic. This is based on the Galois field of 2^8 with $(101100011)_{\text{bin}}$ representing the field's prime modulus polynomial: $x^8+x^6+x^5+x+1$. Thirty-two different generator polynomials which shall in accordance with [Annex D](#) are used for generating the appropriate error correction codewords.

5.5.2 Generating the error correction codewords

The error correction codewords are the remainder after dividing the data codewords by a polynomial $g(x)$ used for Reed-Solomon algorithm (see [Annex D](#)).

The data codewords are the coefficients of the terms of a polynomial with the coefficient of the highest term being the first data codeword and the lowest power term being the last data codeword before the first error correction codeword. The highest order coefficient of the remainder is the first error correction codeword and the zero power coefficient is the last error correction codeword and the last codeword in the block.

The error correction codewords can be generated by using the division circuit as shown in [Figure 18](#). The registers b_0 through b_{k-1} are initialized as zeros. There are two phases to generate the encoding. In the first phase, with the switch in the down position the data codewords are passed both to the output and the circuit. The first phase is complete after n clock pulses. In the second phase ($n+1 \dots n+k$ clock pulses), with the switch in the up position, the error correction codewords are generated by flushing the registers in order and complementing the output while keeping the data input at 0. The codewords output from the shift register are in the order that they are to be placed in the symbol. If interleaving is used, the codewords will not be placed in consecutive symbol characters.