INTERNATIONAL STANDARD

ISO/IEC 16022

Second edition
2006-09-15

Information technology — Automatic identification and data capture techniques — Data Matrix bar code symbology specification

Technologies de l'information — Techniques d'identification automatique et de capture des données — Spécification de symbologie de code à barres Data Matrix

Reference number
ISO/IEC 16022:2006(E)

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

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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

ISO/IEC 16022 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

This second edition cancels and replaces the first edition (ISO/IEC 16022:2000), which has been technically revised. It also incorporates the Technical Corrigendum ISO/IEC 16022:2000/Cor.1:2004.

Introduction

Data Matrix is a two-dimensional matrix symbology which is made up of nominally square modules arranged within a perimeter finder pattern. Though primarily shown and described in this International Standard as a dark symbol on light background, Data Matrix symbols can also be printed to appear as light on dark.

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. The publication of standardised symbology specifications is designed to achieve this.

Information technology — Automatic identification and data capture techniques — Data Matrix bar code symbology specification

1 Scope

This International Standard defines the requirements for the symbology known as Data Matrix. It specifies the Data Matrix symbology characteristics, data character encodation, symbol formats, dimensions and print quality requirements, error correction rules, decoding algorithm, and user-selectable application parameters.

It applies to all Data Matrix symbols produced by any printing or marking technology.

2 Normative references

The following referenced documents are indispensable for the application 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 15424, *Information technology — Automatic identification and data capture techniques — Data Carrier Identifiers (including Symbology Identifiers)*

ISO/IEC 19762-1, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary — Part 1: General terms relating to AIDC*

ISO/IEC 19762-2, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary — Part 2: Optically readable media (ORM)*

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

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

ISO/IEC 646:1991, *Information technology — ISO 7-bit coded character set for information interchange*

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

ISO/IEC 8859-5:1999, *Information technology — 8-bit single-byte coded graphic character sets — Part 5: Latin/Cyrillic alphabet*

AIM Inc. ITS/04-001 International Technical Standard: *Extended Channel Interpretations — Part 1: Identification Schemes and Protocol*

3 Terms, definitions, symbols and mathematical/logical notations

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19762-1, ISO/IEC 19762-2 and the following apply.

3.1.1
codeword
symbol character value, an intermediate level of coding between source data and the graphical encodation in the symbol

3.1.2
module
single cell in a matrix symbology used to encode one bit of data, nominally a square shape in Data Matrix

3.1.3
convolutional coding
error checking and correcting (ECC) algorithm that processes a set of input bits into a set of output bits that can recover from damage by breaking the input bits into blocks, then convolving each input block with the contents of a multi-stage shift register to produce protected output blocks

NOTE These encoders can be constructed in hardware using input and output switches, shift registers, and exclusive-or (XOR) gates.

3.1.4
pattern randomising
procedure to convert an original bit pattern to another bit pattern, intended to reduce the probability of repeating patterns occurring in the symbol, by inverting selected bits

3.2 Symbols

For the purposes of this document, the following mathematical symbols apply unless defined locally.

d number of error correction codewords

e number of erasures

k (for ECC 000 - 140) the number of bits in a complete segment input to the state machine to generate the convolutional code (for ECC 200) total number of error correction codewords

m the memory order of the convolutional code

n (for ECC 000 - 140) the number of bits in a complete segment generated by the state machine producing the convolutional code (for ECC 200) total number of data codewords

N the numerical base in an encodation scheme

p number of codewords reserved for error detection

S symbol character

t number of errors

u the input bit segment to the state machine, taken k bits at a time

v the output bit segment from the state machine, generated n bits at a time

X horizontal and vertical width of a module

ε error correction codeword

3.3 Mathematical/logical notations

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

div integer division operator

mod integer remainder after division

XOR exclusive-or logic function whose output is one only when its two inputs are not equivalent.

LSB least significant bit

MSB most significant bit

4 Symbol description

4.1 Basic characteristics

Data Matrix is a two-dimensional matrix symbology.

There are two types:

ECC 200 which uses Reed-Solomon error correction. ECC 200 is recommended for new applications.

ECC 000 - 140 with several available levels of convolutional error correction, referred to as ECC 000, ECC 050, ECC 080, ECC 100 and ECC 140 respectively. ECC 000 - 140 should only be used in closed applications where a single party controls both the production and reading of the symbols and is responsible for overall system performance.

The characteristics of Data Matrix are:

a) Encodable character set:

1) values 0 – 127 in accordance with the US national version of ISO/IEC 646

NOTE 1 This version consists of the G0 set of ISO/IEC 646 and the C0 set of ISO/IEC 6429 with values 28 – 31 modified to FS, GS, RS and US respectively.

2) values 128 - 255 in accordance with ISO 8859-1. These are referred to as extended ASCII.

b) Representation of data: A dark module is a binary one and a light module is a zero.

NOTE 2 This International Standard specifies Data Matrix symbols in terms of dark modules marked on a light background. However, subclause 4.2 provides that symbols may also be produced with light and dark modules reversed in colour (see 4.2), and in such symbols references in this International Standard to dark modules should be taken as references to light modules, and vice versa.

c) Symbol size in modules (not including quiet zone):

ECC 200	10 x 10 to 144 x 144 even values only
ECC 000 – 140	9 x 9 to 49 x 49, odd values only

d) Data characters per symbol (for maximum symbol size in ECC200):

1) Alphanumeric data: up to 2 335 characters

2) 8-bit byte data: 1 555 characters

3) Numeric data: 3 116 digits.

e) Selectable error correction:

ECC 200: Reed-Solomon error correction.

ECC 000 - 140: Four levels of convolutional error correction, plus the option to apply only error detection

f) Code type: Matrix

g) Orientation independence: Yes

4.2 Summary of additional features

The following summarises additional features which are inherent or optional in Data Matrix:

a) 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). The specifications in this International Standard are based on dark images on a light background, therefore references to dark or light modules should be taken as references to light or dark modules respectively in the case of symbols produced with reflectance reversal.

b) Extended Channel Interpretations: (ECC 200 only, 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.

c) Rectangular symbols: (ECC 200 only, optional): Six symbol formats are specified in a rectangular form.

d) Structured append: (ECC 200 only, optional): This allows files of data to be represented in up to 16 Data Matrix symbols. The original data can be correctly reconstructed regardless of the order in which the symbols are scanned.

4.3 Symbol structure

Each Data Matrix symbol consists of data regions which contain nominally square modules set out in a regular array. In larger ECC 200 symbols, data regions are separated by alignment patterns. The data region, or set of data regions and alignment patterns, is surrounded by a finder pattern, and this shall in turn be surrounded on all four sides by a quiet zone border. Figure 1 illustrates an ECC 140 and two representations of an ECC 200 symbol.

(a) ECC200, dark on light (b) ECC200, light on dark (c) ECC140, dark on light

Figure 1 — ECC 200 (a & b) and ECC 140 (c) encoding "A1B2C3D4E5F6G7H8I9J0K1L2"

4.3.1 Finder pattern

The finder pattern is a perimeter to the data region and is one module wide. Two adjacent sides, the left and lower sides, forming the L boundary, are solid dark lines; these are used primarily to determine physical size, orientation and symbol distortion. The two opposite sides are made up of alternating dark and light modules. These are used primarily to define the cell structure of the symbol, but also can assist in determining physical size and distortion. The extent of the quiet zone is indicated by the corner marks in Figure 1.

4.3.2 Symbol sizes and capacities

ECC 200 symbols have an even number of rows and an even number of columns. Some symbols are square with sizes from 10 x 10 to 144 x 144 not including quiet zones. Some symbols are rectangular with sizes from 8 x 18 to 16 x 48 not including quiet zones. All ECC 200 symbols can be recognised by the upper right corner module being light. The complete attributes of ECC 200 symbols are given in Table 7 in Section 5.5.

ECC 000 - 140 symbols have an odd number of rows and an odd number of columns. Symbols are square with sizes from 9 x 9 to 49 x 49 (modules) not including quiet zones. These symbols can be recognised by the upper right corner module being dark. The complete attributes of ECC 000 - 140 symbols are given in Annex G.

5 ECC 200 requirements

5.1 Encode procedure overview

This section provides an overview of the encoding procedure. Following sections will provide more details. An encoding example for ECC 200 is given in Annex O. The following steps convert user data to an ECC 200 symbol:

Step 1: Data encodation

> Analyse the data stream to identify the variety of different characters to be encoded. ECC 200 includes various encodation schemes which allow a defined set of characters to be converted into codewords more efficiently than the default scheme. Insert additional codewords to switch between the encodation schemes and to perform other functions. Add pad characters as needed to fill the required number of codewords. If the user does not specify the matrix size, then choose the smallest size that accommodates the data. A complete list of matrix sizes is shown in Section 5.5, Table 7.

Table 1 — Encodation schemes for ECC 200

Encodation scheme	Characters	Bits per data character
ASCII	double digit numerics	4
	ASCII values 0 - 127	8
	Extended ASCII values 128 - 255	16
C40	Upper-case alphanumeric	5,33
	Lower case and special characters	10,66[a]
Text	Lower-case alphanumeric	5,33
	Upper case and special characters	10,66[b]
X12	ANSI X12 EDI data set	5,33
EDIFACT	ASCII values 32 - 94	6
Base 256	All byte values 0 - 255	8

[a] encoded as two C40 values as result of use of a shift character

[b] encoded as two Text values as result of use of a shift character

Step 2: Error checking and correcting codeword generation

For symbols with more than 255 codewords, sub-divide the codeword stream into interleaved blocks to enable the error correction algorithms to be processed as shown in Annex A. Generate the error correction codewords for each block. The result of this process expands the codeword stream by the number of error correction codewords. Place the error correction codewords after the data codewords.

Step 3: Module placement in matrix

Place the codeword modules in the matrix. Insert the alignment pattern modules, if any, in the matrix. Add the finder pattern modules around the matrix.

5.2 Data encodation

5.2.1 Overview

The data may be encoded using any combination of six encodation schemes (see Table 1). ASCII encodation is the basic scheme. All other encodation schemes are invoked from ASCII encodation and return to this scheme. The compaction efficiencies given in Table 1 need to be interpreted carefully. The best scheme for a given set of data may not be the one with the fewest bits per data character. If the highest degree of compaction is required, account has to be taken of switching between encodation schemes and between code sets within an encodation scheme (see Annex P). It should also be noted that even if the number of codewords is minimised, the codeword stream might need to be expanded to fill a symbol. This fill process is done using pad characters.

5.2.2 Default character interpretation

The default character interpretation for character values 0 to 127 shall conform to ISO/IEC 646. The default character interpretation for character values 128 to 255 shall conform to ISO 8859-1: Latin Alphabet No. 1. The graphical representation of data characters shown throughout this document complies with the default interpretation. This interpretation can be changed using Extended Channel Interpretation (ECI) escape sequences, see 5.4. The default interpretation corresponds to ECI 000003.

5.2.3 ASCII encodation

ASCII encodation is the default set for the first symbol character in all symbol sizes. It encodes ASCII data, double density numeric data and symbology control characters. Symbology control characters include function characters, the pad character and the switches to other code sets. ASCII data is encoded as codewords 1 to 128 (ASCII value plus 1). Extended ASCII (data values 128 to 255) is encoded using the upper shift symbology control character (see 5.2.4.2). The digit pairs 00 to 99 are encoded with codewords 130 to 229 (numeric value plus 130). The ASCII code assignments are shown in Table 2.

Table 2 — ASCII encodation values

Codeword	Data or function
1 - 128	ASCII data (ASCII value + 1)
129	Pad
130 - 229	2-digit data 00 - 99 (Numeric Value + 130)
230	Latch to C40 encodation
231	Latch to Base 256 encodation
232	FNC1
233	Structured Append
234	Reader Programming
235	Upper Shift (shift to Extended ASCII)
236	05 Macro
237	06 Macro
238	Latch to ANSI X12 encodation
239	Latch to Text encodation
240	Latch to EDIFACT encodation
241	ECI Character
242 - 255	Not to be used in ASCII encodation

5.2.4 Symbology control characters

ECC 200 symbols have several special symbology control characters, which have particular significance to the encodation scheme. These characters shall be used to instruct the decoder to perform certain functions or to send specific data to the host computer as described in 5.2.4.1 to 5.2.4.9. These symbology control characters, with the exception of values from 242 through 255, are found in the ASCII encodation set (see Table 2).

5.2.4.1 Latch characters

A Latch Character shall be used to switch from ASCII encodation to one of the other encodation schemes. All codewords which follow a Latch Character shall be compacted according to the new encodation scheme. The encodation schemes have different methods for returning to the ASCII encodation set.

5.2.4.2 Upper Shift character

The Upper Shift character is used in combination with an ASCII value (1 - 128) to encode an extended ASCII character (129-255). An extended ASCII character encoded in the ASCII, C40, or Text encodation scheme requires a preceding Upper Shift character and the extended ASCII character value decreased by 128 is then encoded according to the rules of the encodation scheme. In ASCII encodation, the Upper Shift character is represented by codeword 235. The reduced data value (i.e. ASCII value minus 128) is transformed into its codeword value by adding 1. For example, to encode ¥ (Yen currency symbol) (ASCII value 165), an Upper Shift character (Codeword 235) is followed by value 37 (165 - 128), which is encoded as codeword 38. If there are long data strings of characters from the extended ASCII range, a Latch to Base 256 encodation should be more efficient.

5.2.4.3 Pad character

If the encoded data, irrespective of the encodation scheme in force, does not fill the data capacity of the symbol, pad characters (value 129 in ASCII encodation) shall be added to fill the remaining data capacity of the symbol. The pad characters shall only be used for this purpose. Before inserting pad characters, it is necessary to return to ASCII encodation if in any other encodation mode. The 253-State pattern randomising algorithm is applied to the pad characters starting at the second pad character and continuing to the end of the symbol (see Annex B.1).

5.2.4.4 Extended Channel Interpretation character

An Extended Channel Interpretation (ECI) character is used to change from the default interpretation used to encode data. The Extended Channel Interpretation protocol is common across a number of symbologies and its application to ECC 200 is defined more fully in 5.4. The ECI character shall be followed by one, two, or three codewords which identify the ECI being invoked. The new ECI remains in place until the end of the encoded data, or until another ECI character is used to invoke another interpretation.

5.2.4.5 Shift characters in C40 and Text encodation

In C40 and Text Encoding, three special characters, called shift characters, are used as a prefix to one of 40 values to encode about three quarters of the ASCII characters. This allows the remaining ASCII characters to be encoded in a more condensed way with single values.

5.2.4.6 FNC1 alternate data type identifier

To encode data to conform to specific industry standards as authorised by AIM Inc., a FNC1 character shall appear in the first or second symbol character position (or in the fifth or sixth data positions of the first symbol of Structured Append). FNC1 encoded in any other position is used as a field separator and shall be transmitted as $^{G}{}_{S}$ control character (ASCII value 29).

5.2.4.7 Macro characters

Data Matrix provides a means of abbreviating an industry specific header and trailer in one symbol character. This feature exists to reduce the number of symbol characters needed to encode data in a symbol using certain structured formats. A Macro character must be in the first character position of a symbol. They shall not be used in conjunction with Structured Append and their functions are defined in Table 3. The header shall be transmitted as a prefix to the data stream and the trailer shall be transmitted as a suffix to the data stream. The symbology identifier, if used, shall precede the header.

Table 3 — Macro functions

Macro codeword	Name	Interpretation	
		Header	Trailer
236	05 Macro	[)>$^{R}{}_{S}$05$^{G}{}_{S}$	$^{R}{}_{S}{}^{E}O_{T}$
237	06 Macro	[)>$^{R}{}_{S}$06$^{G}{}_{S}$	$^{R}{}_{S}{}^{E}O_{T}$

5.2.4.8 Structured Append character

A Structured Append character is used to indicate that the symbol is part of a Structured Append sequence according to the rules defined in 5.6.

5.2.4.9 Reader Programming character

A Reader Programming character indicates that the symbol encodes a message used to program the reader system. The Reader Programming character shall appear as the first codeword of the symbol and Reader Programming shall not be used with Structured Append.

5.2.5 C40 encodation

The C40 encodation scheme is designed to optimise the encoding of upper-case alphabetic and numeric characters but also enables other characters to be encoded by the use of shift characters in conjunction with the data character.

C40 characters are partitioned into 4 subsets. Characters of the first set, called the basic set, are the three special shift characters, the space character, and the ASCII characters A-Z and 0-9. They are assigned to a single C40 values. Characters of the other sets are assigned to one of the three shift characters, pointing to one of the 3 remaining subset, followed by one of the C40 values (see Annex C, Table C.1).

As a first stage, each data character is converted into a single C40 value or a pair of C40 values. The complete string of C40 values is then decomposed into groups of three values (special rules apply if one or two values remain at the end, see 5.2.5.2.). Each triplet (C1, C2, C3) is then encoded into a 16-bit value according to the formula: (1600 * C1) + (40 * C2) + C3 + 1. Each 16-bit value is then separated into 2 codewords by taking the most significant 8 bits and the least significant 8 bits.

5.2.5.1 Switching to and from C40 encodation

It is possible to switch to C40 encodation from ASCII encodation using the appropriate latch codeword (230). Codeword 254 immediately following a pair of codewords in C40 encodation acts as an Unlatch codeword to switch back to ASCII encodation. Otherwise, the C40 encodation remains in effect to the end of the data encoded in the symbol.

5.2.5.2 C40 encodation rules

Each pair of codewords represents a 16-bit value where the first codeword represents the most significant 8 bits. Three C40 values ($C1$, $C2$, $C3$) shall be encoded as:

$$(1600 * C1) + (40 * C2) + C3 + 1$$

which produces a value from 1 to 64000. Figure 2 illustrates three C40 values compacted into two codewords. Characters in the Shift 1, Shift 2 and Shift 3 sets shall be encoded by first encoding the appropriate shift character, and then the C40 value for the data. C40 encodation may be in effect at the end of the symbol's codewords which encode data.

The following rules apply when only one or two symbol characters remain in the symbol before the start of the error correction codewords:

a) If two symbol characters remain and three C40 values remain to be encoded (which may include both data and shift characters) encode the three C40 values in the last two symbol characters. A final Unlatch codeword is not required.

b) If two symbol characters remain and two C40 values remain to be encoded (the first C40 value may be a shift or data character but the second must represent a data character); encode the two remaining C40 values followed by a pad C40 value of 0 (Shift 1) in the last two symbol characters. A final Unlatch codeword again is not required.

c) If two symbol characters remain and only one C40 value (data character) remains to be encoded, the first symbol character is encoded as an Unlatch character and the last symbol character is encoded with the data character using the ASCII encodation scheme.

d) If one symbol character remains and one C40 value (data character) remains to be encoded, the last symbol character is encoded with the data character using the ASCII encodation scheme. The Unlatch character is not encoded, but is assumed, before the last symbol character.

   In all other cases, either an Unlatch character is used to exit the C40 encodation scheme before the end of the symbol, or a larger symbol size is required to encode the data.

Data characters	AIM
C40 values	14, 22, 26
Calculate 16-bit value	(1600 * 14) + (40 * 22) + 26 + 1 = 23307
1st codeword: (16-bit value) div 256	23307 div 256 = 91
2nd codeword: (16-bit value) mod 256	23307 mod 256 = 11
Codewords	91, 11

Figure 2 — Example of C40 encoding

5.2.5.3 Use of Upper Shift with C40

In C40 encodation the Upper Shift character is not a symbology function character but a shift within the encodation set. When a data character from the extended ASCII character range is encountered, three or four values in C40 encodation need to be encoded according to the following rule:

IF [ASCII value - 128] is in the Basic Set then:

[1(Shift 2)] [30(Upper Shift)] [V(ASCII value - 128)]

ELSE

[1(Shift 2)] [30(Upper Shift)] [0, 1, or 2(Shift 1, 2, or 3)] [V(ASCII value - 128)]

In the rule the number in [ ] equates to the C40 values from Annex C.1; V has been used to indicate the appropriate C40 value.

5.2.6 Text encodation

Text encodation is designed to encode normal printed text, which is predominantly lowercase characters. It is similar in structure to the C40 encodation set, except that lowercase alphabetic characters are directly encoded (i.e. without using a shift). Upper-case alphabetic characters are preceded by a Shift 3. The full Text encodation character set assignments are shown in Annex C, Table C.2.

5.2.6.1 Switching to and from Text encodation

It is possible to switch to Text encodation from ASCII encodation using the appropriate latch codeword (239). Codeword 254 immediately following a pair of codewords in text encodation acts as an Unlatch codeword to switch back to ASCII encodation. Otherwise, the Text encodation remains in effect to the end of the data encoded in the symbol.

5.2.6.2 Text encodation rules

The rules for C40 encodation apply.

5.2.7 ANSI X12 encodation

ANSI X12 encodation is used to encode the standard ANSI X12 electronic data interchange characters, which are compacted three data characters to two codewords in a manner similar to C40 encodation. It encodes upper-case alphabetic characters, numerics, space and the three standard ANSI X12 terminator and separator characters. The ANSI X12 code assignments are shown in Table 4. There are no shift characters in the ANSI X12 encodation set.

Table 4 — ANSI X12 encodation set

X12 value	Encoded characters	ASCII values
0	X12 segment terminator <CR>	13
1	X12 segment separator *	42
2	X12 sub-element separator >	62
3	space	32
4 - 13	0 - 9	48 - 57
14 - 39	A - Z	65 - 90

5.2.7.1 Switching to and from ANSI X12 encodation

It is possible to switch to ANSI X12 encodation from ASCII encodation using the appropriate latch codeword (238). Codeword 254 immediately following a pair of codewords in ANSI X12 encodation acts as an Unlatch codeword to switch back to ASCII encodation. Otherwise, the ANSI X12 encodation remains in effect to the end of the data encoded in the symbol.

5.2.7.2 ANSI X12 encodation rules

The rules of C40 encodation apply. The exception is at the end of encoding ANSI X12 data. If the data characters do not fully utilise pairs of codewords, then following the last complete pair of codewords switch to ASCII using codeword 254 and continue using ASCII encodation, except when a single symbol character is left at the end before the first error correction character. This single symbol character uses the ASCII encodation scheme without requiring an Unlatch codeword.

5.2.8 EDIFACT encodation

The EDIFACT encodation scheme includes 63 ASCII values (values from 32 to 94) plus an Unlatch character (binary 011111) to return to ASCII encodation. EDIFACT encodation encodes four data characters in three codewords. It includes all the numeric, alphabetic and punctuation characters defined in the EDIFACT Level A character set without any of the shifts required in C40 encodation.

5.2.8.1 Switching to and from EDIFACT encodation

It is possible to switch to EDIFACT encodation from ASCII encodation using the appropriate latch codeword (240). The Unlatch character in EDIFACT encodation shall be used as a terminator at the end of EDIFACT encodation, which reverts to ASCII encodation.

5.2.8.2 EDIFACT encodation Rules

The EDIFACT encodation character set is defined in Annex C, Table C.3. There is a simple relationship between the 6-bit EDIFACT value and the ASCII 8-bit byte. The leading two bits of the 8-bit byte are ignored to create the EDIFACT 6-bit value, as illustrated in Figure 3. Strings of four EDIFACT characters are encoded in three codewords. For a simple encodation process, the leading two bits of the 8-bit byte are removed. The remaining 6-bit byte is the EDIFACT value and shall be directly encoded into the codeword as illustrated in Figure 4. When EDIFACT encodation is terminated with the Unlatch character, any remaining bits left in the single symbol character shall be filled with zeros. ASCII mode starts with the next symbol character. If EDIFACT encodation is in effect at the end of the symbol before the first error correction character, and only one or two codewords remain after the last EDIFACT codeword triplet, these remaining codewords shall be encoded in ASCII encodation without requiring an Unlatch character.

Data character	ASCII		EDIFACT value
	Decimal value	8-bit binary value	
A	65	01000001	000001
9	57	00111001	111001

NOTE During the decode process, if the leading (6th) bit is 1, the bits 00 are prefixed to create the 8-bit byte. If the leading (6th) bit is 0, the bits 01 are prefixed to create the 8-bit byte. The exception to this is the EDIFACT value 011111 which is the symbology control Unlatch character to return to ASCII encodation.

Figure 3 — The relationship between the EDIFACT value and the 8-bit byte value

Data characters	D			A			T			A		
Binary values (Table C.3)	00	01	00	00	00	01	01	01	00	00	00	01
Divide into 3 8-bit bytes	00	01	00	00	00	01	01	01	00	00	00	01
Codeword values	16				21				1			

Figure 4 — Example of EDIFACT encodation

5.2.9 Base 256 encodation

The Base 256 encodation scheme shall be used to encode any 8-bit byte data, including extended channel interpretations and binary data. The default interpretation is defined in 5.2.2. The 255-State pattern randomising algorithm is applied to each Base 256 sequence within the encoded data (see B.2). It starts after the latch to Base 256 encodation and ends at the last character specified by the Base 256 field length.

5.2.9.1 Switching to and from Base 256 encodation

It is possible to switch to Base 256 encodation from ASCII encodation using the appropriate latch codeword (231). At the end of Base 256 encodation, encodation automatically reverts to ASCII encodation. The appropriate ECI, if other than the default, shall be invoked prior to switching. The ECI sequence need not occur immediately before switching to Base 256 encodation.

5.2.9.2 Base 256 encodation rules

After switching to Base 256 encodation, the first one (*d1*) or two (*d1*, *d2*) codewords define the data field length in bytes. Table 5 specifies how the field length is defined. Thereafter, all encodation shall be of the byte values.

Table 5 — Base 256 field length

Field Length	Values of *d1*, *d2*	Permitted Values of d
Remainder of Symbol	*d1* = 0	*d1* = 0
1 to 249	*d1* = length	*d1* = 1 to 249
250 to 1555	*d1* = (length DIV 250) + 249	*d1* = 250 to 255
	d2 = length MOD 250	*d2* = 0 to 249

5.3 User considerations

ECC 200 offers flexibility in the way data is encoded. Alternate character sets may be invoked using the ECI protocol. Data may be encoded in square or rectangular symbols. Where the message length exceeds the capacity of a single symbol, it is also possible to encode it in a Structured Append sequence of up to 16 separate but logically linked ECC 200 symbols (see 5.6).

5.3.1 User selection of Extended Channel Interpretation

The use of an alternative Extended Channel Interpretation to identify a particular code page or more specific data interpretation requires additional codewords to invoke the feature. The use of the Extended Channel Interpretation protocol (see 5.4) provides the capability to encode data from alphabets other than the Latin alphabet (ISO 8859-1 Latin Alphabet No. 1) supported by the default interpretation (ECI 000003).

5.3.2 User selection of symbol size and shape

ECC 200 has twenty-four square and six rectangular symbol configurations. The size and shape may be selected to suit the requirement of the application. These configurations are technically specified in 5.5.

5.4 Extended Channel Interpretation

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

a) international character sets (or code pages)

b) general purpose interpretations such as encryption and compaction

c) user defined interpretations for closed systems

d) control information for structured append in unbuffered mode.

The Extended Channel Interpretation protocol is fully specified in AIM Inc. International Technical Specification – Extended Channel Interpretations Part 1. The protocol provides a consistent method to specify particular interpretations on byte values before printing and after decoding. The Extended Channel Interpretation is identified by a 6-digit number which is encoded in the Data Matrix symbol by the ECI character followed by one to three codewords. Specific interpretations are listed in AIM Inc. Extended Channel Interpretations Character Set Register. The Extended Channel Interpretation can only be used with readers enabled to transmit the symbology identifiers. Readers that are not enabled to transmit the symbology identifier shall not transmit the data from any symbol containing an ECI. An exception can be made if the ECI(s) can be handled entirely within the reader.

The Extended Channel Interpretation protocol shall only be applied to ECC 200 symbols. A specified Extended Channel Interpretation may be invoked anywhere in the encoded message.

5.4.1 Encoding ECIs

The various encodation schemes of Data Matrix for ECC 200 (defined in Table 1) may be applied under any of the Extended Channel Interpretations. The ECI can only be invoked from ASCII encodation; once this has occurred, switching may take place between any of the encodation schemes. The encodation mode used is determined strictly by the 8-bit data values being encoded and does not depend on the Extended Channel Interpretation in force. For example, a sequence of values in the range 48 to 57 (decimal) would be most efficiently encoded in numeric mode even if they were not to be interpreted as numbers. The ECI assignment is invoked using codeword 241 (ECI character) in ASCII encodation. One, two, or three additional codewords are used to encode the ECI Assignment number. The encodation rules are defined in Table 6.

The following examples illustrate the encodation:

<u>ECI = 015000</u>

Codewords:

[241] [(15000 - 127) div 254 + 128] [(15000 - 127) mod 254 + 1]

= [241] [58 + 128] [141 + 1]

= [241] [186] [142]

<u>ECI = 090000</u>

Codewords:

[241] [(90000 - 16383) div 64516 + 192] [((90000 - 16383) div 254) mod 254 + 1] [(90000 - 16383) mod 254 + 1]

= [241] [1 + 192] [289 mod 254 + 1] [211 + 1]

= [241] [193] [36] [212]

Table 6 — Encoding ECI assignment numbers in ECC 200

ECI assignment value	Codeword sequence	Codeword values	Ranges
000000 to 000126	C_0	241	
	C_1	*ECI_no* + 1	C_1 = (1 to 127)
000127 to 016382	C_0	241	
	C_1	(*ECI_no* - 127) div 254 + 128	C_1 = (128 to 191)
	C_2	(*ECI_no* - 127) mod 254 + 1	C_2 = (1 to 254)
0016383 to 999999	C_0	241	
	C_1	(*ECI_no* - 16383) div 64516 +192	C_1 = (192 to 207)
	C_2	[(*ECI_no* - 16383) div 254] mod 254 + 1	C_2 = (1 to 254)
	C_3	(*ECI_no* - 16383) mod 254 + 1	C_3 = (1 to 254)

5.4.2 ECIs and Structured Append

ECIs may occur anywhere in the message encoded in a single or Structured Append (see 5.6) set of Data Matrix symbols. Any ECI invoked shall apply until the end of the encoded data, or until another ECI is encountered. Thus the interpretation of the ECI may straddle two or more symbols.

5.4.3 Post-decode protocol

The protocol for transmitting ECI data shall be as defined in 11.4. When using ECIs, symbology identifiers (see 11.5) shall be fully implemented and the appropriate symbology identifier transmitted as a preamble.

5.5 ECC 200 symbol attributes

5.5.1 Symbol sizes and capacity

There are 24 square symbols and 6 rectangular symbols available in ECC 200. These are as specified in Table 7.

Table 7 — ECC 200 symbol attributes

Symbol size[a]		Data region		Mapping matrix size	Total codewords		Reed-Solomon block		Inter–leaved blocks	Maximum data capacity			% of codewords used for error correction	Max. correctable codewords
Row	Col	Size	No.		Data	Error	Data	Error		Num.	Alphanum.[d]	Byte		Error/ erasure[b]
10	10	8 x 8	1	8 x 8	3	5	3	5	1	6	3	1	62,5	2/0
12	12	10 x 10	1	10 x 10	5	7	5	7	1	10	6	3	58,3	3/0
14	14	12 x 12	1	12 x 12	8	10	8	10	1	16	10	6	55,6	5/7
16	16	14 x 14	1	14 x 14	12	12	12	12	1	24	16	10	50	6/9
18	18	16 x 16	1	16 x 16	18	14	18	14	1	36	25	16	43,8	7/11
20	20	18 x 18	1	18 x 18	22	18	22	18	1	44	31	20	45	9/15
22	22	20 x 20	1	20 x 20	30	20	30	20	1	60	43	28	40	10/17
24	24	22 x 22	1	22 x 22	36	24	36	24	1	72	52	34	40	12/21
26	26	24 x 24	1	24 x 24	44	28	44	28	1	88	64	42	38,9	14/25
32	32	14 x 14	4	28 x 28	62	36	62	36	1	124	91	60	36,7	18/33
36	36	16 x 16	4	32 x 32	86	42	86	42	1	172	127	84	32,8	21/39
40	40	18 x 18	4	36 x 36	114	48	114	48	1	228	169	112	29,6	24/45
44	44	20 x 20	4	40 x 40	144	56	144	56	1	288	214	142	28	28/53
48	48	22 x 22	4	44 x 44	174	68	174	68	1	348	259	172	28,1	34/65
52	52	24 x 24	4	48 x 48	204	84	102	42	2	408	304	202	29,2	42/78
64	64	14 x 14	16	56 x 56	280	112	140	56	2	560	418	277	28,6	56/106
72	72	16 x 16	16	64 x 64	368	144	92	36	4	736	550	365	28,1	72/132
80	80	18 x 18	16	72 x 72	456	192	114	48	4	912	682	453	29,6	96/180
88	88	20 x 20	16	80 x 80	576	224	144	56	4	1 152	862	573	28	112/212
96	96	22 x 22	16	88 x 88	696	272	174	68	4	1 392	1 042	693	28,1	136/260
104	104	24 x 24	16	96 x 96	816	336	136	56	6	1 632	1 222	813	29,2	168/318
120	120	18 x 18	36	108 x 108	1 050	408	175	68	6	2 100	1 573	1 047	28	204/390
132	132	20 x 20	36	120 x 120	1 304	496	163	62	8	2 608	1 954	1 301	27,6	248/472
144	144	22 x 22	36	132 x 132	1 558	620	156	62	8[c]	3 116	2 335	1 555	28,5	310/590
							155	62	2[c]					
Rectangular Symbols														
8	18	6 x 16	1	6 x 16	5	7	5	7	1	10	6	3	58,3	3/0
8	32	6 x 14	2	6 x 28	10	11	10	11	1	20	13	8	52,4	5/0
12	26	10 x 24	1	10 x 24	16	14	16	14	1	32	22	14	46,7	7/11
12	36	10 x 16	2	10 x 32	22	18	22	18	1	44	31	20	45,0	9/15
16	36	14 x 16	2	14 x 32	32	24	32	24	1	64	46	30	42,9	12/21
16	48	14 x 22	2	14 x 44	49	28	49	28	1	98	72	47	36,4	14/25

a symbol size does not include quiet zones

b See 5.7.3

c In the largest symbol (144 x 144), the first eight Reed-Solomon blocks are 218 codewords long encoding 156 data codewords, and the last two blocks encode 217 codewords (155 data codewords). All the blocks have 62 error correction codewords.

d Based on text or C40 encoding without switching or shifting; for other encoding schemes, this value may vary depending on the mix and grouping of character sets

5.5.2 Insertion of Alignment Patterns into larger symbols

As shown in Table 7, square symbols 32 x 32 and larger and four rectangular symbols (8 x 32, 12 x 36, 16 x 36, and 16 x 48) have two or more data regions. These data regions are bounded by alignment patterns (see Annex D). The square symbols are divided into 4, 16, or 36 data regions (as illustrated in Annex D, Figures D.1, D.2, and D.3). The rectangular symbols are divided into two data regions (as illustrated in Annex D, Figure D.4). The alternating dark modules of the alignment pattern shall be to the top and right of a data region and identify the even columns and rows.

5.6 Structured Append

5.6.1 Basic principles

Up to 16 ECC 200 symbols may be appended in a structured format. If a symbol is part of a Structured Append, this is indicated by codeword 233 in the first symbol character position. This is immediately followed by three structured append codewords. The first codeword is the symbol sequence indicator. The second and third codewords are the file identification.

5.6.2 Symbol sequence indicator

This codeword indicates the position of the symbol within the set (up to 16) of ECC 200 symbols in the Structured Append format in the form *m* of *n* symbols. The first 4 bits of this codeword identify the position of the particular symbol as the binary value of (m - 1). The last 4 bits identify the total number of the symbols to be concatenated in the Structured Append format as the binary value of (17 - n). The 4-bit patterns shall conform with those defined in Table 8.

Table 8 — Structured Append symbol position bits

Symbol position	Bits 1234	Total number of symbols	Bits 5678
1	0000		
2	0001	2	1111
3	0010	3	1110
4	0011	4	1101
5	0100	5	1100
6	0101	6	1011
7	0110	7	1010
8	0111	8	1001
9	1000	9	1000
10	1001	10	0111
11	1010	11	0110
12	1011	12	0101
13	1100	13	0100
14	1101	14	0011
15	1110	15	0010
16	1111	16	0001

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

3rd position: 0010

Total 7 symbols: 1010

Bit pattern: 00101010

Codeword value: 42

5.6.3 File identification

The file identification is defined by the value of its two codewords. Each file identification codeword may have a value 1 to 254, allowing 64516 different file identifications. The purpose of the file identification is to increase the probability that only logically linked symbols are processed as part of the same message.

5.6.4 FNC1 and Structured Append

If Structured Append is used in conjunction with FNC1 (see 5.2.4.6), the first four codewords shall be used for Structured Append and the fifth and sixth codewords are available for FNC1 usage. FNC1 shall not be repeated in these positions in the second and subsequent symbols, except when used as a field separator.

5.6.5 Buffered and unbuffered operation

The message within a Structured Append sequence can be buffered in the reader in its entirety and transmitted after all of the symbols have been read. Alternatively, the reader may transmit the decoded data in each symbol as it is read. In this unbuffered operation, the ECI protocol for structured append (specified in AIM ITS 04/001, Part 1) defines a control block that shall be prefixed to the beginning of the data transmitted for each symbol.

5.7 Error detection and correction

5.7.1 Reed-Solomon error correction

ECC 200 symbols employ Reed-Solomon error correction. For ECC 200 symbols with less than 255 total codewords, the error correction codewords are calculated from data codewords with no interleaving. For ECC 200 symbols with more than 255 total codewords, the error correction codewords are calculated from data codewords with the interleaving procedure described in Annex A. Each ECC 200 symbol has a specific number of data and error correction codewords which are divided into a specific number of blocks, as defined in Table 7, and to which the interleaving procedure defined in Annex A is applied.

The polynomial arithmetic for ECC 200 shall be calculated using bit-wise modulo 2 arithmetic and byte-wise modulo 100101101 (decimal 301) arithmetic. This is a Galois field of 2^8 with 100101101 representing the field's prime modulus polynomial: $x^8 + x^5 + x^3 + x^2 + 1$. Sixteen different generator polynomials are used for generating the appropriate error correction codewords. These are given in E.1.

5.7.2 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 codes (see E.1).

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

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. This can be implemented by using the division circuit as shown in Figure 5. The registers b_0 through b_{k-1} are initialised as zeros. There are two phases to generate the encoding. In the first phase, with the switch in the down position the data codewords are passed both to the output and the circuit. The first phase is complete after *n* clock pulses. In the second phase ($n + 1$ ... $n + k$ clock pulses), with the switch in the up position, the error correction codewords $\varepsilon_{k-1}, \ldots, \varepsilon_0$ are generated by flushing the registers in order 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. (See Annex A).

Note: n and k are defined in 3.2 as the number of data codewords and the number of error correction codewords respectively.

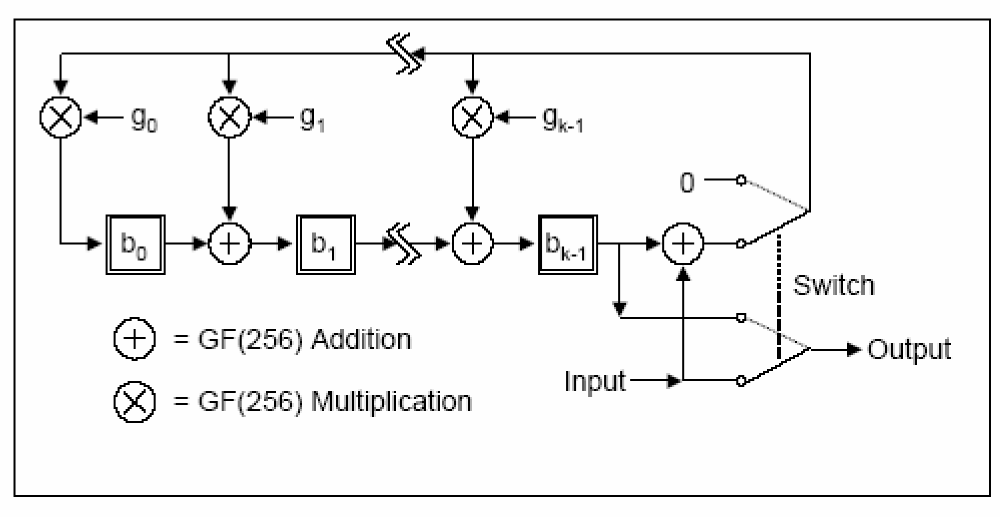


Figure 5 — Error correction codeword encoding circuit

5.7.3 Error correction capacity

The error correction codewords can correct two types of erroneous codewords: erasures (erroneous codewords at known locations) and errors (erroneous codewords at unknown locations). An erasure is an unscanned or undecodable symbol character. An error is a misdecoded symbol character. The number of erasures and errors that can be corrected is given by the following formula:

$e + 2t \leq d - p$

where:

e = number of erasures

t = number of errors

d = number of error correction codewords

p = number of codewords reserved for error detection.

In the general case, $p = 0$. However, if most of the error correction capacity is used to correct erasures, then the possibility of an undetected error is increased. Whenever the number of erasures is more than half the number of error correction codewords, $p = 3$. For small symbols (10 x 10, 12 x 12, 8 x 18, and 8 x 32), erasure correction should not be used ($e = 0$ and $p = 1$).

5.8 Symbol construction

Given the codeword sequence obtained in the previous sections, an ECC 200 symbol is constructed using the following steps:

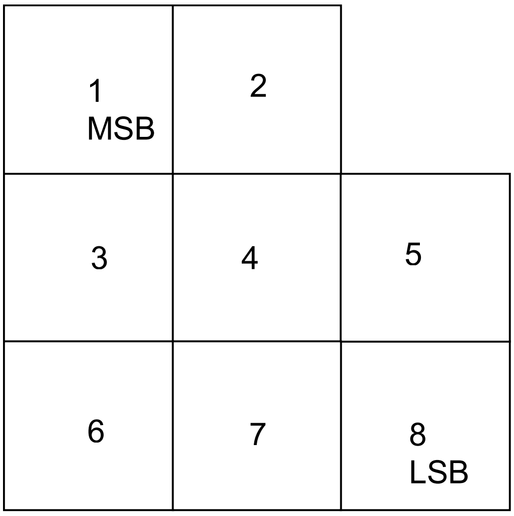
a) Place codeword modules in a mapping matrix

b) Insert alignment pattern modules, if any

c) Place finder modules along the perimeter

5.8.1 Symbol character placement

Each symbol character shall be represented by eight modules which are nominally square in shape; each module represents a binary bit. A dark module is a one and a light module is a zero. The eight modules are in order from left to right and top to bottom to form a symbol character as shown in Figure 6. Because the symbol character shape defined in Figure 6 cannot be perfectly nested at the symbol boundary, some symbol characters are split into portions. Symbol character placement is defined in the C language program in F.1, described in F.2 and illustrated in F.3.



LSB = Least significant bit
MSB = Most significant bit

Figure 6 — Representation of a codeword in a symbol character for ECC 200

5.8.2 Alignment Pattern module placement

This step is only needed for larger matrices: square: 32 x 32 and larger rectangular: 8 x 32, 12 x 36 and larger The mapping matrix is sub-divided into data regions, of the sizes defined in Table 7, for the chosen symbol format. The data regions are separated from each other by two-module-wide alignment patterns. This will result in some of the symbol characters being split between two adjacent data regions. For square matrices, the alignment patterns are placed between the data regions horizontally and vertically in pairs with a total alignment pattern count of 2, 6, or 10 as shown in Figures D.1 - D.3. For rectangular matrices, only a single vertical alignment pattern is placed between the data regions as shown in Figure D.4.

5.8.3 Finder Pattern module placement

Modules are placed along the perimeter of the matrix to construct the finder pattern as described in Section 4.3.1.

6 ECC 000 - 140 requirements

6.1 Use recommendations

For new applications or open systems ECC 200 is recommended (See Clause 5). There is no known application where ECC 200 will be more likely to succumb to symbol damage than ECC 000 to 140 for a given symbol size.

6.2 Encode procedure overview

This section provides an overview of the encoding procedure. Following sections will provide more details. An example encode for ECC 050 is given in Annex Q. The following steps convert user data to an ECC 000 - 140 symbol:

Step 1: Data encodation

The user data is analysed to identify the variety of different characters to be encoded. For maximum compaction efficiency, the lowest level encodation scheme capable of encoding the data should be selected. If the user does not specify the matrix size, then choose the smallest size that accommodates the data. The result of this step is called the Encoded Data Bit Stream.

Step 2: Data prefix construction

A Data Prefix Bit Stream is constructed from the Format ID, CRC, and Data Length bit fields. This Data Prefix Bit Stream is prefixed to the Encoded Data Bit Stream to produce the Unprotected Bit Stream.

Step 3: Error checking and correction

The Unprotected Bit Stream is processed by the user specified convolutional coding encode algorithm to produce the Protected Bit Stream. This step is omitted for ECC 000.

Step 4: Header and trailer construction

A header containing only the ECC bit field is prefixed to the Protected Bit Stream. A trailer containing pad bits (zeros) is appended to the Protected Bit Stream. The Protected Bit Stream with the header and trailer added is called the Unrandomised Bit Stream.

Step 5: Pattern randomising

The Unrandomised Bit Stream is processed by the pattern randomising algorithm and produces the Randomised Bit Stream.

Step 6: Module placement in matrix

Modules are placed in a matrix to construct the finder pattern. The Randomised Bit Stream is placed into the matrix one module at a time according to the data module placement algorithm given in Annex H. Figure 7 shows the various bit streams during the encode process.

6.3 Data encodation

The data shall be encoded using one of six encodation schemes (see Table 9). The encodation scheme is fixed for the entire symbol, and thus the selection of the most appropriate encodation scheme can have a considerable effect on the number of bits required to encode any given data. The same data may be represented in ECC 000 - 140 symbols in different ways through the use of the different encodation schemes. The character sets of all the encodation schemes, except the 8-bit byte scheme, are given in Annex I. The 8-bit byte scheme is user definable. The most efficient scheme to use is the lowest base number scheme which is capable of encoding all the characters in the message. Thus if all the characters can be encoded in Base 27, it is not efficient to use Base 37, Base 41 or ASCII.

Table 9 — Encodation schemes

Encodation scheme	Characters	Bits per data character
Base 11	Numeric data	3,5
Base 27	Upper-case alphabetic	4,8
Base 37	Upper-case alphanumeric	5,25
Base 41	Upper-case alphanumeric and punctuation	5,5
ASCII	Full 128 ASCII set	7
8-bit Byte	User defined	8

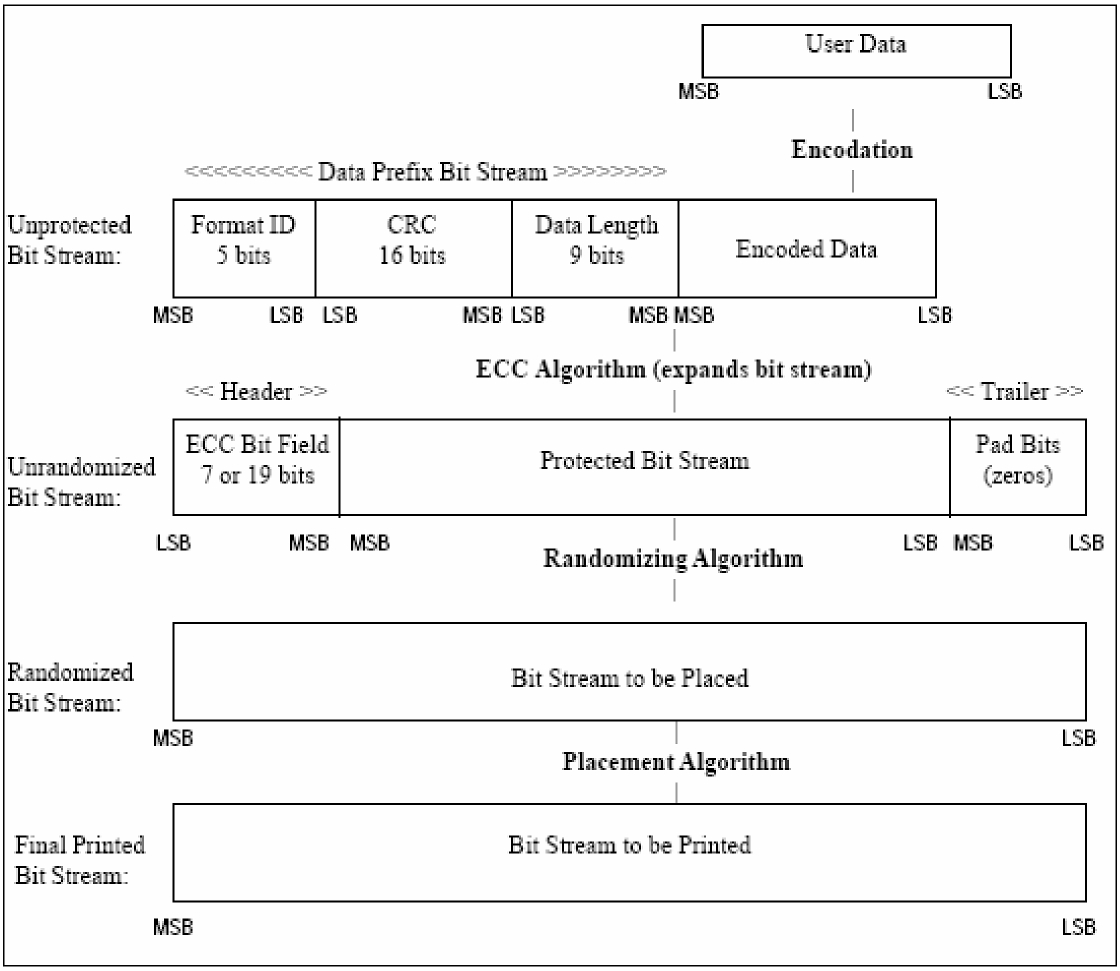


Figure 7 — ECC 000-140 encode process bit streams

To determine the appropriate encodation scheme, the data to be encoded should be analysed. The character sets of each of the Base *N* encodation schemes should be compared with the data character set to be encoded starting with the Base 11 character set. If this is suitable then it should be used, if not, the comparisons should continue with Base 27, Base 37 and Base 41, until the appropriate lowest level encodation scheme is found. If data characters beyond the capability of Base 41 need to be encoded, the ASCII set should be used, unless characters are beyond this; in which case the 8-bit byte set should be used.

For all encodation schemes, each compressed sequence of 4 to 24 bits is placed into the Encoded Bit Stream in reverse order (LSB first). This means that each individual compressed sequence is composed, then reversed, and output immediately to the Encoded Bit Stream. This does not mean that a complete compressed bit stream is formed, then reversed.

The details of each encodation scheme are given in the following clauses.

6.3.1 Base 11 - Numeric encodation

The Base 11 (Numeric) encodation scheme encodes 6 data characters as 21 bits, achieving an encodation density of 3,5 bits per data character. The Base 11 code set enables the following 11 characters to be encoded:

0 to 9

space

The data is encoded in two stages. In the first stage, the actual data characters shall be replaced by their Base 11 code values, as given in Annex I. In the second phase, the Base 11 code values shall be compacted using a Base 11 to Base 2 conversion according to the procedures defined in I.1.

6.3.2 Base 27 - Upper-case Alphabetic encodation

The Base 27 (Upper-case Alphabetic) encodation scheme encodes 5 data characters as 24 bits, achieving an encodation density of 4,8 bits per data character. The Base 27 code set enables the following 27 characters to be encoded:

A to Z

space

The data is encoded in two stages. In the first stage, the actual data characters shall be replaced by their Base 27 code values, as given in Annex I. In the second stage, the Base 27 code values shall be compacted using a Base 27 to Base 2 conversion according to the procedures defined in I.2.

6.3.3 Base 37 - Upper-case Alphanumeric encodation

The Base 37 (Upper-case Alphanumeric) encodation scheme encodes 4 data characters as 21 bits, achieving an encodation density of 5,25 bits per data character. The Base 37 code set enables the following 37 characters to be encoded:

A to Z

0 to 9

space

The data is encoded in two stages. In the first stage, the actual data characters shall be replaced by their Base 37 code values, as given in Annex I. In the second stage, the Base 37 code values shall be compacted using a Base 37 to Base 2 conversion according to the procedures defined in I.3.

6.3.4 Base 41 - Upper-case Alphanumeric plus Punctuation encodation

The Base 41 (Upper-case Alphanumeric plus Punctuation) encodation scheme encodes 4 data characters as 22 bits, achieving an encodation density of 5,5 bits per data character. The Base 41 code set enables the following 41 characters to be encoded:

A to Z

0 to 9

space

. (period)

, (comma)

- (minus or hyphen)

/ (forward slash or solidus)

The data is encoded in two stages. In the first stage, the actual data characters shall be replaced by their Base 41 code values, as given in Annex I. In the second stage, the Base 41 code values shall be compacted using a Base 41 to Base 2 conversion according to the procedures defined in I.4.

6.3.5 ASCII encodation

The ASCII encodation scheme enables all 128 characters from ISO/IEC 646 to be encoded. Each data character shall be encoded as a 7-bit byte equivalent to the decimal value shown in the ASCII column of Table I.1 of Annex I.

6.3.6 8-bit byte encodation

The 8-bit byte encodation scheme shall be used for closed applications, where the data interpretation shall be determined by the user. Each data character shall be encoded as an 8-bit byte.

6.4 User selection of error correction level

6.4.1 Selection of error correction level

ECC 000 - 140 symbols offer five levels of error correction using convolutional code error correction, as set out in Table 10. In an application, it is important to understand that these error correction levels result in the generation of a proportional increase in the number of bits in the message (and hence increase in the size of the symbol), and offer different levels of error recovery.

Table 10 — Error correction, error recovery and overhead percentages

Error correction code level	Maximum % damage	% increase in user bits from ECC 000
000	none	none
050	2,8	33
080	5,5	50
100	12,6	100
140	25	300

6.4.2 Other error correction levels based on convolutional code algorithms

Other levels of error correction, based on convolutional code algorithms, have been used in Data Matrix applications implemented prior to the publication of this International Standard. Information on these non-standard Error Correction levels is available from AIM Inc. Such symbols do not conform with this International Standard.

6.5 Constructing the Unprotected Bit Stream

Figure 7 illustrates that the Unprotected Bit Stream has the Data Prefix Bit Stream as a prefix to the encoded data bits. The component parts of the Data Prefix Bit Stream are defined below.

6.5.1 Format ID Bit Field

The format ID defines the data encodation scheme. The format ID has a decimal value for the purposes of definition and a 5-bit segment value for encoding as defined in Table 11.

Table 11 — Encoding the Format ID

Format ID	Encodation scheme	Binary segment value
		MSB LSB
1	Base 11	00000
2	Base 27	00001
3	Base 41	00010
4	Base 37	00011
5	ASCII	00100
6	8-bit Byte	00101

6.5.2 CRC Bit Field

The CRC Bit Field is generated by the CRC algorithm. The CRC Value is generated from the original user data as 8-bit bytes before encodation and so produces an independent error check on the user data. Annex J describes the complete procedure for generating the CRC Value.

6.5.3 Data Length Bit Field

The Data Length Bit Field is 9 bits in length and represents, as a binary value, the number of user data characters being encoded.

6.5.4 Data prefix construction

The Data Prefix Bit Stream is constructed as 30 bits as illustrated in Figure 8.

Format ID 5 bits	CRC Value 16 bits	Data Length 9 bits
MSB LSB	LSB MSB	LSB MSB

Figure 8 — Structure of Data Prefix Bit Stream

NOTE Some bit fields start with the MSB, others start with the LSB.

6.5.5 Completing the Unprotected Bit Stream

The encoded data bits are added as a suffix to the Data Prefix Bit Stream to construct the Unprotected Bit Stream.

6.6 Constructing the Unrandomised Bit Stream

Figure 7 illustrates that the Unrandomised Bit Stream has three constituent parts:

a) Header

b) Protected Bit Stream

c) Trailer

The component parts shall be generated as defined below.

6.6.1 Header construction

The header of the Unrandomised Bit Stream contains the ECC Bit Field, which identifies the convolutional code structure used to protect the data encoded in the symbol. The ECC Bit Field is 7 or 19 bits long and the values are shown in Table 12.

Table 12 — ECC Bit Field

ECC Level	Binary Segment Identifier MSB LSB
000	1111110
050	0001110000000001110
080	1110001110000001110
100	1111111110000001110
140	1111110001110001110

6.6.2 Applying convolutional coding to create the Protected Bit Stream

One of the five error correction levels shall be applied. The selection criteria are defined in Section 6.4. No error correction is applied for ECC 000, so the Unprotected Bit Stream becomes the Protected Bit Stream. For the other four error correction levels, convolutional coding is applied. This expands the user data proportionally throughout its length. The encoded bit stream shall be created by processing the unprotected bit stream through the appropriate error correction state machine and reading the results. The circuit diagrams of the four state machines for ECC 050 to 140 are given in Annex K.

6.6.3 Trailer construction

A Trailer containing pad bits (zeros) is appended to the Protected Bit Stream. Pad bits shall be added at the end of the bit stream to ensure that the square root of the total number of bits in the Unrandomised Bit Stream shall be an odd integer between 7 and 47. This ensures that the symbol is square.

6.6.4 Completing the Unrandomised Bit Stream

The Protected Bit Stream, with the header and trailer added, is called the Unrandomised Bit Stream and is shown in Figure 7.

6.7 Pattern randomising

The Unrandomised Bit Stream is processed by the pattern randomising algorithm and produces the Randomised Bit Stream. The pattern randomising algorithm consists of a bitwise XOR operation between the Unrandomised Bit Stream and the Master Random Bit Stream as given in Annex L starting with the MSB position and continuing for the length of the Unrandomised Bit Stream.

6.8 Module placement in matrix

The size of the sides of the data module grid is given by the odd integer square root (between 7 and 47) calculated in the procedure defined in 6.6.3. The Randomised Bit Stream is placed into the matrix one module at a time according to the data module placement grids given in Annex H. The finder pattern (as defined in 4.3.1) shall be placed to produce an external border to the data module grid.

7 Symbol dimensions

7.1 Dimensions

Data Matrix symbols shall conform to the following dimensions:

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

finder pattern: the width of the finder pattern shall equal *X.*

alignment pattern: the width of the alignment pattern shall equal 2*X.*

Quiet zone: The minimum quiet zone is equal to *X* on all four sides. For applications with moderate to excessive reflected noise in close proximity to the symbol, a Quiet Zone of 2*X* to 4*X* is recommended

8 Symbol quality

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

Some marking technologies may not be able to produce symbols conforming to this specification without taking special precautions. Annex T gives additional guidance to help any printing system achieve valid Data Matrix symbols.

8.1 Symbol quality parameters

8.1.1 Fixed pattern damage

Annex M defines the measurement and grading basis for Fixed Pattern Damage.

NOTE As provided for in Annex A of ISO/IEC 15415, the measurements and values defined in Annex M of this International Standard override those indicated in Annex A of ISO/IEC 15415.

8.1.2 Scan grade and overall symbol grade

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

8.1.3 Grid non-uniformity

The ideal grid is calculated by using the four corner points of the sampling grid for each data region and subdividing it equally in both axes.

8.1.4 Decode

The reference decode algorithm specified in this international standard shall be applied to determine the grade for Decode. A failure of the reference decode algorithm to successfully decode the symbol shall result in a grade of 0 for decode.

8.2 Process control measurements

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

9 Reference decode algorithm for Data Matrix

This reference decode algorithm finds a Data Matrix symbol in an image and decodes it.

a) Define measurement parameters and form a digitised image:

1) Define a distance d_{min} which is 7,5 times the aperture diameter defined by the application. This will be the minimum length of the "L" pattern's side.

2) Define a distance g_{max} which is 7,5 times the aperture diameter. This is the largest gap in the "L" finder that will be tolerated by the finder algorithm in step b).

3) Define a distance m_{min} which is 1,25 times the aperture diameter. This is the nominal minimum module size.

4) Form a black/white image using a threshold determined according to the method defined in ISO/IEC 15415.

b) Search horizontal and vertical scan lines for the two outside edges of the Data Matrix "L":

1) Extend a scan line horizontally in both directions from the centre point of the image. Sample along the scan line. For each white/black or black/white transition found along the scan line resolved to the pixel boundary:

i) Follow the edge upward sampling pixel by pixel until either it reaches a point 3,5m_{min} distant from the intersection of the scan line and the edge starting point, or the edge turns back toward the intersection of the scan line and the edge - the starting point.

ii) Follow the edge downward pixel by pixel until either it reaches a point 3,5m_{min} distant from the intersection of the scan line and the edge starting point, or the edge turns back toward the intersection of the scan line and the edge - the starting point.

iii) If the upward edge reaches a point 3,5m_{min} from the starting point

I) Plot a line A connecting the end points of the upward edge.

II) Test whether the intermediate edge points lie within 0,5m_{min} from line A and the edge point is farther from the starting point than the previous edge point. If so, continue to step iii. Otherwise proceed to step 1)iv) to follow the edge in the opposite direction.

III) Continue following the edge upward until the edge departs 0,5m_{min} from line A. Back up to the closest edge point greater than or equal to m_{min} from the last edge point along the edge before the departing point and save this as the edge end point. This edge point should be along the "L" candidate outside edge.

IV) Continue following the edge downward until the edge departs 0,5m_{min} from line A. Back up to the closest edge point greater than or equal to m_{min} from the last edge point along the edge before the departing point and save this as the edge end point. This edge point should be along the "L" candidate outside edge.

V) Calculate a new adjusted line A1 that is a "best fit" line to the edge in the two previous steps. The "best fit" line uses the linear regression algorithm (using the end points to select the proper dependent axis, i.e. if closer to horizontal, the dependent axis is x) applied to each point. The "best fit" line terminates lines at points p1 and p2 that are the points on the "best fit" line closest to the endpoints of the edge.

VI) Save the line A1 segment two end points, p1 and p2. Also save the colour of the left side of the edge viewed from p1 to p2.

iv) If step iii) failed or did not extend downward by 3,5m_{min} in step iii) IV), test if the downward edge reaches a point 3,5m_{min} from the starting point. If so, repeat the steps in iii) but with the downward edge.

v) If neither steps iii) or iv) were successful, test if both the upward and downward edges terminated at least 2m_{min} from the starting point. If so, form an edge comprised of the appended 2m_{min} length upward and downward edge segments and repeat the steps in iii) but with the appended edge.

vi) Proceed to and process the next transitions on the scan line, repeating from step i), until the edge of the image is reached.

2) Extend a scan line vertically in both directions from the centre point of the image. Look for line segments using the same logic in step 1) above but following each edge transition first left and then right.

3) Search among the saved line A1 segments for pairs of line segments that meet the following four criteria:

   i) Verify that the closest endpoints of the two line segments are less than g_{max} from each other.

   ii) Verify that the two lines are co-linear within 5 degrees.

   iii) Verify that the two lines have the same colour if their p1 to p2 directions are the same or that the colours are opposite if their p1 to p2 directions are opposite to each other.

   iv) Form two temporary lines by extending each line to reach to the point on the extension that is closest to the furthest end point of the other line segment. Verify that the two extended lines are separated by less than $0,5m_{min}$ at any point between the two extended lines.

4) For each pair of lines meeting the criteria of step 3) above, replace the pair of line segments with a longer A1 line segment that is a "best fit" line to the four end points of the pair of shorter line segments. Also save the colour of the left side of the edge of the new longer line viewed from its p1 endpoint to its p2 endpoint.

5) Repeat steps 3) and 4) until no more A1 line pairs can be combined.

6) Select line segments that are at least as long as d_{min}. Flag them as "L" side candidates.

7) Look for pairs of "L" side candidates that meet the following three criteria:

   i) Verify that the closest points on each line are separated by less than $1,5g_{max}$.

   ii) Verify that they are perpendicular within 5 degrees.

   iii) Verify that the same colour is on the inside of the "L" formed by the two lines. Note that if one or both lines extend past their intersection, then the two or four "L" patterns formed will need to be tested for matching colour and maintaining a minimum length of d_{min} for the truncated side or sides before they can become "L" candidates.

8) For each candidate "L" pair found in step 7) form an "L" candidate by extending the segments to their intersection point.

9) If the "L" candidate was formed from line segments with the colour white on the inside of the "L", form a colour inverted image to decode. Attempt to decode the symbol starting with the appropriate normal or inverted image starting from step D below using each of the "L" candidates from step 8) as the "L" shaped finder. If none decode, proceed to step c).

c) Maintain the line A1 line segments and "L" side candidates from the previous steps. Continue searching for "L" candidates using horizontal and vertical scan lines offset from previous scan lines:

   1) Using a new horizontal scan line $3m_{min}$ above the centre horizontal scan line, repeat the process in step b) 1), except starting from the offset from the centre point, and then b)3) through b)9). If there is no decode, proceed to the next step.

   2) Using a new vertical scan line $3m_{min}$ left of the centre vertical scan line, repeat the process in step b)2), except starting from the offset from the centre point, and then steps b)3) through b)9). If there is no decode, proceed to the next step 3).

   3) Repeat step 1) above except using a new horizontal scan line $3m_{min}$ below the centre horizontal scan line. If there is no decode, repeat step 2) above except using a new vertical scan line $3m_{min}$ right of the centre vertical scan line. If there is no decode, proceed to step 4) below.

4) Continue processing horizontal and vertical scan lines as in steps 1) through 3) that are $3m_{min}$ above, then left, then below, then right of the previously processed scan lines until either a symbol is decoded or the edge of the image is reached.

d) First assume that the candidate area contains a square symbol. If the area fails to decode as a square symbol, then try to find and decode a rectangular symbol starting from procedure j). For a square symbol, first plot a normalised graph of transitions for the equal sides of the candidate area in order to find the alternating module finder pattern:

1) Project a line through the candidate area bisecting the interior angle of the two sides of the "L" found above as shown in figure 9. Define the two equal areas formed by the bisecting line as the right side and the left side as viewed from the corner of the "L".

2) For each side, form a line called a “search line” between a point d_{min} distance from the corner along the “L” line, parallel to the other “L” side line, and extending to the bisecting line as shown in Figure 9.

3) Move each search line away from the corner of the “L” as shown in Figure 9, lengthening each line as it expands to span its two bounding lines, the “L” line and the bisecting line. Keep each search line parallel to the other “L” side line. As each side is moved by an image pixel, plot the sum of number of black/white and white/black transitions multiplied by the length of the longest “L” side divided by the current length of the search line measured between the two bounding lines:

T = (number of transitions) (“L” max. line length) / (search line length).

This formula normalises T to keep it from increasing because the line lengthens.

Continue to calculate the T values until the search line is longer than the longest axis of the candidate area plus 50%.

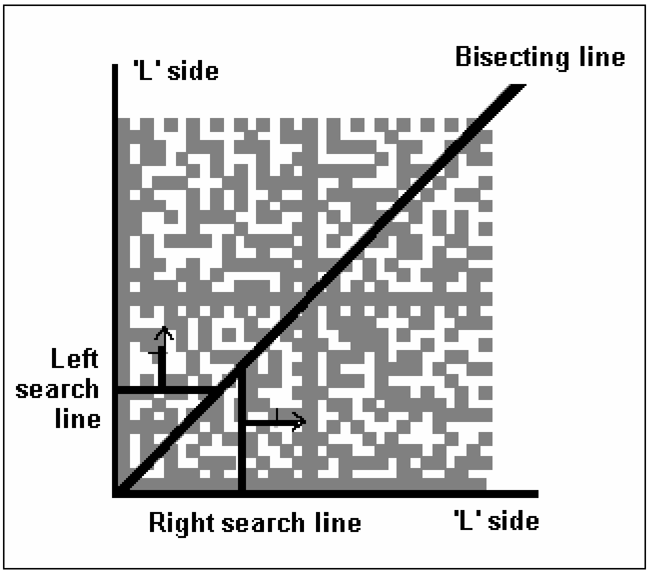


Figure 9 — Expanding search lines

4) Form a plot of the T values for each side, where the Y-axis is the T value and the X-axis is the search line’s distance from the corner of the “L”. A sample plot is shown in Figure 10.

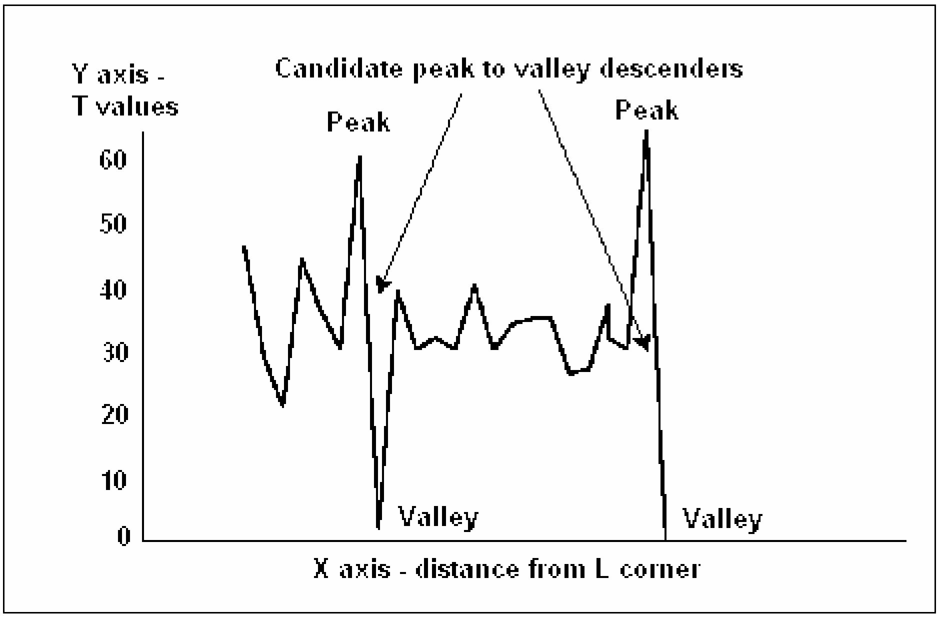


Figure 10 — Example plot of T as the search line expands

5) Starting from the *T* value with the smallest *X* in the right side's plot and then increasing *X*, find the first instance of a descending line where the T_S value (T_S = maximum of zero and *T* - 1) at the valley is less than 15% of the peak's *T* value. If the peak or valley in the plot has a flat plateau or floor, select the peak or valley point closest to the descending line in the plot. The search line at the peak may correspond to an alternating finder pattern side. At the valley, the search line may correspond to the solid dark interior line or a light quiet zone.

6) Find a peak and valley in the left side's plot which most nearly matches the right peak and valley *X* values. If either of the selected left side peak or valley *X* values differ by more than 15% from the equivalent right side peak or valley *X* values, discard the right side peak and valley and continue searching from step d) 5) for the next peak and valley. The 15% specifies the maximum allowed foreshortening.

7) The right side's valley search line, the left side's valley search line, and the two sides of the "L" outline a possible symbol's data region. Process the data region according to step E. If the decode fails, discard the right side peak and valley and continue searching from step d) 5) for the next peak and valley.

e) For each of the two sides of the alternating pattern, find the line passing through the centre of the alternating light and dark modules:

1) For each side, form a rectangular region bounded by the side's peak and valley search lines as the longer two sides of the rectangle, and the "L" side and the other side's valley search line as the shorter two sides, as shown in Figure 11.

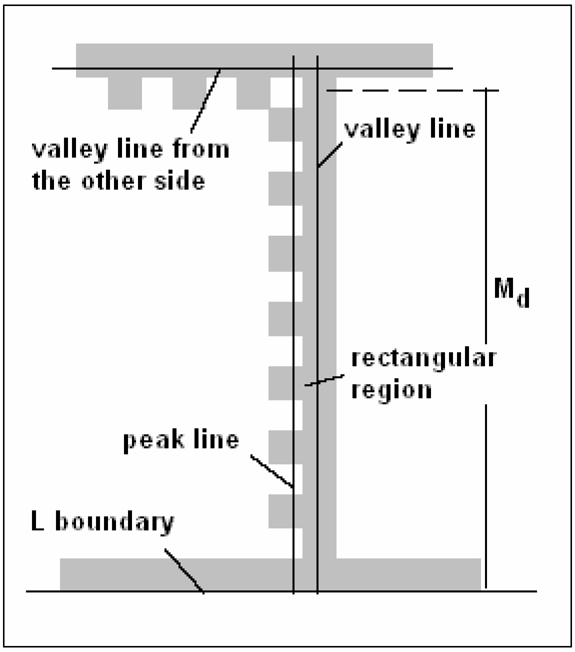


Figure 11 — Rectangular region construction

2) Within the rectangular region, find pixel edge pairs on the outside boundary of teeth:

   i) Traverse test lines starting with and parallel to the minimum line looking for transitions to the opposite colour normally orthogonal to the test line. Select only transitions that are either dark to light or light to dark where the first colour matches the predominate colour of the image along the valley line.

   ii) If the number of transitions found is less than 15% of the number of pixels comprising the valley line, and the test line is not the peak line, move the test line toward the peak line by approximately one pixel and repeat step a. If the 15% criterion is met or the peak line is reached continue to the next step.

   iii) Calculate a preliminary "best fit line" with linear regression using the points on the edge between the selected pixel pairs.

   iv) Discard the 25% of the points which are furthest from the preliminary "best fit line". Calculate a final "best fit line" with linear regression using the remaining 75% of points. This line should pass along the outside of the alternating pattern, shown as the "best fit line" in Figure 12.

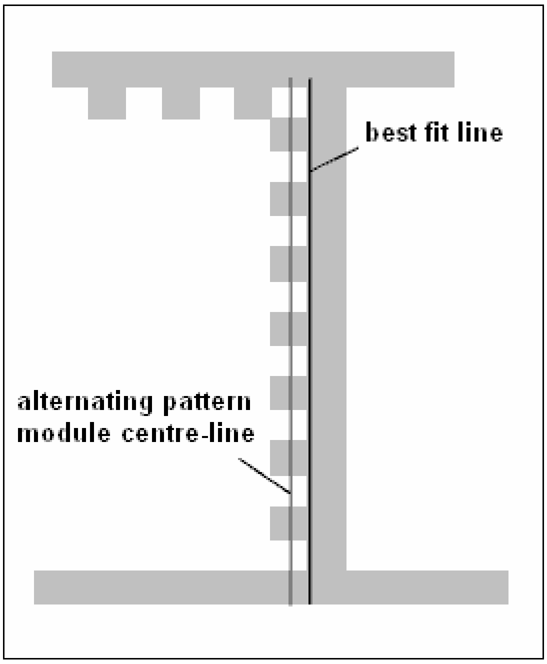


Figure 12 — Alternating pattern module centre-line

3) For each side, construct a line parallel to the step e) 2) line which is offset toward the “L” corner by the length of the peak search line divided by twice the number of transitions in the peak search line:

Offset = length of peak line / (number of transitions * 2)

Each of the two constructed lines should correspond to the centre-line or midline of the alternating module pattern on that side, see Figure 13.

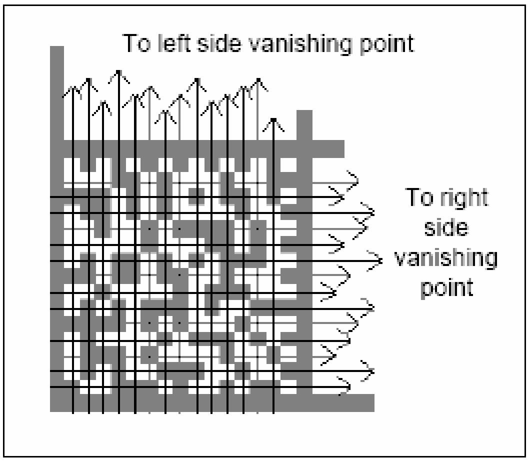


Figure 13 — Module sampling grid construction

f) For each side, determine the number of data modules in the side of the square symbol or data region:

1) Bound the alternating pattern mid-line constructed in step e)3) by the adjacent “L” line and the other alternating pattern mid-line from step e)3). Call the length of this line M_d (see Figure 11).

2) Along the bounded mid-line, measure the edge-to-edge distances between all the similar edges of all two-element pairs, i.e. dark/light and light/dark element pairs.

3) Select the median edge-to-edge measurement and set the current edge-to-edge measurement estimate, *EE_Dist*, to the median measurement.

4) Discard all element pairs with edge-to-edge measurements that differ more than 25% from *EE_Dist*.

5) Calculate the average of the remaining measurements for the side. Call the average E_{avg}.

6) The calculated number of data modules "*dm*" is defined by the formula:

$$dm = ((M_d * 2) / E_{avg}) - 1{,}5$$

where "*dm*" is rounded to the nearest integer value.

7) If "*dm*" differs for the two sides, discard the right side peak and valley and continue searching from step d)5) for the next peak and valley. Otherwise, *dm* is the size of the square data region.

g) For each side, find the centre points of the alternating pattern modules:

1) Using the remaining element pair measurements from f) 4), calculate the average ink spread (vertical or horizontal depending on the segment side) by the average of the element pair's ink spread, where *bar* is the dark element width and *space* is the light element width in a remaining element pair:

ink_spread = Average ((*bar* - ((*bar* + *space*) / 2)) / ((*bar* + *space*) / 2))

2) Calculate the centre of the bar in the median element pair using the following offset into the bar from the outside edge of the bar in the median pair:

offset = (*EE_Dist* * (1 + *ink_spread*)) / 4

3) Starting from the centre of the bar in the median element pair from step f).3), and proceeding in the direction of the space in the element pair, until reaching the end of the bounded mid-line, calculate each element's centre, shown by the speckled pattern in Figure 14, by the following steps:

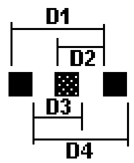


Figure 14 — Edge-to-edge measurements for finding an element centre

(While three bars and two spaces are shown in Figure 14, if a space is the element for which the centre is to be calculated, then the diagram would have three spaces instead of the bars and two bars instead of the spaces. For light elements adjacent to the element at the end of the mid-line, either *D1* or *D4* measurements are omitted as they would fall outside the symbol's or segment's measurable element boundaries.)

i) Calculate a point p1 along the mid-line which is *EE_Dist*/2 from the previously calculated element centre in the direction of the new element.

ii) Calculate d_1 through d_4 where:

$d_1 = D1 / 2$

$d_2 = D2$

$d_3 = D3$

$d_4 = D4 / 2$

iii) If one of the values d_1 through d_4 is within 25% of *EE_Dist*, select the one which is closest to *EE_Dist*, and set the new *EE_Dist* to be the average of the current *EE_Dist* and the selected d_1 through d_4 distance.

I) If d_1 or d_4 are selected, select the corresponding *D1* or *D4* edge closest to the element the centre of which is to be calculated. Offset this edge by *(ink_spread/2) * (EE_Dist/2)* in the appropriate direction (i.e., if *ink_spread* is positive, the offset will move the edge toward the space included in the distance D1 or D4 and if negative, the offset will move away from this space). Calculate a point p2 along the mid-line which is 0,75 times the selected d_1 or d_4 value from the offset edge and toward the element centre to be calculated.

II) If d_2 or d_3 are selected, select the corresponding *D2* or *D3* edge closest to the element the centre of which is to be calculated. Offset this edge by *(ink_spread/2) * (EE_Dist/2)*in the appropriate direction (i.e., if *ink_spread* is positive, the offset will move the edge toward the space included in the distance D2 or D3 and if negative, the offset will move away from this space). Calculate a point p2 along the mid-line which is 0,25 times the selected d_2 or d_3 value from the offset edge and toward the element centre to be calculated.

III) Set the element's centre as halfway between p1 and p2.

iv) Otherwise if none of the values d_1 through d_4 is within 25% of *EE_Dist*, leave *EE_Dist* at its current value, use p1 as the new element's centre, and proceed to the next element.

4) Starting from the bar in the median element pair, and proceeding in the opposite direction from step 3), until reaching the other end of the bounded mid-line, calculate each element's centre, following the procedures in step 3).

h) Plot the data module sampling grid in the data region by extending the alternating pattern module centres:

1) Extend each side's step e)3) midline and the opposite side's "L" line to form the vanishing point of the two nearly parallel or parallel extended lines.

2) Extend rays from each vanishing point passing through the step G module centres of the nearly perpendicular step e)3) line.

3) The intersection of the two sets of nearly perpendicular rays should correspond to the centres of the data modules in the data region, as shown in Figure 13.

i) Continue to fill in the remaining data regions.

1) When a data region is processed, form a new "L" for the next data section to the "left" or "above" using one of two processes:

i) a. If the new data region is still bounded on one side by the original "L" from procedure B, repeat from procedure C to process the new data region using the selected set of points from step e)2) and the set of points on the "L" from step b)2) which lie beyond the step e)2) line.

ii) b. If the new data region is bounded on two sides by data regions, repeat from procedure c) to process the new data region using the selected set of points from step e)2) for each data region which are adjacent and bound the new region on two sides

2) If a data region does not match the number of modules in previously processed regions trim the symbol to the largest number of regions which correspond to a legal symbol.

3) Decode the symbol with its one or more data regions starting with procedure k).

j) Find the data sections of a rectangular symbol.

1) For each side of the "L" move a line perpendicular to the side and scanning along the length of the other side of the "L". As each side is moved by a pixel, plot the sum of number of black/white and white/black transitions:

T = (number of transitions) ("L" max line length) / (search line length).

Continue until the parallel line moves further than the perpendicular leg of the "L" plus 10%.

2) Starting from the origin of the plot, for each direction, find the first instance of a descending line where the *T* value at the valley is less than 15% of the peak's value. If the peak or valley in the plot has a flat plateau or floor, select the peak or valley point closest to the descending line in the plot. The valley line at this point may form a side of a symbol or data region.

3) Find the alternating pattern lines for each side of the region similar to procedure e).

4) Plot the module sample grid in the data region or symbol as in procedures f), g), and h). Skip step f) 6) which requires that the region is square.

5) If the data region defined is not a valid rectangular symbol, try to form a new data region using further valid peak to valley plot transitions.

6) Process any additional regions as in procedure i).

7) If a valid data region or two regions are detected, attempt to decode the symbol as in procedures k) and l). If the region(s) were not valid or the decode fails, disregard the candidate area.

k) If the number of data modules is even or the symbol forms a valid rectangular symbol, decode the symbol using Reed-Solomon error correction:

1) Sample the data modules at their predicted centres. Black at the centre is a one and white is a zero.

2) Convert the eight module samples in the defined codeword patterns into 8-bit symbol character values.

3) Apply Reed-Solomon error correction to the symbol character values.

4) Decode the symbol characters into data characters according to the specified encodation schemes.

l) Otherwise the number of data modules is odd, so decode the symbol using convolution code error correction:

1) Sample the data modules at their predicted centres. Black at the centre is a one and white is a zero.

2) Apply the black/white balancing mask.

3) Use the bit ordering table to convert the data into a bit stream.

4) Apply the appropriate convolution code error correction.

5) Convert the bit stream to data characters according to the encodation scheme specified.

6) Verify that the CRC is correct.

10 User guidelines

10.1 Human readable interpretation

Because Data Matrix symbols are capable of encoding thousands of characters, a human readable interpretation of the data characters may not be practical. As an alternative, descriptive text rather than the encoded text may accompany the symbol. The character size and font are not specified, and the message may be printed anywhere in the area surrounding the symbol. The human readable interpretation should not interfere with the symbol itself or the quiet zones.

10.2 Autodiscrimination capability

Data Matrix can be used in an autodiscrimination environment with a number of other symbologies. (See Annex S).

10.3 System considerations

Data Matrix applications must be viewed as a total system solution (see Annex T).

11 Transmitted data

This section describes the standard transmission protocol for compliant readers. These readers may be programmable to support other transmission options. All encoded data characters are included in the data transmission. The symbology control characters and error correction characters are not transmitted. More complex interpretations are addressed below.

11.1 Protocol for FNC1 (ECC 200 only)

When FNC1 appears in the first symbol character position (or in the fifth symbol character position of the first symbol of a Structured Append sequence), it shall signal that the data conforms to the GS1 Application Identifier standard format. FNC1 in any other later position in such symbols acts as a field separator. Transmission of symbology identifiers shall be enabled. The first FNC1 shall not be represented in the transmitted data, although its presence is indicated by the use of the appropriate option value (2) in the symbology identifier (see 11.5).

When used as a field separator, FNC1 shall be represented in the transmitted message by the ASCII character <$^{G}{}_{S}$> (ASCII value 29).

11.2 Protocol for FNC1 in the second position (ECC 200 only)

When FNC1 is in the second symbol character position (or in the sixth symbol character position of the first symbol of a Structured Append sequence), it shall signal that the data conforms to a particular industry standard format. Transmission of symbology identifiers shall be enabled. The first FNC1 shall not be represented in the transmitted data, although its presence is indicated by the use of the appropriate option value (3) in the symbology identifier (see 11.5).

The data encoded in the first symbol character shall be transmitted as normal at the beginning of the data. When used as a field separator, FNC1 shall be represented in the transmitted message by the ASCII character <$^{G}{}_{S}$> (ASCII value 29).

11.3 Protocol for Macro characters in the first position (ECC 200 only)

This protocol is used to encode two specific message headers and trailers in an abbreviated manner in ECC 200 symbols.

When a Macro character is in the first position a preamble and postamble shall be transmitted. If the first symbol character is 236 (i.e. encoding Macro 05), then the preamble [)>$^{R}_{S}$05$^{G}_{S}$ shall precede the encoded data that follows it. If the first symbol character is 237 (i.e. encoding Macro 06), then the preamble [)>$^{R}_{S}$06$^{G}_{S}$ shall precede the encoded data that follows it. The postamble $^{R}_{S}{}^{E}O_{T}$ shall be transmitted after the data in both cases.

11.4 Protocol for ECIs (ECC 200 only)

In systems where ECIs are supported, the use of a symbology identifier prefix is required with every transmission. Whenever an ECI codeword is encountered, it shall be transmitted as the escape character 92_{DEC} (or $5C_{HEX}$), which represents the character "\" (backslash or reverse solidus) in the default interpretation. The next codeword(s) are converted into a 6-digit value, inverting the rules defined in Table 6. The 6-digit value is transmitted as the appropriate ASCII values (48 - 57). Application software recognising \nnnnnn should interpret all subsequent characters as being from the ECI defined by the 6-digit sequence. This interpretation remains in effect until the end of the encoded data or until another ECI sequence is encountered. If the backslash (byte 92_{DEC}) needs to be used as encoded data, transmission shall be as follows. Whenever (ASCII 92_{DEC}) occurs as data, two bytes of that value shall be transmitted, thus a single occurrence is always an escape character and a double occurrence indicates true data.

EXAMPLE

Encoded data: A\\B\C

Transmission: A\\\\B\\C

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

11.5 Symbology identifier

ISO/IEC 15424 provides a standard procedure for reporting the symbology which has been read, together with options set in the decoder and special features encountered in the symbol. Once the structure of the data (including the use of any ECI) has been identified, the appropriate symbology identifier should be added by the decoder as a preamble to the transmitted data. The symbology identifier is required if ECIs appear anywhere in the symbol, or if FNC1 is used as defined in 11.1 or 11.2. See Annex N for the symbology identifier and option values which apply to Data Matrix.

11.6 Transmitted data example

In this example, the two-character message "¶Ж" is to be encoded in ECC 200, using the ASCII encodation scheme. "¶" is represented by a byte value of 182 in Data Matrix's default character set (ECI 000003, which is equivalent to ISO 8859-1). "Ж" is a Cyrillic character not available in ECI 000003, but which can be represented in ISO 8859-5 (ECI 000007) by the same byte value of 182. The complete message can therefore be represented by inserting a switch to ECI 000007 after the first character, as follows: The symbol encodes the message <¶> <Switch to ECI 000007> <Ж>, using the following series of Data Matrix codewords: [Upper Shift] [55] [ECI] [8] [Upper Shift] [55], with decimal values of [235], [55], [241], [8], [235], [55].

NOTE 1 An Upper Shift character, followed by a codeword of value 55, encodes a byte value of 182.

NOTE 2 ECIs are encoded in Data Matrix as the ECI number plus one.

The decoder transmits the following bytes (including the symbology identifier prefix with an option value of 4, which indicates use of the ECI protocol):

93, 100, 52, 182, 92, 48, 48, 48, 48, 48, 55, 182

which, if viewed entirely in the default interpretation, would appear graphically as:]d4¶\000007¶

The decoder is responsible for signalling the switch to ECI 000007, but not for interpreting the result. ECI-aware software in the receiving application would delete the ECI escape sequence \000007, and the Cyrillic character "Ж" would be represented in a system-dependent manner (e.g., by changing the font in a desktop-publishing file). The final result would match the original message of "¶Ж".

Annex A
(normative)

ECC 200 interleaving process

A.1 Schematic illustration

Using the example of the 72 x 72 symbol size, four levels of interleaving are required to encode a total of 368 data codewords and 144 error correction codewords. These are divided into four blocks of 92 data codewords and 36 error correction codewords, a total block length of 128 codewords.

CODEWORD STREAM	data codewords d										error correction codewords ε									
	1	2	3	4	…	…	365	366	367	368	1	2	3	4	…	...	141	142	143	144
BLOCK 1	data codewords d										error correction codewords ε									
	1	5	…	…	…	361	365				1	5	…	…		137	141			
BLOCK 2		data codewords d										error correction codewords ε								
		2	6	…	…	…	362	366				2	6	…	…		138	142		
BLOCK 3			data codewords d										error correction codewords ε							
			3	7	…	…	…	363	367				3	7	…	…		139	143	
BLOCK 4				data codewords d										error correction codewords ε						
				4	8	…	…	…	364	368				4	8	…	…	…	140	144

Figure A.1 — Illustration of interleaving for 72 x 72 symbol

A.2 Starting sequence for interleaving in different sized symbols

The sequence of the interleaved data codewords and error correction codewords is given in Table A.1.

Table A.1 — Sequence of data and error correction codewords for different symbol sizes

Symbol size	Reed-Solomon block	Sequence of data codewords			Sequence of error correction codewords		
52 x 52	1	1, 3, 5	...	201, 203	1, 3, 5	...	81, 83
	2	2, 4, 6	...	202, 204	2, 4, 6	...	82, 84
64 x 64	1	1, 3, 5	...	277, 279	1, 3, 5	...	109, 111
	2	2, 4, 6	...	278, 280	2, 4, 6	...	110, 112
72 x 72	1	1, 5, 9	...	361, 365	1, 5, 9	...	137, 141
	2	2, 6, 10	...	362, 366	2, 6, 10	...	138, 142
	3	3, 7, 11	...	363, 367	3, 7, 11	...	139, 143
	4	4, 8, 12	...	364, 368	4, 8, 12	...	140, 144
80 x 80	1	1, 5, 9	...	449, 453	1, 5, 9	...	185, 189
	2	2, 6, 10	...	450, 454	2, 6, 10	...	186, 190
	3	3, 7, 11	...	451, 455	3, 7, 11	...	187, 191
	4	4, 8, 12	...	452, 456	4, 8, 12	...	188, 192
88 x 88	1	1, 5, 9	...	569, 573	1, 5, 9	...	217, 221
	2	2, 6, 10	...	570, 574	2, 6, 10	...	218, 222
	3	3, 7, 11	...	571, 575	3, 7, 11	...	219, 223
	4	4, 8, 12	...	572, 576	4, 8, 12	...	220, 224
96 x 96	1	1, 5, 9	...	689, 693	1, 5, 9	...	265, 269
	2	2, 6, 10	...	690, 694	2, 6, 10	...	266, 270
	3	3, 7, 11	...	691, 695	3, 7, 11	...	267, 271
	4	4, 8, 12	...	692, 696	4, 8, 12	...	268, 272
104 x 104	1	1, 7, 13	...	805, 811	1, 7, 13	...	325, 331
	2	2, 8, 14	...	806, 812	2, 8, 14	...	326, 332
	3	3, 9, 15	...	807, 813	3, 9, 15	...	327, 333
	4	4, 10, 16	...	808, 814	4, 10, 16	...	328, 334
	5	5, 11, 17	...	809, 815	5, 11, 17	...	329, 335
	6	6, 12, 18	...	810, 816	6, 12, 18	...	330, 336
120 x 120	1	1, 7, 13	...	1039, 1045	1, 7, 13	...	397, 403
	2	2, 8, 14	...	1040, 1046	2, 8, 14	...	398, 404
	3	3, 9, 15	...	1041, 1047	3, 9, 15	...	399, 405
	4	4, 10, 16	...	1042, 1048	4, 10, 16	...	400, 406
	5	5, 11, 17	...	1043, 1049	5, 11, 17	...	401, 407
	6	6, 12, 18	...	1044, 1050	6, 12, 18	...	402, 408

Symbol size	Reed-Solomon block	Sequence of data codewords			Sequence of error correction codewords		
132 x 132	1	1, 9, 17	...	1289, 1297	1, 9, 17	...	481, 489
	2	2, 10, 18	...	1290, 1298	2, 10, 18	...	482, 490
	3	3, 11, 19	...	1291, 1299	3, 11, 19	...	483, 491
	4	4, 12, 20	...	1292, 1300	4, 12, 20	...	484, 492
	5	5, 13, 21	...	1293, 1301	5, 13, 21	...	485, 493
	6	6, 14, 22	...	1294, 1302	6, 14, 22	...	486, 494
	7	7, 15, 23	...	1295, 1303	7, 15, 23	...	487, 495
	8	8, 16, 24	...	1296, 1304	8, 16, 24	...	488, 496
144 x 144	1	1, 11, 21	...	1541, 1551	1, 11, 21	...	601, 611
	2	2, 12, 22	...	1542, 1552	2, 12, 22	...	602, 612
	3	3, 13, 23	...	1543, 1553	3, 13, 23	...	603, 613
	4	4, 14, 24	...	1544, 1554	4, 14, 24	...	604, 614
	5	5, 15, 25	...	1545, 1555	5, 15, 25	...	605, 615
	6	6, 16, 26	...	1546, 1556	6, 16, 26	...	606, 616
	7	7, 17, 27	...	1547, 1557	7, 17, 27	...	607, 617
	8	8, 18, 28	...	1548, 1558	8, 18, 28	...	608, 618
	9	9, 19, 29	...	1549	9, 19, 29	...	609, 619
	10	10, 20, 30	...	1550	10, 20, 30	...	610, 620

Annex B
(normative)

ECC 200 pattern randomising

The pattern randomising algorithms convert an input codeword at a given position to a new randomised output codeword.

B.1 253-state algorithm

This algorithm adds a pseudo-random number to the Pad codeword value. The pseudo-random number will always be in the range 1 to 253 and the randomised Pad codeword value will be in the range 1 to 254.

The variable Pad_codeword_position is the number of data codewords from the beginning of encoded data.

B.1.1 253-state randomising algorithm

INPUT (Pad_codeword_value, Pad_codeword_position)

pseudo_random_number = ((149 * Pad_codeword_position) mod 253) + 1

temp_variable = Pad_codeword_value + pseudo_random_number

IF (temp_variable <= 254)

OUTPUT (randomised_Pad_codeword_value = temp_variable)

ELSE

OUTPUT (randomised_Pad_codeword_value = temp_variable - 254)

B.1.2 253-state un-randomising algorithm

INPUT (randomised_Pad_codeword_value, Pad_codeword_position)

pseudo_random_number = ((149 * Pad_codeword_position) mod 253) + 1

temp_variable = randomised_Pad_codeword_value - pseudo_random_number

IF (temp_variable >= 1)

OUTPUT (Pad_codeword_value = temp_variable)

ELSE

OUTPUT (Pad_codeword_value = temp_variable + 254)

B.2 255-state algorithm

This algorithm adds a pseudo-random number to the Base 256 encodation codeword value. The pseudorandom number will always be in the range 1 to 255 and the randomised Base 256 codeword value will be in the range 0 to 255.

The variable Base256_codeword_position is the number of data codewords from the beginning of encoded data.

B.2.1 255-state randomising algorithm

```
INPUT ( Base256_codeword_value, Base256_codeword_position )

pseudo_random_number = ( ( 149 * Base256_codeword_position ) mod 255 ) + 1

temp_variable = Base256_codeword_value + pseudo_random_number

IF ( temp_variable <= 255 )

    OUTPUT (randomised_Base256_codeword_value = temp_variable )

ELSE

    OUTPUT (randomised_Base256_codeword_value = temp_variable - 256 )
```

B.2.2 255-state un-randomising algorithm

```
INPUT ( randomised_Base256_codeword_value, Base256_codeword_position )

pseudo_random_number = ( ( 149 * Base256_codeword_position ) mod 255 ) + 1

temp_variable=randomised_Base256_codeword_value - pseudo_random_number

IF ( temp_variable >= 0 )

    OUTPUT ( Base256_codeword_value = temp_variable )

ELSE

    OUTPUT ( Base256_codeword_value = temp_variable + 256 )
```

Annex C
(normative)

ECC 200 encodation character sets

Table C.1 — C40 encodation character set

C40 Value	Basic set		Shift 1 set		Shift 2 set		Shift 3 set	
	Char	Decimal	Char	Decimal	Char	Decimal	Char	Decimal
0	Shift 1		NUL	0	!	33	‘	96
1	Shift 2		SOH	1	“	34	a	97
2	Shift 3		STX	2	#	35	b	98
3	space	32	ETX	3	$	36	c	99
4	0	48	EOT	4	%	37	d	100
5	1	49	ENQ	5	&	38	e	101
6	2	50	ACK	6	‘	39	f	102
7	3	51	BEL	7	(	40	g	103
8	4	52	BS	8	)	41	h	104
9	5	53	HT	9	*	42	i	105
10	6	54	LF	10	+	43	j	106
11	7	55	VT	11	,	44	k	107
12	8	56	FF	12	-	45	l	108
13	9	57	CR	13	.	46	m	109
14	A	65	SO	14	/	47	n	110
15	B	66	SI	15	:	58	o	111
16	C	67	DLE	16	;	59	p	112
17	D	68	DC1	17	<	60	q	113
18	E	69	DC2	18	=	61	r	114
19	F	70	DC3	19	>	62	s	115
20	G	71	DC4	20	?	63	t	116
21	H	72	NAK	21	@	64	u	117
22	I	73	SYN	22	[	91	v	118
23	J	74	ETB	23	\	92	w	119
24	K	75	CAN	24	]	93	x	120
25	L	76	EM	25	^	94	y	121
26	M	77	SUB	26	_	95	z	122
27	N	78	ESC	27	FNC1		{	123
28	O	79	FS	28			|	124
29	P	80	GS	29			}	125
30	Q	81	RS	30	Upper Shift		~	126
31	R	82	US	31			DEL	127
32	S	83						
33	T	84						
34	U	85						
35	V	86						

C40 Value	Basic set		Shift 1 set		Shift 2 set		Shift 3 set	
	Char	Decimal	Char	Decimal	Char	Decimal	Char	Decimal
36	W	87						
37	X	88						
38	Y	89						
39	Z	90						

NOTE The relationship between the ASCII decimal value and the C40 value remains constant regardless of which ECI is in effect.

Table C.2 — Text encodation character set

Text value	Basic set		Shift 1 set		Shift 2 set		Shift 3 set	
	Char	Decimal	Char	Decimal	Char	Decimal	Char	Decimal
0	Shift	1	NUL	0	!	33	‘	96
1	Shift	2	SOH	1	“	34	A	65
2	Shift	3	STX	2	#	35	B	66
3	space	32	ETX	3	$	36	C	67
4	0	48	EOT	4	%	37	D	68
5	1	49	ENQ	5	&	38	E	69
6	2	50	ACK	6	‘	39	F	70
7	3	51	BEL	7	(	40	G	71
8	4	52	BS	8	)	41	H	72
9	5	53	HT	9	*	42	I	73
10	6	54	LF	10	+	43	J	74
11	7	55	VT	11	,	44	K	75
12	8	56	FF	12	-	45	L	76
13	9	57	CR	13	.	46	M	77
14	a	97	SO	14	/	47	N	78
15	b	98	SI	15	:	58	O	79
16	c	99	DLE	16	;	59	P	80
17	d	100	DC1	17	<	60	Q	81
18	e	101	DC2	18	=	61	R	82
19	f	102	DC3	19	>	62	S	83
20	g	103	DC4	20	?	63	T	84
21	h	104	NAK	21	@	64	U	85
22	i	105	SYN	22	[	91	V	86
23	j	106	ETB	23	\	92	W	87
24	k	107	CAN	24	]	93	X	88
25	l	108	EM	25	^	94	Y	89
26	m	109	SUB	26	_	95	Z	90
27	n	110	ESC	27	FNC1		{	123
28	o	111	FS	28			|	124
29	p	112	GS	29			}	125
30	q	113	RS	30	Upper	Shift	~	126
31	r	114	US	31			DEL	127
32	s	115						
33	t	116						
34	u	117						
35	v	118						
36	w	119						
37	x	120						

Text value	Basic set		Shift 1 set		Shift 2 set		Shift 3 set	
	Char	Decimal	Char	Decimal	Char	Decimal	Char	Decimal
38	y	121						
39	z	122						

NOTE The relationship between the ASCII decimal value and the Text value remains constant regardless of which ECI is in effect.

Table C.3 — EDIFACT encodation character set

Data character			EDIFACT binary value	Data character			EDIFACT binary value
Char	Decimal value	Binary value		Char	Decimal value	Binary value	
@	64	01000000	000000	space	32	00100000	100000
A	65	01000001	000001	!	33	00100001	100001
B	66	01000010	000010	“	34	00100010	100010
C	67	01000011	000011	#	35	00100011	100011
D	68	01000100	000100	$	36	00100100	100100
E	69	01000101	000101	%	37	00100101	100101
F	70	01000110	000110	&	38	00100110	100110
G	71	01000111	000111	‘	39	00100111	100111
H	72	01001000	001000	(	40	00101000	101000
I	73	01001001	001001	)	41	00101001	101001
J	74	01001010	001010	*	42	00101010	101010
K	75	01001011	001011	+	43	00101011	101011
L	76	01001100	001100	,	44	00101100	101100
M	77	01001101	001101	-	45	00101101	101101
N	78	01001110	001110	.	46	00101110	101110
O	79	01001111	001111	/	47	00101111	101111
P	80	01010000	010000	0	48	00110000	110000
Q	81	01010001	010001	1	49	00110001	110001
R	82	01010010	010010	2	50	00110010	110010
S	83	01010011	010011	3	51	00110011	110011
T	84	01010100	010100	4	52	00110100	110100
U	85	01010101	010101	5	53	00110101	110101
V	86	01010110	010110	6	54	00110110	110110
W	87	01010111	010111	7	55	00110111	110111
X	88	01011000	011000	8	56	00111000	111000
Y	89	01011001	011001	9	57	00111001	111001
Z	90	01011010	011010	:	58	00111010	111010
[	91	01011011	011011	;	59	00111011	111011
\	92	01011100	011100	<	60	00111100	111100
]	93	01011101	011101	=	61	00111101	111101
^	94	01011110	011110	>	62	00111110	111110
Unlatch		01011111	011111	?	63	00111111	111111

NOTE The relationship between the ASCII decimal value and the EDIFACT value remain constant regardless of which ECI is in effect.

Annex D
(normative)

ECC 200 alignment patterns

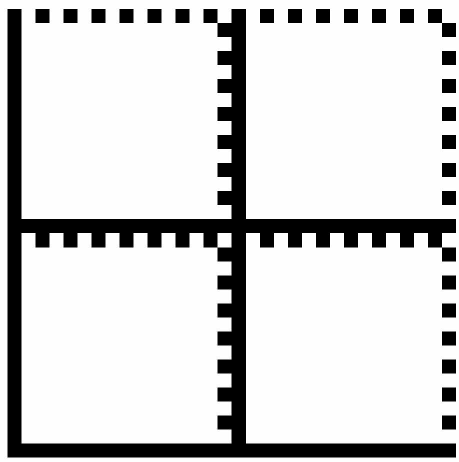


Figure D.1 — Alignment pattern configuration for 32 x 32 square symbol

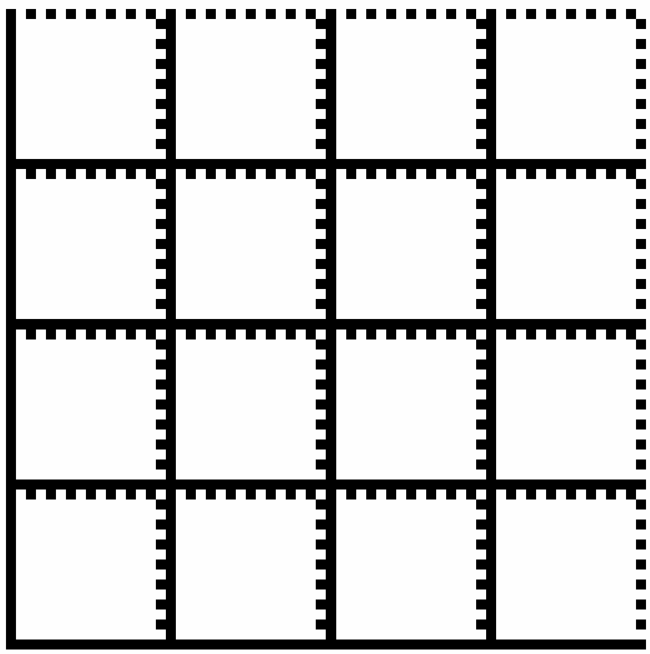


Figure D.2 — Alignment pattern configuration for 64 x 64 square symbol

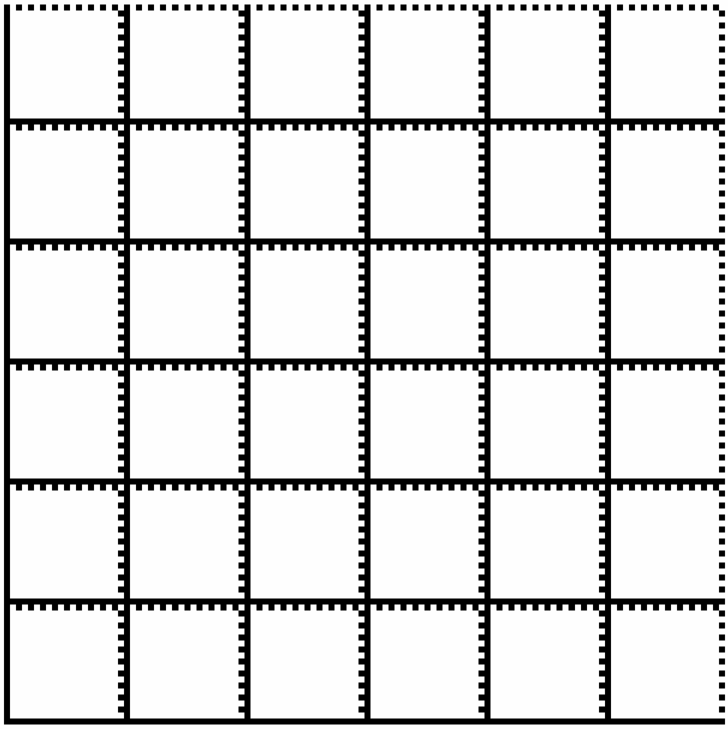


Figure D.3 — Alignment pattern configuration for 120 x 120 square symbol

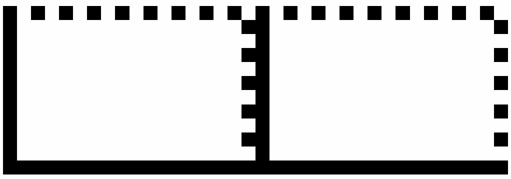


Figure D.4 — Alignment pattern configuration for 12 x 36 rectangular symbol

Annex E
(normative)

ECC 200 Reed-Solomon error detection and correction

E.1 Error correction codeword generator polynomials

The error correction codewords are the coefficients of the remainder resulting from first multiplying the symbol data polynomial d(x) by x^k and then dividing it by the generator polynomial g(x). Each generator polynomial is the product of the first-degree polynomials: $x - 2^1$, $x - 2^2$, ..., $x - 2^n$; where n is the degree of the generator polynomial.

For example the fifth degree generator polynomial is:

$(x + 2)(x + 4)(x + 8)(x + 16)(x + 32)$

$= x^5 + (2 + 4 + 8 + 16 + 32)x^4 + ((2 * 4) + (2 * 8) + (2 * 16) + (2 * 32) + (4 * 8) + (4 * 16) + (4 * 32) + (8 * 16) + (8 * 32) + (16 * 32))x^3 + ((2 * 4 * 8) + (2 * 4 * 16) + (2 * 4 * 32) + (2 * 8 * 16) + (2 * 8 * 32) + (2 * 16 * 32) + (4 * 8 * 16) + (4 * 8 * 32) + (4 * 16 * 32) + (8 * 16 * 32))x^2 + ((2 * 4 * 8 * 16) + (2 * 4 * 8 * 32) + (2 * 4 * 16 * 32) + (2 * 8 * 16 * 32) + (4 * 8 * 16 * 32))x + (2 * 4 * 8 * 16 * 32)$

$= x^5 + 62x^4 + 111x^3 + 15x^2 + 48x + 228.$

Note that this Galois Field arithmetic is not normal integer arithmetic: - is equivalent to +, which is an "exclusive-or" operation in this Field, and multiplication is byte-wise modulo 100101101 for each binary polynomial term generated by bit-by-bit multiplication.

The polynomial divisor for generating 5 check characters is:

$g(x) = x^5 + 62x^4 + 111x^3 + 15x^2 + 48x + 228.$

The polynomial divisor for generating 7 check characters is:

$g(x) = x^7 + 254x^6 + 92x^5 + 240x^4 + 134x^3 + 144x^2 + 68x + 23.$

The polynomial divisor for generating 10 check characters is:

$g(x) = x^{10} + 61x^9 + 110x^8 + 255x^7 + 116x^6 + 248x^5 + 223x^4 + 166x^3 + 185x^2 + 24x + 28.$

The polynomial divisor for generating 11 check characters is:

$g(x) = x^{11} + 120x^{10} + 97x^9 + 60x^8 + 245x^7 + 39x^6 + 168x^5 + 194x^4 + 12x^3 + 205x^2 + 138x + 175.$

The polynomial divisor for generating 12 check characters is:

$g(x) = x^{12} + 242x^{11} + 100x^{10} + 178x^9 + 97x^8 + 213x^7 + 142x^6 + 42x^5 + 61x^4 + 91x^3 + 158x^2 + 153x + 41.$

The polynomial divisor for generating 14 check characters is:

$g(x) = x^{14} + 185x^{13} + 83x^{12} + 186x^{11} + 18x^{10} + 45x^9 + 138x^8 + 119x^7 + 157x^6 + 9x^5 + 95x^4 + 252x^3 + 192x^2 + 97x + 156.$

The polynomial divisor for generating 18 check characters is:

$$g(x) = x^{18} + 188x^{17} + 90x^{16} + 48x^{15} + 225x^{14} + 254x^{13} + 94x^{12} + 129x^{11} + 109x^{10} + 213x^{9} + 241x^{8} + 61x^{7} + 66x^{6} + 75x^{5} + 188x^{4} + 39x^{3} + 100x^{2} + 195x + 83.$$

The polynomial divisor for generating 20 check characters is:

$$g(x) = x^{20} + 172x^{19} + 186x^{18} + 174x^{17} + 27x^{16} + 82x^{15} + 108x^{14} + 79x^{13} + 253x^{12} + 145x^{11} + 153x^{10} + 160x^{9} + 188x^{8} + 2x^{7} + 168x^{6} + 71x^{5} + 233x^{4} + 9x^{3} + 244x^{2} + 195x + 15.$$

The polynomial divisor for generating 24 check characters is:

$$g(x) = x^{24} + 193x^{23} + 50x^{22} + 96x^{21} + 184x^{20} + 181x^{19} + 12x^{18} + 124x^{17} + 254x^{16} + 172x^{15} + 5x^{14} + 21x^{13} + 155x^{12} + 223x^{11} + 251x^{10} + 197x^{9} + 155x^{8} + 21x^{7} + 176x^{6} + 39x^{5} + 109x^{4} + 205x^{3} + 88x^{2} + 190x + 52.$$

The polynomial divisor for generating 28 check characters is:

$$g(x) = x^{28} + 255x^{27} + 93x^{26} + 168x^{25} + 233x^{24} + 151x^{23} + 120x^{22} + 136x^{21} + 141x^{20} + 213x^{19} + 110x^{18} + 138x^{17} + 17x^{16} + 121x^{15} + 249x^{14} + 34x^{13} + 75x^{12} + 53x^{11} + 170x^{10} + 151x^{9} + 37x^{8} + 174x^{7} + 103x^{6} + 96x^{5} + 71x^{4} + 97x^{3} + 43x^{2} + 231x + 211.$$

The polynomial divisor for generating 36 check characters is:

$$g(x) = x^{36} + 112x^{35} + 81x^{34} + 98x^{33} + 225x^{32} + 25x^{31} + 59x^{30} + 184x^{29} + 175x^{28} + 44x^{27} + 115x^{26} + 119x^{25} + 95x^{24} + 137x^{23} + 101x^{22} + 33x^{21} + 68x^{20} + 4x^{19} + 2x^{18} + 18x^{17} + 229x^{16} + 182x^{15} + 80x^{14} + 251x^{13} + 220x^{12} + 179x^{11} + 84x^{10} + 120x^{9} + 102x^{8} + 181x^{7} + 162x^{6} + 250x^{5} + 130x^{4} + 218x^{3} + 242x^{2} + 127x + 245.$$

The polynomial divisor for generating 42 check characters is:

$$g(x) = x^{42} + 5x^{41} + 9x^{40} + 5x^{39} + 226x^{38} + 177x^{37} + 150x^{36} + 50x^{35} + 69x^{34} + 202x^{33} + 248x^{32} + 101x^{31} + 54x^{30} + 57x^{29} + 253x^{28} + x^{27} + 21x^{26} + 121x^{25} + 57x^{24} + 111x^{23} + 214x^{22} + 105x^{21} + 167x^{20} + 9x^{19} + 100x^{18} + 95x^{17} + 175x^{16} + 8x^{15} + 242x^{14} + 133x^{13} + 245x^{12} + 2x^{11} + 122x^{10} + 105x^{9} + 247x^{8} + 153x^{7} + 22x^{6} + 38x^{5} + 19x^{4} + 31x^{3} + 137x^{2} + 193x + 77.$$

The polynomial divisor for generating 48 check characters is:

$$g(x) = x^{48} + 19x^{47} + 225x^{46} + 253x^{45} + 92x^{44} + 213x^{43} + 69x^{42} + 175x^{41} + 160x^{40} + 147x^{39} + 187x^{38} + 87x^{37} + 176x^{36} + 44x^{35} + 82x^{34} + 240x^{33} + 186x^{32} + 138x^{31} + 66x^{30} + 100x^{29} + 120x^{28} + 88x^{27} + 131x^{26} + 205x^{25} + 170x^{24} + 90x^{23} + 37x^{22} + 23x^{21} + 118x^{20} + 147x^{19} + 16x^{18} + 106x^{17} + 191x^{16} + 87x^{15} + 237x^{14} + 188x^{13} + 205x^{12} + 231x^{11} + 238x^{10} + 133x^{9} + 238x^{8} + 22x^{7} + 117x^{6} + 32x^{5} + 96x^{4} + 223x^{3} + 172x^{2} + 132x + 245.$$

The polynomial divisor for generating 56 check characters is:

$$g(x) = x^{56} + 46x^{55} + 143x^{54} + 53x^{53} + 233x^{52} + 107x^{51} + 203x^{50} + 43x^{49} + 155x^{48} + 28x^{47} + 247x^{46} + 67x^{45} + 127x^{44} + 245x^{43} + 137x^{42} + 13x^{41} + 164x^{40} + 207x^{39} + 62x^{38} + 117x^{37} + 201x^{36} + 150x^{35} + 22x^{34} + 238x^{33} + 144x^{32} + 232x^{31} + 29x^{30} + 203x^{29} + 117x^{28} + 234x^{27} + 218x^{26} + 146x^{25} + 228x^{24} + 54x^{23} + 132x^{22} + 200x^{21} + 38x^{20} + 223x^{19} + 36x^{18} + 159x^{17} + 150x^{16} + 235x^{15} + 215x^{14} + 192x^{13} + 230x^{12} + 170x^{11} + 175x^{10} + 29x^{9} + 100x^{8} + 208x^{7} + 220x^{6} + 17x^{5} + 12x^{4} + 238x^{3} + 223x^{2} + 9x + 175.$$

The polynomial divisor for generating 62 check characters is:

$$g(x) = x^{62} + 204x^{61} + 11x^{60} + 47x^{59} + 86x^{58} + 124x^{57} + 224x^{56} + 166x^{55} + 94x^{54} + 7x^{53} + 232x^{52} + 107x^{51} + 4x^{50} + 170x^{49} + 176x^{48} + 31x^{47} + 163x^{46} + 17x^{45} + 188x^{44} + 130x^{43} + 40x^{42} + 10x^{41} + 87x^{40} + 63x^{39} + 51x^{38} + 218x^{37} + 27x^{36} + 6x^{35} + 147x^{34} + 44x^{33} + 161x^{32} + 71x^{31} + 114x^{30} + 64x^{29} + 175x^{28} + 221x^{27} + 185x^{26} + 106x^{25} + 250x^{24} + 190x^{23} + 197x^{22} + 63x^{21} + 245x^{20} + 230x^{19} + 134x^{18} + 112x^{17} + 185x^{16} + 37x^{15} + 196x^{14} + 108x^{13} + 143x^{12} + 189x^{11} + 201x^{10} + 188x^{9} + 202x^{8} + 118x^{7} + 39x^{6} + 210x^{5} + 144x^{4} + 50x^{3} + 169x^{2} + 93x + 242.$$

The polynomial divisor for generating 68 check characters is:

$g(x) = x^{68} + 186x^{67} + 82x^{66} + 103x^{65} + 96x^{64} + 63x^{63} + 132x^{62} + 153x^{61} + 108x^{60} + 54x^{59} + 64x^{58} + 189x^{57} + 211 x^{56} + 232x^{55} + 49x^{54} + 25x^{53} + 172x^{52} + 52x^{51} + 59x^{50} + 241x^{49} + 181x^{48} + 239x^{47} + 223x^{46} + 136x^{45} + 231x^{44} + 210x^{43} + 96x^{42} + 232x^{41} + 220x^{40} + 25x^{39} + 179x^{38} + 167x^{37} + 202x^{36} + 185x^{35} + 153x^{34} + 139x^{33} + 66x^{32} + 236 x^{31} + 227x^{30} + 160x^{29} + 15x^{28} + 213x^{27} + 93x^{26} + 122x^{25} + 68x^{24} + 177x^{23} + 158x^{22} + 197x^{21} + 234x^{20} + 180x^{19} + 248x^{18} + 136x^{17} + 213x^{16} + 127x^{15} + 73x^{14} + 36x^{13} + 154x^{12} + 244x^{11} + 147x^{10} + 33x^{9} + 89x^{8} + 56x^{7} + 159x^{6} + 149x^{5} + 251x^{4} + 89x^{3} + 173x^{2} + 228x + 220.$

E.2 Error correction calculation

The Peterson-Gorenstein-Zierler algorithm may be used to correct errors in decoded ECC 200 symbols.

The calculation described below follows this error correcting algorithm, using the Reed-Solomon error correction codewords.

Erasures shall be corrected as errors by initially filling any erasure codeword positions with dummy values.

All calculations shall be done using GF(2^8) arithmetic operations. Addition and subtraction are equivalent to the binary XOR operation. Multiplication and division can be performed using log and antilog tables.

Construct the symbol character polynomial $C(x) = C_{n-1}x^{n-1} + C_{n-2}x^{n-2} + ... + C_1x^1 + C_0$ where the n coefficients are the codewords read with C_{n-1} being the first symbol character and where n is the total number of symbol characters.

Calculate i syndrome values S_0 through S_{i-1} by evaluating $C(x)$ at $x = 2^k$ for $k = 1$ through i, where i is the number of error correction codewords in the symbol.

Form and solve j simultaneous equations with j unknowns L_0 through L_{j-1} using the i syndromes:

$$S_0L_0 + S_1L_1 + ... + S_{j-1}L_{j-1} = S_j$$

$$S_1L_0 + S_2L_1 + ... + S_jL_{j-1} = S_{j+1}$$

:

:

$$S_{j-1}L_0 + S_jL_1 + ... + S_{2j-2}L_{j-1} = S_{2j-1}$$

where j is $i/2$.

Construct the error locator polynomial:

$$L(x) = L_{j-1}x^j + L_{j-2}x^{j-1} + ... + L_0x + 1$$

from the j values of L obtained above. Evaluate $L(x)$ at $x = 2^k$ for $k = 0$ through $n - 1$ where n is the total number of symbol characters in the symbol.

Whenever $L(2^k) = 0$, an error location is given by $n - 1 - k$. If more than j error locations are found, the symbol is not correctable.

Save the error locations in m error location variables E_0 through E_{m-1} where m is the number of error locations found. Form and solve m simultaneous equations with m unknowns X_0 through X_{m-1} (the error magnitudes) using the error location variables E and the first m syndromes S:

$$E_0X_0 + E_1X_1 + ... + E_{m-1}X_{m-1} = S_0$$

$$E_0^2X_0 + E_1^2X_1 + ... + E_{(m-1)}^2X_{m-1} = S_1$$

$$E_0^3X_0 + E_1^3X_1 + ... + E_{(m-1)}^3X_{m-1} = S_2$$

:

:

$$E_0^mX_0 + E_1^mX_1 + ... + E_{(m-1)}^mX_{m-1} = S_{m-1}$$

Add the error magnitudes X_0 through $X_{m-}1$ to the symbol character values at the corresponding error locations E_0 through E_{m-1} to correct the errors.

NOTE E_0 …. E_{m-1} – are the roots of the error locator polynomial.

This algorithm, written in C, is available from AIM, Inc. on the Data Matrix Developers Diskette (see Bibliography).

E.3 Calculation of error correction codewords

The following is an example of a generic routine, written in C, which calculates the error correction codewords for a given data codeword string of length "nd", stored as an integer array wd[]. The function ReedSolomon() first generates log and antilog tables for the Galois Field of size "gf" (in the case of ECC 200, 2^8) with prime modulus "pp" (in the case of ECC 200, 301), then uses them in the function prod(), first to calculate coefficients of the generator polynomial of order "nc" and then to calculate "nc" additional check codewords which are appended to the data in wd[].

```
/* "prod(x,y,log,alog,gf)" returns the product "x" times "y" */
int prod(int x, int y, int *log, int *alog, int gf) {
   if (!x || !y) return 0;
   ELSE return alog[(log[x] + log[y]) % (gf-1)];
}

/* "ReedSolomon(wd,nd,nc,gf.pp)" takes "nd" data codeword values in wd[] */
/* and adds on "nc" check codewords, all within GF(gf) where "gf" is a */
/* power of 2 and "pp" is the value of its prime modulus polynomial */
void ReedSolomon(int *wd, int nd, int nc, int gf, int pp) {
   int i, j, k. *log,*alog,*c;

/* allocate, then generate the log & antilog arrays: */
   log = malloc(sizeof(int) * gf);
   alog = malloc(sizeof(int) * gf);
   log[0] = 1-gf; alog[0] = 1;
   for (i = 1; i < gf; i++) {
      alog[i] = alog[i-1] * 2;
      if (alog[i] >= gf) alog[i] ^= pp;
      log[alog[i]] = i;
}

/* allocate, then generate the generator polynomial coefficients: */
   c = malloc(sizeof(int) * (nc+1));
   for (i=1; i<=nc; i++) c[i] = 0; c[0] = 1;
```

```
    for (i=1; i<=nc; i++) {
        c[i] = c[i-1];
        for (j=i-1; j>=1; j--) {
            c[j] = c[j-1] ^ prod(c[j],alog[i],log,alog,gf);
        }
        c[0] = prod(c[0],alog[i],log,alog,gf);
    }

/* clear, then generate "nc" checkwords in the array wd[] : */
    for (i=nd; i<=(nd+nc); i++) wd[i] = 0;
    for (i=0; i<nd; i++) {
        k = wd[nd] ^ wd[i] ;
        for (j=0; j<nc; j++) {
            wd[nd+j] = wd[nd+j+1] ^ prod(k,c[nc-j-1],log, alog,gf);
        }
    }

    free(c);
    free(alog);
    free(log);
}
```

Annex F
(normative)

ECC 200 symbol character placement

F.1 Symbol character placement

The following C language program generates symbol character placement diagrams:

```
#include <stdio.h>
#include <alloc.h>

int nrow, ncol, *array;

/* "module" places "chr+bit" with appropriate wrapping within array[] */
void module(int row, int col, int chr, int bit)
{ if (row < 0) { row += nrow; col += 4 - ((nrow+4)%8); }
   if (col < 0) { col += ncol; row += 4 - ((ncol+4)%8); }
   array[row*ncol+col] = 10*chr + bit;
}
/* "utah" places the 8 bits of a utah-shaped symbol character in ECC200 */
void utah(int row, int col, int chr)
{ module(row-2,col-2,chr,1);
   module(row-2,col-1,chr,2);
   module(row-1,col-2,chr,3);
   module(row-1,col-1,chr,4);
   module(row-1,col,chr,5);
   module(row,col-2,chr,6);
   module(row,col-1,chr,7);
   module(row,col,chr,8);
}
/* "cornerN" places 8 bits of the four special corner cases in ECC200 */
void corner1(int chr)
{ module(nrow-1,0,chr,1);
   module(nrow-1,1,chr,2);
   module(nrow-1,2,chr,3);
   module(0,ncol-2,chr,4);
   module(0,ncol-1,chr,5);
   module(1,ncol-1,chr,6);
   module(2,ncol-1,chr,7);
   module(3,ncol-1,chr,8);
}
void corner2(int chr)
{ module(nrow-3,0,chr,1);
   module(nrow-2,0,chr,2);
   module(nrow-1,0,chr,3);
   module(0,ncol-4,chr,4);
   module(0,ncol-3,chr,5);
   module(0,ncol-2,chr,6);
   module(0,ncol-1,chr,7);
   module(1,ncol-1,chr,8);
}
void corner3(int chr)
{ module(nrow-3,0,chr,1);
   module(nrow-2,0,chr,2);
   module(nrow-1,0,chr,3);
   module(0,ncol-2,chr,4);
   module(0,ncol-1,chr,5);
   module(1,ncol-1,chr,6);
```

```
   module(2,ncol-1,chr,7);
   module(3,ncol-1,chr,8);
}
void corner4(int chr)
{ module(nrow-1,0,chr,1);
   module(nrow-1,ncol-1,chr,2);
   module(0,ncol-3,chr,3);
   module(0,ncol-2,chr,4);
   module(0,ncol-1,chr,5);
   module(1,ncol-3,chr,6);
   module(1,ncol-2,chr,7);
   module(1,ncol-1,chr,8);
}
/* "ECC200" fills an nrow x ncol array with appropriate values for ECC200 */
void ECC200(void)
{ int row, col, chr;

/* First, fill the array[] with invalid entries */
   for (row=0; row<nrow; row++) {
      for (col=0; col<ncol; col++) {
         array[row*ncol+col] = 0;
      }
   }
/* Starting in the correct location for character #1, bit 8,... */
   chr = 1; row = 4; col = 0;

   do {
/* repeatedly first check for one of the special corner cases, then... */
      if ((row == nrow) && (col == 0)) corner1(chr++);
      if ((row == nrow-2) && (col == 0) && (ncol%4)) corner2(chr++);
      if ((row == nrow-2) && (col == 0) && (ncol%8 == 4)) corner3(chr++);
      if ((row == nrow+4) && (col == 2) && (!(ncol%8))) corner4(chr++);
/* sweep upward diagonally, inserting successive characters,... */
   do {
      if ((row < nrow) && (col >= 0) && (!array[row*ncol+col]))
         utah(row,col,chr++);
      row -= 2; col += 2;
   } while ((row >= 0) && (col < ncol));
   row += 1; col += 3;

/* & then sweep downward diagonally, inserting successive characters,... */
   +
   do {
      if ((row >= 0) && (col < ncol) && (!array[row*ncol+col]))
         utah(row,col,chr++);
      row += 2; col -= 2;
   } while ((row < nrow) && (col >= 0));
   row += 3; col += 1;

/* ... until the entire array is scanned */
   } while ((row < nrow) || (col < ncol));

/* Lastly, if the lower righthand corner is untouched, fill in fixed pattern */
   if (!array[nrow*ncol-1]) {
      array[nrow*ncol-1] = array[nrow*ncol-ncol-2] = 1;
   }
}

/* "main" checks for valid command line entries, then computes & displays array
*/
void main(int argc, char *argv[])
{ int x, y, z;

   if (argc =< 3) {
      printf("Command line: ECC200 #_of_Data_Rows #_of_Data_Columns\n");
```

```
    } ELSE {
        nrow = ncol = 0;
        nrow = atoi(argv[1]); ncol = atoi(argv[2]);
        if ((nrow >= 6) && (~nrow&0x01) && (ncol >= 6) && (~ncol&0x01)) {
            array = malloc(sizeof(int) * nrow * ncol);

            ECC200();

            for (x=0; x<nrow; x++) {
                for (y=0; y<ncol; y++) {
                    z = array[x*ncol+y];
                    if (z == 0) printf(" WHI");
                        ELSE if (z == 1) printf("BLK");
                            ELSE printf("%3d.%d",z/10,z%10);
                }
                printf("\n");
            }
            free(array);
        }
    }
}
```

F.2 Symbol character placement rules

F.2.1 Non-standard symbol character shapes

Because the standard symbol character shape cannot always fit at the data module boundaries of the symbol and at some corners, a small set of non-standard symbol characters is required. There are six conditions: two boundary conditions which affect all symbol formats, and four different corner conditions which apply to certain symbol formats:

a. One portion of the symbol character shape is placed on one side and the other on the opposite side. This applies to two basic symbol character shapes (see Figure F.1). Variants of these arrangements concern the row-to-row relationship between the left and right hand boundary (see Table F.1).

b. One portion of the symbol character is placed on the top boundary and the other portion on the bottom boundary. This applies to two basic symbol character shapes (see Figure F.2). Variants of these arrangements concern the column-to-column relationship between the top and bottom boundary (see Table F.1).

c. Four symbol character shapes are split between two or three corners (see Figures F.3 to F.6) The non-standard symbol shapes are placed at opposite boundaries. The number of these pairings increases in general proportion to the size of the perimeter of the mapping matrix. The basic pattern is as illustrated in Figures F.1 and F.2. In Figure F.1, modules a8 and a7 are in the same row, as are modules b7 and b6. In Figure F.2 module c6 and c3 are in the same column as are modules d3 and d1. There are seven cases for boundary placement, which define the relative vertical position of the symbol characters illustrated in Figure F.1, the horizontal position of the symbol characters illustrated in Figure F.2, and the corner conditions.

Table F.1 — Factors which determine the boundary placement cases

Boundary placement case	Row relationship of module a8 and a7	Column relationship of module c6 and c3	Corner condition Figure No.	Mapping matrices affected	Refer to Annex F. Figure no. for example
1	a7 Row = a8 Row	c3 Column = c6 Column	None	Square: 8^2, 16^2, 24^2, 32^2, 40^2 48^2, 56^2, 64^2, 72^2, 80^2, 88^2, 96^2, & 120^2	Figure F.9 & F.16
2	a7 Row = a8 Row - 2	c3 Column = c6 Column - 2	None	Square: 10^2 & 18^2	Figure F.10 & F.17
3	a7 Row = a8 Row + 4	c3 Column = c6 Column + 4	F.3	Square: 12^2, 20^2,28^2, 36^2, 44^2, 108^2, & 132^2	Figure F.11 & F.18
4	a7 Row = a8 Row + 2	c3 Column = c6 Column + 2	F.4	Square: 14^2 &22^2	Figure F.12 & F.19
5	a7 Row = a8 Row	c3 Column = c6 Column + 2	F.5	Rectangular: 6 x 16 & 14 x 32	Figure F.13
6	a7 Row = a8 Row	c3 Column = c6 Column - 2	None	Rectangular: 10 x 24 & 10 x 32	Figure F.14
7	a7 Row = a8 Row + 4	c3 Column = c6 Column + 2	F.6	Rectangular: 6 x 28 & 14 x 44	Figure F.15

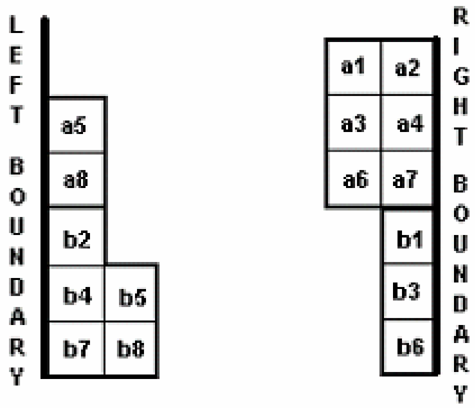


Figure F.1 — Left and right symbol characters

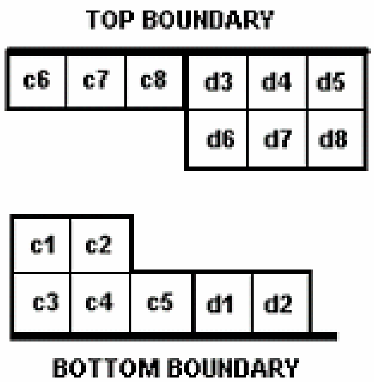


Figure F.2 — Top and bottom symbol characters

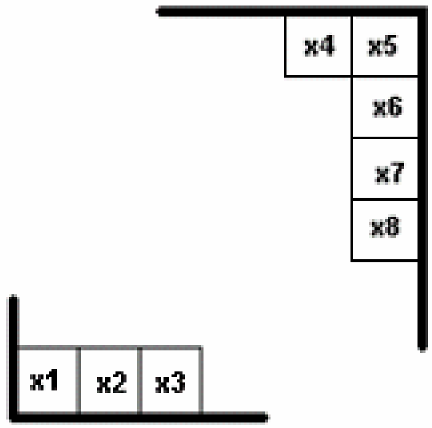


Figure F.3 — Corner condition 1

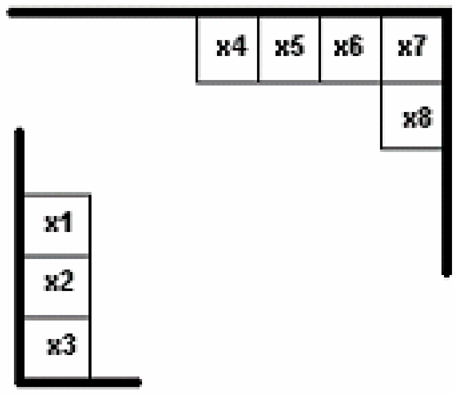


Figure F.4 — Corner condition 2

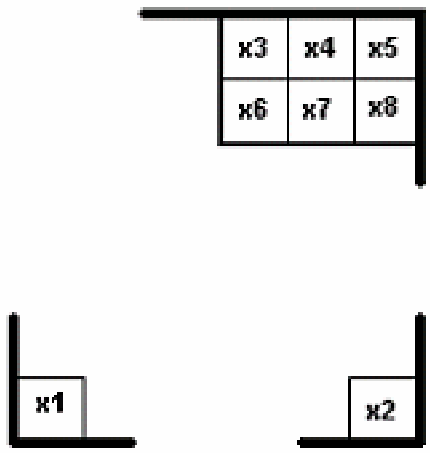


Figure F.5 — Corner condition 3

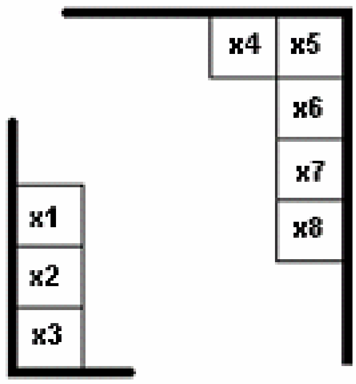


Figure F.6 — Corner condition 4

NOTE 1 Algebraic notation has been used to identify the symbol characters because these vary depending on the symbol format

NOTE 2 The corner characters are identified by the module in the bottom left and top right corners.

F.2.2 Symbol character arrangement

The symbol characters are placed in a matrix in the following manner:

a) A mapping matrix is created.

   1) For small symbols with only one data region, this equates to the mapping matrix.

   2) For larger symbols with more than one data region, the mapping matrix equates to an area the size of the abutted data regions. In effect, the mapping matrix has no separating alignment patterns. For example, the 36 x 36 format symbol has four 16 x 16 data regions which abut to create a mapping matrix 32 x 32. The size of the mapping matrix for each symbol format is given in Table 7. The boundary placement case is given in Table F.1.

b) Symbol character 2 is placed in the uppermost left position, with its modules conforming to the bit (or module) sequence defined in Figure 11. Using the notation 2.1 to identify module 1 of symbol character 2, this module is in the top row and leftmost column of every mapping matrix. The module array sequence shown in Figure F.7 is constant for all mapping matrices.

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8
2.6	2.7	2.8	5.3	5.4	5.5		
1.a	6.1	6.2	5.6	5.7	5.8		
1.b	6.3	6.4	6.5				
	6.6	6.7	6.8				

Figure F.7 — Starting sequence for module placement

NOTE The values a and b depend on the size of the mapping matrix

c) The corner shapes are positioned according to Table F.1 and the appropriate Figures F.3 to F.6. Plotting of the standard symbol character shapes continues, nesting the shapes as illustrated above for symbol characters 2, 5, and 6. The non-standard symbol characters are positioned as per Table F.1. This process results in the mapping matrix being completely covered in symbol characters, most of which are un-numbered.

d) The sequence of symbol characters is determined as follows. Symbol characters are arranged on 45-degree parallel diagonal lines between the lower left and upper right, generally linking through the centres on module 8.

e) The first diagonal line starts with the line through module 8 of symbol character 1; this is module 8 except in the case of the 6 x 28 mapping matrix, where the corner condition, as defined in Figure F.6, determines the values of modules in symbol character 1 (i.e. making the module identified in Figure F.7 as 1.b represent module 1,2). The diagonal line continues through modules 2.8 and 3.8.

f) At this point, the diagonal line crosses the top row boundary. The next diagonal line is started 4 modules to the right in the top row, or in the case of the 8 x 8 mapping matrix, 3 modules right and 1 module down; i.e. the diagonal line is always displaced by 4 modules. Symbol characters are numbered in order, based on the placement path crossing module 8. Thus the next characters are determined by the downward diagonal line crossing modules 4.8, 5.8, 6.8 and so on.

g) As shown in Figure F.8, the placement path continues as diagonal lines four modules to the right (or four modules down, or combinations thereof) from the previous diagonal line. The first, and all odd numbered, diagonal lines map the symbol character sequence from bottom left to top right. The second, and all even numbered, diagonal lines map the symbol character sequence from the top right to the bottom left.

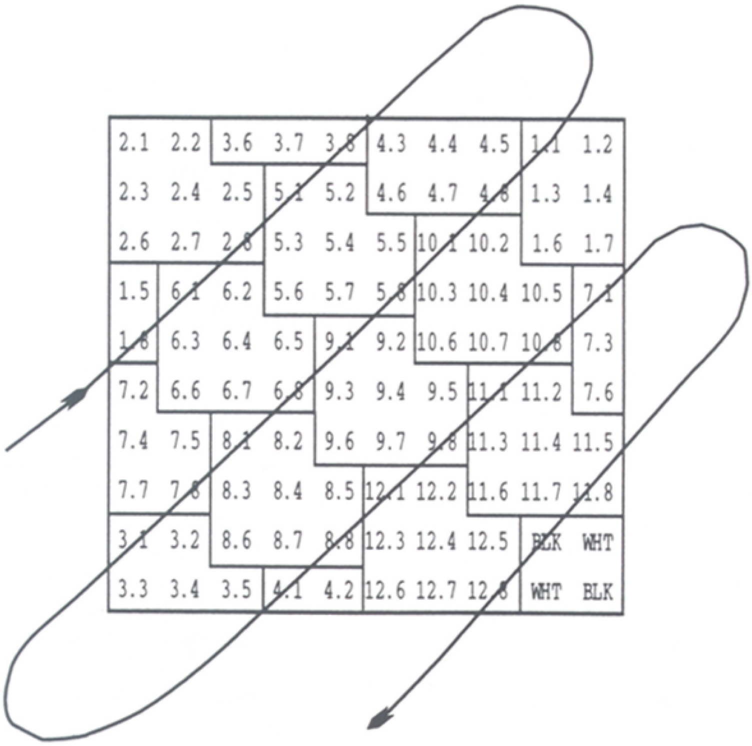


Figure F.8 — Symbol character placement sequence

h) When the placement path encounters a non-standard symbol character shape, which is not completely contained within the boundaries of the mapping matrix, that symbol character is continued on the opposite side of the matrix. This has the effect of numbering the opposite portions of these symbol characters before the placement path crosses that position. For example, in the illustrated mapping matrix (see Figure F.8) the other portions of symbol character 3 and 7 are pre-numbered before the placement path crosses them. Thus the placement path only numbers un-numbered symbol characters. These boundary and corner conditions are specified in Table F.1. This can be seen in Figure F.8 for symbol characters 1, 3, 4, and 7. The corner conditions also affect the numbering sequence. The bottom left corner as illustrated in:

Figure F.3 is numbered immediately before the symbol character above it (see Figures F.11 and F.18 for examples).

Figure F.4 is numbered immediately before the symbol character above it (see Figures F.12 and F.19 for examples).

Figure F.5 is numbered immediately after the symbol character to its right (see Figure F.13 for an example).

Figure F.6 is numbered immediately before the symbol character above it (see Figure F.15 for an example).

The remaining modules of the corner are numbered before the placement path crosses them.

i) The placement procedure continues until all symbol characters are placed, and it ends in the lower right of the mapping matrix. Four sizes of mapping matrix (10 x 10, 14 x 14, 18 x 18, and 22 x 22) have a 2 x 2 area remaining in the bottom right hand corner. The top left and bottom right modules of this area are dark (nominally encoding binary 1). This is illustrated in Figure F.8.

Typical mapping matrices conforming to this procedure are illustrated in F.3. Figures F.9 to F.15 cover respective cases 1 to 7 for boundary placement. Figures F.16 to F.19 are another set of examples for cases 1 to 4. F.1 provides a C language program capable of mapping all encoded bits into the appropriate mapping matrix.

F.3 Symbol character placement examples for ECC 200

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8
2.6	2.7	2.8	5.3	5.4	5.5	1.1	1.2
1.5	6.1	6.2	5.6	5.7	5.8	1.3	1.4
1.8	6.3	6.4	6.5	8.1	8.2	1.6	1.7
7.2	6.6	6.7	6.8	8.3	8.4	8.5	7.1
7.4	7.5	3.1	3.2	8.6	8.7	8.8	7.3
7.7	7.8	3.3	3.4	3.5	4.1	4.2	7.6

Figure F.9 — Codeword placement for square mapping matrix of size 8

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5	1.1	1.2
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8	1.3	1.4
2.6	2.7	2.8	5.3	5.4	5.5	10.1	10.2	1.6	1.7
1.5	6.1	6.2	5.6	5.7	5.8	10.3	10.4	10.5	7.1
1.8	6.3	6.4	6.5	9.1	9.2	10.6	10.7	10.8	7.3
7.2	6.6	6.7	6.8	9.3	9.4	9.5	11.1	11.2	7.6
7.4	7.5	8.1	8.2	9.6	9.7	9.8	11.3	11.4	11.5
7.7	7.8	8.3	8.4	8.5	12.1	12.2	11.6	11.7	11.8
3.1	3.2	8.6	8.7	8.8	12.3	12.4	12.5	BLK	WHT
3.3	3.4	3.5	4.1	4.2	12.6	12.7	12.8	WHT	BLK

Figure F.10 — Codeword placement for square mapping matrix of size 10

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5	13.1	13.2	8.4	8.5
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8	13.3	13.4	13.5	8.6
2.6	2.7	2.8	5.3	5.4	5.5	12.1	12.2	13.6	13.7	13.8	8.7
1.5	6.1	6.2	5.6	5.7	5.8	12.3	12.4	12.5	14.1	14.2	8.8
1.8	6.3	6.4	6.5	11.1	11.2	12.6	12.7	12.8	14.3	14.4	14.5
7.2	6.6	6.7	6.8	11.3	11.4	11.5	15.1	15.2	14.6	14.7	14.8
7.4	7.5	10.1	10.2	11.6	11.7	11.8	15.3	15.4	15.5	1.1	1.2
7.7	7.8	10.3	10.4	10.5	16.1	16.2	15.6	15.7	15.8	1.3	1.4
9.1	9.2	10.6	10.7	10.8	16.3	16.4	16.5	18.1	18.2	1.6	1.7
9.3	9.4	9.5	17.1	17.2	16.6	16.7	16.8	18.3	18.4	18.5	7.1
9.6	9.7	9.8	17.3	17.4	17.5	3.1	3.2	18.6	18.7	18.8	7.3
8.1	8.2	8.3	17.6	17.7	17.8	3.3	3.4	3.5	4.1	4.2	7.6

Figure F.11 — Codeword placement for square mapping matrix of size 12

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5	13.1	13.2	8.4	8.5	8.6	8.7
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8	13.3	13.4	13.5	14.1	14.2	8.8
2.6	2.7	2.8	5.3	5.4	5.5	12.1	12.2	13.6	13.7	13.8	14.3	14.4	14.5
1.5	6.1	6.2	5.6	5.7	5.8	12.3	12.4	12.5	15.1	15.2	14.6	14.7	14.8
1.8	6.3	6.4	6.5	11.1	11.2	12.6	12.7	12.8	15.3	15.4	15.5	1.1	1.2
7.2	6.6	6.7	6.8	11.3	11.4	11.5	16.1	16.2	15.6	15.7	15.8	1.3	1.4
7.4	7.5	10.1	10.2	11.6	11.7	11.8	16.3	16.4	16.5	22.1	22.2	1.6	1.7
7.7	7.8	10.3	10.4	10.5	17.1	17.2	16.6	16.7	16.8	22.3	22.4	22.5	7.1
9.1	9.2	10.6	10.7	10.8	17.3	17.4	17.5	21.1	21.2	22.6	22.7	22.8	7.3
9.3	9.4	9.5	18.1	18.2	17.6	17.7	17.8	21.3	21.4	21.5	23.1	23.2	7.6
9.6	9.7	9.8	18.3	18.4	18.5	20.1	20.2	21.6	21.7	21.8	23.3	23.4	23.5
8.1	19.1	19.2	18.6	18.7	18.8	20.3	20.4	20.5	24.1	24.2	23.6	23.7	23.8
8.2	19.3	19.4	19.5	3.1	3.2	20.6	20.7	20.8	24.3	24.4	24.5	BLK	WHT
8.3	19.6	19.7	19.8	3.3	3.4	3.5	4.1	4.2	24.6	24.7	24.8	WHT	BLK

Figure F.12 — Codeword placement for square mapping matrix of size 14

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5	9.1	9.2	10.6	10.7	10.8	7.3	7.4	7.5
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8	9.3	9.4	9.5	11.1	11.2	7.6	7.7	7.8
2.6	2.7	2.8	5.3	5.4	5.5	8.1	8.2	9.6	9.7	9.8	11.3	11.4	11.5	1.1	1.2
1.5	6.1	6.2	5.6	5.7	5.8	8.3	8.4	8.5	12.1	12.2	11.6	11.7	11.8	1.3	1.4
1.8	6.3	6.4	6.5	3.1	3.2	8.6	8.7	8.8	12.3	12.4	12.5	10.1	10.2	1.6	1.7
7.1	6.6	6.7	6.8	3.3	3.4	3.5	4.1	4.2	12.6	12.7	12.8	10.3	10.4	10.5	7.2

Figure F.13 — Codeword placement for 6 x 16 rectangular mapping matrix

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5	11.1	11.2	12.6	12.7	12.8	13.3	13.4	13.5	21.1	21.2	22.6	22.7	22.8	23.3	23.4	23.5
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8	11.3	11.4	11.5	14.1	14.2	13.6	13.7	13.8	21.3	21.4	21.5	24.1	24.2	23.6	23.7	23.8
2.6	2.7	2.8	5.3	5.4	5.5	10.1	10.2	11.6	11.7	11.8	14.3	14.4	14.5	20.1	20.2	21.6	21.7	21.8	24.3	24.4	24.5	1.1	1.2
1.5	6.1	6.2	5.6	5.7	5.8	10.3	10.4	10.5	15.1	15.2	14.6	14.7	14.8	20.3	20.4	20.5	25.1	25.2	24.6	24.7	24.8	1.3	1.4
1.8	6.3	6.4	6.5	9.1	9.2	10.6	10.7	10.8	15.3	15.4	15.5	19.1	19.2	20.6	20.7	20.8	25.3	25.4	25.5	29.1	29.2	1.6	1.7
7.2	6.6	6.7	6.8	9.3	9.4	9.5	16.1	16.2	15.6	15.7	15.8	19.3	19.4	19.5	26.1	26.2	25.6	25.7	25.8	29.3	29.4	29.5	7.1
7.4	7.5	8.1	8.2	9.6	9.7	9.8	16.3	16.4	16.5	18.1	18.2	19.6	19.7	19.8	26.3	26.4	26.5	28.1	28.2	29.6	29.7	29.8	7.3
7.7	7.8	8.3	8.4	8.5	17.1	17.2	16.6	16.7	16.8	18.3	18.4	18.5	27.1	27.2	26.6	26.7	26.8	28.3	28.4	28.5	30.1	30.2	7.6
3.1	3.2	8.6	8.7	8.8	17.3	17.4	17.5	12.1	12.2	18.6	18.7	18.8	27.3	27.4	27.5	22.1	22.2	28.6	28.7	28.8	30.3	30.4	30.5
3.3	3.4	3.5	4.1	4.2	17.6	17.7	17.8	12.3	12.4	12.5	13.1	13.2	27.6	27.7	27.8	22.3	22.4	22.5	23.1	23.2	30.6	30.7	30.8

Figure F.14 — Codeword placement for 10 x 24 rectangular mapping matrix

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5	8.1	8.2	9.6	9.7	9.8	10.3	10.4	10.5	14.1	14.2	15.6	15.7	15.8	16.3	16.4	16.5	20.1	20.2	1.4	1.5
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8	8.3	8.4	8.5	11.1	11.2	10.6	10.7	10.8	14.3	14.4	14.5	17.1	17.2	16.6	16.7	16.8	20.3	20.4	20.5	1.6
2.6	2.7	2.8	5.3	5.4	5.5	7.1	7.2	8.6	8.7	8.8	11.3	11.4	11.5	13.1	13.2	14.6	14.7	14.8	17.3	17.4	17.5	19.1	19.2	20.6	20.7	20.8	1.7
1.1	6.1	6.2	5.6	5.7	5.8	7.3	7.4	7.5	12.1	12.2	11.6	11.7	11.8	13.3	13.4	13.5	18.1	18.2	17.6	17.7	17.8	19.3	19.4	19.5	21.1	21.2	1.8
1.2	6.3	6.4	6.5	3.1	3.2	7.6	7.7	7.8	12.3	12.4	12.5	9.1	9.2	13.6	13.7	13.8	18.3	18.4	18.5	15.1	15.2	19.6	19.7	19.8	21.3	21.4	21.5
1.3	6.6	6.7	6.8	3.3	3.4	3.5	4.1	4.2	12.6	12.7	12.8	9.3	9.4	9.5	10.1	10.2	18.6	18.7	18.8	15.3	15.4	15.5	16.1	16.2	21.6	21.7	21.8

Figure F.15 — Codeword placement for 6 x 28 rectangular mapping matrix

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5	13.1	13.2	14.6	14.7	14.8	15.3	15.4	15.5
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8	13.3	13.4	13.5	16.1	16.2	15.6	15.7	15.8
2.6	2.7	2.8	5.3	5.4	5.5	12.1	12.2	13.6	13.7	13.8	16.3	16.4	16.5	1.1	1.2
1.5	6.1	6.2	5.6	5.7	5.8	12.3	12.4	12.5	17.1	17.2	16.6	16.7	16.8	1.3	1.4
1.8	6.3	6.4	6.5	11.1	11.2	12.6	12.7	12.8	17.3	17.4	17.5	27.1	27.2	1.6	1.7
7.2	6.6	6.7	6.8	11.3	11.4	11.5	18.1	18.2	17.6	17.7	17.8	27.3	27.4	27.5	7.1
7.4	7.5	10.1	10.2	11.6	11.7	11.8	18.3	18.4	18.5	26.1	26.2	27.6	27.7	27.8	7.3
7.7	7.8	10.3	10.4	10.5	19.1	19.2	18.6	18.7	18.8	26.3	26.4	26.5	28.1	28.2	7.6
9.1	9.2	10.6	10.7	10.8	19.3	19.4	19.5	25.1	25.2	26.6	26.7	26.8	28.3	28.4	28.5
9.3	9.4	9.5	20.1	20.2	19.6	19.7	19.8	25.3	25.4	25.5	29.1	29.2	28.6	28.7	28.8
9.6	9.7	9.8	20.3	20.4	20.5	24.1	24.2	25.6	25.7	25.8	29.3	29.4	29.5	8.1	8.2
8.5	21.1	21.2	20.6	20.7	20.8	24.3	24.4	24.5	30.1	30.2	29.6	29.7	29.8	8.3	8.4
8.8	21.3	21.4	21.5	23.1	23.2	24.6	24.7	24.8	30.3	30.4	30.5	32.1	32.2	8.6	8.7
22.2	21.6	21.7	21.8	23.3	23.4	23.5	31.1	31.2	30.6	30.7	30.8	32.3	32.4	32.5	22.1
22.4	22.5	3.1	3.2	23.6	23.7	23.8	31.3	31.4	31.5	14.1	14.2	32.6	32.7	32.8	22.3
22.7	22.8	3.3	3.4	3.5	4.1	4.2	31.6	31.7	31.8	14.3	14.4	14.5	15.1	15.2	22.6

Figure F.16 — Codeword placement for square mapping matrix of size 16

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5	13.1	13.2	14.6	14.7	14.8	15.3	15.4	15.5	1.1	1.2
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8	13.3	13.4	13.5	16.1	16.2	15.6	15.7	15.8	1.3	1.4
2.6	2.7	2.8	5.3	5.4	5.5	12.1	12.2	13.6	13.7	13.8	16.3	16.4	16.5	29.1	29.2	1.6	1.7
1.5	6.1	6.2	5.6	5.7	5.8	12.3	12.4	12.5	17.1	17.2	16.6	16.7	16.8	29.3	29.4	29.5	7.1
1.8	6.3	6.4	6.5	11.1	11.2	12.6	12.7	12.8	17.3	17.4	17.5	28.1	28.2	29.6	29.7	29.8	7.3
7.2	6.6	6.7	6.8	11.3	11.4	11.5	18.1	18.2	17.6	17.7	17.8	28.3	28.4	28.5	30.1	30.2	7.6
7.4	7.5	10.1	10.2	11.6	11.7	11.8	18.3	18.4	18.5	27.1	27.2	28.6	28.7	28.8	30.3	30.4	30.5
7.7	7.8	10.3	10.4	10.5	19.1	19.2	18.6	18.7	18.8	27.3	27.4	27.5	31.1	31.2	30.6	30.7	30.8
9.1	9.2	10.6	10.7	10.8	19.3	19.4	19.5	26.1	26.2	27.6	27.7	27.8	31.3	31.4	31.5	8.1	8.2
9.3	9.4	9.5	20.1	20.2	19.6	19.7	19.8	26.3	26.4	26.5	32.1	32.2	31.6	31.7	31.8	8.3	8.4
9.6	9.7	9.8	20.3	20.4	20.5	25.1	25.2	26.6	26.7	26.8	32.3	32.4	32.5	38.1	38.2	8.6	8.7
8.5	21.1	21.2	20.6	20.7	20.8	25.3	25.4	25.5	33.1	33.2	32.6	32.7	32.8	38.3	38.4	38.5	22.1
8.8	21.3	21.4	21.5	24.1	24.2	25.6	25.7	25.8	33.3	33.4	33.5	37.1	37.2	38.6	38.7	38.8	22.3
22.2	21.6	21.7	21.8	24.3	24.4	24.5	34.1	34.2	33.6	33.7	33.8	37.3	37.4	37.5	39.1	39.2	22.6
22.4	22.5	23.1	23.2	24.6	24.7	24.8	34.3	34.4	34.5	36.1	36.2	37.6	37.7	37.8	39.3	39.4	39.5
22.7	22.8	23.3	23.4	23.5	35.1	35.2	34.6	34.7	34.8	36.3	36.4	36.5	40.1	40.2	39.6	39.7	39.8
3.1	3.2	23.6	23.7	23.8	35.3	35.4	35.5	14.1	14.2	36.6	36.7	36.8	40.3	40.4	40.5	BLK	WHT
3.3	3.4	3.5	4.1	4.2	35.6	35.7	35.8	14.3	14.4	14.5	15.1	15.2	40.6	40.7	40.8	WHT	BLK

Figure F.17 — Codeword placement for square mapping matrix of size 18

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5	13.1	13.2	14.6	14.7	14.8	15.3	15.4	15.5	32.1	32.2	23.4	23.5
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8	13.3	13.4	13.5	16.1	16.2	15.6	15.7	15.8	32.3	32.4	32.5	23.6
2.6	2.7	2.8	5.3	5.4	5.5	12.1	12.2	13.6	13.7	13.8	16.3	16.4	16.5	31.1	31.2	32.6	32.7	32.8	23.7
1.5	6.1	6.2	5.6	5.7	5.8	12.3	12.4	12.5	17.1	17.2	16.6	16.7	16.8	31.3	31.4	31.5	33.1	33.2	23.8
1.8	6.3	6.4	6.5	11.1	11.2	12.6	12.7	12.8	17.3	17.4	17.5	30.1	30.2	31.6	31.7	31.8	33.3	33.4	33.5
7.2	6.6	6.7	6.8	11.3	11.4	11.5	18.1	18.2	17.6	17.7	17.8	30.3	30.4	30.5	34.1	34.2	33.6	33.7	33.8
7.4	7.5	10.1	10.2	11.6	11.7	11.8	18.3	18.4	18.5	29.1	29.2	30.6	30.7	30.8	34.3	34.4	34.5	1.1	1.2
7.7	7.8	10.3	10.4	10.5	19.1	19.2	18.6	18.7	18.8	29.3	29.4	29.5	35.1	35.2	34.6	34.7	34.8	1.3	1.4
9.1	9.2	10.6	10.7	10.8	19.3	19.4	19.5	28.1	28.2	29.6	29.7	29.8	35.3	35.4	35.5	45.1	45.2	1.6	1.7
9.3	9.4	9.5	20.1	20.2	19.6	19.7	19.8	28.3	28.4	28.5	36.1	36.2	35.6	35.7	35.8	45.3	45.4	45.5	7.1
9.6	9.7	9.8	20.3	20.4	20.5	27.1	27.2	28.6	28.7	28.8	36.3	36.4	36.5	44.1	44.2	45.6	45.7	45.8	7.3
8.5	21.1	21.2	20.6	20.7	20.8	27.3	27.4	27.5	37.1	37.2	36.6	36.7	36.8	44.3	44.4	44.5	46.1	46.2	7.6
8.8	21.3	21.4	21.5	26.1	26.2	27.6	27.7	27.8	37.3	37.4	37.5	43.1	43.2	44.6	44.7	44.8	46.3	46.4	46.5
22.2	21.6	21.7	21.8	26.3	26.4	26.5	38.1	38.2	37.6	37.7	37.8	43.3	43.4	43.5	47.1	47.2	46.6	46.7	46.8
22.4	22.5	25.1	25.2	26.6	26.7	26.8	38.3	38.4	38.5	42.1	42.2	43.6	43.7	43.8	47.3	47.4	47.5	8.1	8.2
22.7	22.8	25.3	25.4	25.5	39.1	39.2	38.6	38.7	38.8	42.3	42.4	42.5	48.1	48.2	47.6	47.7	47.8	8.3	8.4
24.1	24.2	25.6	25.7	25.8	39.3	39.4	39.5	41.1	41.2	42.6	42.7	42.8	48.3	48.4	48.5	50.1	50.2	8.6	8.7
24.3	24.4	24.5	40.1	40.2	39.6	39.7	39.8	41.3	41.4	41.5	49.1	49.2	48.6	48.7	48.8	50.3	50.4	50.5	22.1
24.6	24.7	24.8	40.3	40.4	40.5	3.1	3.2	41.6	41.7	41.8	49.3	49.4	49.5	14.1	14.2	50.6	50.7	50.8	22.3
23.1	23.2	23.3	40.6	40.7	40.8	3.3	3.4	3.5	4.1	4.2	49.6	49.7	49.8	14.3	14.4	14.5	15.1	15.2	22.6

Figure F.18 — Codeword placement for square mapping matrix of size 20

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5	13.1	13.2	14.6	14.7	14.8	15.3	15.4	15.5	32.1	32.2	23.4	23.5	23.6	23.7
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8	13.3	13.4	13.5	16.1	16.2	15.6	15.7	15.8	32.3	32.4	32.5	33.1	33.2	23.8
2.6	2.7	2.8	5.3	5.4	5.5	12.1	12.2	13.6	13.7	13.8	16.3	16.4	16.5	31.1	31.2	32.6	32.7	32.8	33.3	33.4	33.5
1.5	6.1	6.2	5.6	5.7	5.8	12.3	12.4	12.5	17.1	17.2	16.6	16.7	16.8	31.3	31.4	31.5	34.1	34.2	33.6	33.7	33.8
1.8	6.3	6.4	6.5	11.1	11.2	12.6	12.7	12.8	17.3	17.4	17.5	30.1	30.2	31.6	31.7	31.8	34.3	34.4	34.5	1.1	1.2
7.2	6.6	6.7	6.8	11.3	11.4	11.5	18.1	18.2	17.6	17.7	17.8	30.3	30.4	30.5	35.1	35.2	34.6	34.7	34.8	1.3	1.4
7.4	7.5	10.1	10.2	11.6	11.7	11.8	18.3	18.4	18.5	29.1	29.2	30.6	30.7	30.8	35.3	35.4	35.5	49.1	49.2	1.6	1.7
7.7	7.8	10.3	10.4	10.5	19.1	19.2	18.6	18.7	18.8	29.3	29.4	29.5	36.1	36.2	35.6	35.7	35.8	49.3	49.4	49.5	7.1
9.1	9.2	10.6	10.7	10.8	19.3	19.4	19.5	28.1	28.2	29.6	29.7	29.8	36.3	36.4	36.5	48.1	48.2	49.6	49.7	49.8	7.3
9.3	9.4	9.5	20.1	20.2	19.6	19.7	19.8	28.3	28.4	28.5	37.1	37.2	36.6	36.7	36.8	48.3	48.4	48.5	50.1	50.2	7.6
9.6	9.7	9.8	20.3	20.4	20.5	27.1	27.2	28.6	28.7	28.8	37.3	37.4	37.5	47.1	47.2	48.6	48.7	48.8	50.3	50.4	50.5
8.5	21.1	21.2	20.6	20.7	20.8	27.3	27.4	27.5	38.1	38.2	37.6	37.7	37.8	47.3	47.4	47.5	51.1	51.2	50.6	50.7	50.8
8.8	21.3	21.4	21.5	26.1	26.2	27.6	27.7	27.8	38.3	38.4	38.5	46.1	46.2	47.6	47.7	47.8	51.3	51.4	51.5	8.1	8.2
22.2	21.6	21.7	21.8	26.3	26.4	26.5	39.1	39.2	38.6	38.7	38.8	46.3	46.4	46.5	52.1	52.2	51.6	51.7	51.8	8.3	8.4
22.4	22.5	25.1	25.2	26.6	26.7	26.8	39.3	39.4	39.5	45.1	45.2	46.6	46.7	46.8	52.3	52.4	52.5	58.1	58.2	8.6	8.7
22.7	22.8	25.3	25.4	25.5	40.1	40.2	39.6	39.7	39.8	45.3	45.4	45.5	53.1	53.2	52.6	52.7	52.8	58.3	58.4	58.5	22.1
24.1	24.2	25.6	25.7	25.8	40.3	40.4	40.5	44.1	44.2	45.6	45.7	45.8	53.3	53.4	53.5	57.1	57.2	58.6	58.7	58.8	22.3
24.3	24.4	24.5	41.1	41.2	40.6	40.7	40.8	44.3	44.4	44.5	54.1	54.2	53.6	53.7	53.8	57.3	57.4	57.5	59.1	59.2	22.6
24.6	24.7	24.8	41.3	41.4	41.5	43.1	43.2	44.6	44.7	44.8	54.3	54.4	54.5	56.1	56.2	57.6	57.7	57.8	59.3	59.4	59.5
23.1	42.1	42.2	41.6	41.7	41.8	43.3	43.4	43.5	55.1	55.2	54.6	54.7	54.8	56.3	56.4	56.5	60.1	60.2	59.6	59.7	59.8
23.2	42.3	42.4	42.5	3.1	3.2	43.6	43.7	43.8	55.3	55.4	55.5	14.1	14.2	56.6	56.7	56.8	60.3	60.4	60.5	BLK	WHT
23.3	42.6	42.7	42.8	3.3	3.4	3.5	4.1	4.2	55.6	55.7	55.8	14.3	14.4	14.5	15.1	15.2	60.6	60.7	60.8	WHT	BLK

Figure F.19 — Codeword placement for square mapping matrix of size 22

Annex G
(normative)

ECC 000 - 140 symbol attributes

Table G.1 — ECC 000

Symbol size[a]		Data region size		Numeric capacity	Alphanum capacity	8-bit byte capacity	% of codewords used for error correction	% correctable
Row	Col	Row	Col					
9	9	7	7	3	2	1	0,0	0,0
11	11	9	9	12	8	5	0,0	0,0
13	13	11	11	24	16	10	0,0	0,0
15	15	13	13	37	25	16	0,0	0,0
17	17	15	15	53	35	23	0,0	0,0
19	19	17	17	72	48	31	0,0	0,0
21	21	19	19	92	61	40	0,0	0,0
23	23	21	21	115	76	50	0,0	0,0
25	25	23	23	140	93	61	0,0	0,0
27	27	25	25	168	112	73	0,0	0,0
29	29	27	27	197	131	86	0,0	0,0
31	31	29	29	229	153	100	0,0	0,0
33	33	31	31	264	176	115	0,0	0,0
35	35	33	33	300	200	131	0,0	0,0
37	37	35	35	339	226	148	0,0	0,0
39	39	37	37	380	253	166	0,0	0,0
41	41	39	39	424	282	185	0,0	0,0
43	43	41	41	469	313	205	0,0	0,0
45	45	43	43	500	345	226	0,0	0,0
47	47	45	45	560	378	248	0,0	0,0
49	49	47	47	596	413	271	0,0	0,0

[a] excluding quiet zones

Table G.2 — ECC 050

Symbol size[a]		Data region size		Numeric capacity	Alphanum. capacity	8-bit byte capacity	% of codewords used for error correction	% correctable
Row	Col	Row	Col					
11	11	9	9	1	1	0[b]	25,0	2,8
13	13	11	11	10	6	4	25,0	2,8
15	15	13	13	20	13	9	25,0	2,8
17	17	15	15	32	21	14	25,0	2,8
19	19	17	17	46	30	20	25,0	2,8
21	21	19	19	61	41	27	25,0	2,8
23	23	21	21	78	52	34	25,0	2,8
25	25	23	23	97	65	42	25,0	2,8
27	27	25	25	118	78	51	25,0	2,8
29	29	27	27	140	93	61	25,0	2,8
31	31	29	29	164	109	72	25,0	2,8
33	33	31	31	190	126	83	25,0	2,8
35	35	33	33	217	145	95	25,0	2,8
37	37	35	35	246	164	108	25,0	2,8
39	39	37	37	277	185	121	25,0	2,8
41	41	39	39	310	206	135	25,0	2,8
43	43	41	41	344	229	150	25,0	2,8
45	45	43	43	380	253	166	25,0	2,8
47	47	45	45	418	278	183	25,0	2,8
49	49	47	47	457	305	200	25,0	2,8

[a] excluding quiet zone

[b] this combination is not possible

Table G.3 — ECC 080

Symbol size[a]		Data region size		Numeric capacity	Alphanum. capacity	8-bit byte capacity	% of codewords used for error correction	% correctable
Row	Col	Row	Col					
13	13	11	11	4	3	2	33,3	5,5
15	15	13	13	13	9	6	33,3	5,5
17	17	15	15	24	16	10	33,3	5,5
19	19	17	17	36	24	16	33,3	5,5
21	21	19	19	50	33	22	33,3	5,5
23	23	21	21	65	43	28	33,3	5,5
25	25	23	23	82	54	36	33,3	5,5
27	27	25	25	100	67	44	33,3	5,5
29	29	27	27	120	80	52	33,3	5,5
31	31	29	29	141	94	62	33,3	5,5
33	33	31	31	164	109	72	33,3	5,5
35	35	33	33	188	125	82	33,3	5,5
37	37	35	35	214	143	94	33,3	5,5
39	39	37	37	242	161	106	33,3	5,5
41	41	39	39	270	180	118	33,3	5,5
43	43	41	41	301	201	132	33,3	5,5
45	45	43	43	333	222	146	33,3	5,5
47	47	45	45	366	244	160	33,3	5,5
49	49	47	47	402	268	176	33,3	5,5

a excluding quiet zones

Table G.4 — ECC 100

Symbol size[a]		Data region size		Numeric capacity	Alphanum. capacity	8-bit byte capacity	% of codewords used for error correction	% correctable
Row	Col	Row	Col					
13	13	11	11	1	1	0[b]	50,0	12,6
15	15	13	13	8	5	3	50,0	12,6
17	17	15	15	16	11	7	50,0	12,6
19	19	17	17	25	17	11	50,0	12,6
21	21	19	19	36	24	15	50,0	12,6
23	23	21	21	47	31	20	50,0	12,6
25	25	23	23	60	40	26	50,0	12,6
27	27	25	25	73	49	32	50,0	12,6
29	29	27	27	88	59	38	50,0	12,6
31	31	29	29	104	69	45	50,0	12,6
33	33	31	31	121	81	53	50,0	12,6
35	35	33	33	140	93	61	50,0	12,6
37	37	35	35	159	106	69	50,0	12,6
39	39	37	37	180	120	78	50,0	12,6
41	41	39	39	201	134	88	50,0	12,6
43	43	41	41	224	149	98	50,0	12,6
45	45	43	43	248	165	108	50,0	12,6
47	47	45	45	273	182	119	50,0	12,6
49	49	47	47	300	200	131	50,0	12,6

[a] excluding quiet zones

[b] this combination is not possible

Table G.5 — ECC 140

Symbol size[a]		Data region size		Numeric capacity	Alphanum. capacity	8-bit byte capacity	% of codewords used for error correction	% correctable
Row	Col	Row	Col					
17	17	15	15	2	1	1	75,0	25,0
19	19	17	17	6	4	3	75,0	25,0
21	21	19	19	12	8	5	75,0	25,0
23	23	21	21	17	11	7	75,0	25,0
25	25	23	23	24	16	10	75,0	25,0
27	27	25	25	30	20	13	75,0	25,0
29	29	27	27	38	25	16	75,0	25,0
31	31	29	29	46	30	20	75,0	25,0
33	33	31	31	54	36	24	75,0	25,0
35	35	33	33	64	42	28	75,0	25,0
37	37	35	35	73	49	32	75,0	25,0
39	39	37	37	84	56	36	75,0	25,0
41	41	39	39	94	63	41	75,0	25,0
43	43	41	41	106	70	46	75,0	25,0
45	45	43	43	118	78	51	75,0	25,0
47	47	45	45	130	87	57	75,0	25,0
49	49	47	47	144	96	63	75,0	25,0

[a] excluding quiet zones

Annex H
(normative)

ECC 000 - 140 data module placement grids

Table H.1 — 7 x 7 data

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

Table H.2 — 9 x 9 data

2	19	55	10	46	28	64	73	1
62	17	53	35	71	8	80	44	26
49	31	67	4	76	40	22	58	13
69	6	78	42	24	60	15	51	33
74	38	20	56	11	47	29	65	37
25	61	16	52	34	70	7	79	43
12	48	30	66	63	75	39	21	57
32	68	5	77	41	23	59	14	50
0	72	36	18	54	9	45	27	3

Table H.3 — 11 x 11 data

2	26	114	70	15	103	59	37	81	4	1
117	73	18	106	62	40	84	7	95	51	29
12	100	56	34	78	92	89	45	23	111	67
65	43	87	10	98	54	32	120	76	21	109
82	5	93	49	27	115	71	16	104	60	38
96	52	30	118	74	19	107	63	41	85	8
24	112	68	13	101	57	35	79	48	90	46
75	20	108	64	42	86	9	97	53	31	119
102	58	36	80	77	91	47	25	113	69	14
39	83	6	94	50	28	116	72	17	105	61
0	88	44	22	110	66	11	99	55	33	3

Table H.4 — 13 x 13 data

2	159	29	133	81	16	120	68	42	146	94	91	1
37	141	89	24	128	76	50	154	102	11	115	63	167
83	18	122	70	44	148	96	5	109	57	161	31	135
125	73	47	151	99	8	112	60	164	34	138	86	21
40	144	92	107	105	53	157	27	131	79	14	118	66
103	12	116	64	168	38	142	90	25	129	77	51	155
110	58	162	32	136	84	19	123	71	45	149	97	6
165	35	139	87	22	126	74	48	152	100	9	113	61
132	80	15	119	67	41	145	93	55	106	54	158	28
23	127	75	49	153	101	10	114	62	166	36	140	88
69	43	147	95	4	108	56	160	30	134	82	17	121
150	98	7	111	59	163	33	137	85	20	124	72	46
0	104	52	156	26	130	78	13	117	65	39	143	3

Table H.5 — 15 x 15 data

2	187	37	157	97	217	22	142	82	202	52	172	112	7	1
41	161	101	221	26	146	86	206	56	176	116	11	131	71	191
93	213	18	138	78	198	48	168	108	105	123	63	183	33	153
28	148	88	208	58	178	118	13	133	73	193	43	163	103	223
80	200	50	170	110	5	125	65	185	35	155	95	215	20	140
54	174	114	9	129	69	189	39	159	99	219	24	144	84	204
106	127	121	61	181	31	151	91	211	16	136	76	196	46	166
134	74	194	44	164	104	224	29	149	89	209	59	179	119	14
186	36	156	96	216	21	141	81	201	51	171	111	6	126	66
160	100	220	25	145	85	205	55	175	115	10	130	70	190	40
212	17	137	77	197	47	167	107	67	122	62	182	32	152	92
147	87	207	57	177	117	12	132	72	192	42	162	102	222	27
199	49	169	109	4	124	64	184	34	154	94	214	19	139	79
173	113	8	128	68	188	38	158	98	218	23	143	83	203	53
0	120	60	180	30	150	90	210	15	135	75	195	45	165	3

Table H.6 — 17 x 17 data

2	69	205	35	171	103	239	18	154	86	222	52	188	120	256	273	1
220	50	186	118	254	33	169	101	237	67	203	135	271	16	288	152	84
178	110	246	25	161	93	229	59	195	127	263	8	280	144	76	212	42
250	29	165	97	233	63	199	131	267	12	284	148	80	216	46	182	114
157	89	225	55	191	123	259	4	276	140	72	208	38	174	106	242	21
235	65	201	133	269	14	286	150	82	218	48	184	116	252	31	167	99
193	125	261	6	278	142	74	210	40	176	108	244	23	159	91	227	57
265	10	282	146	78	214	44	180	112	248	27	163	95	231	61	197	129
274	138	70	206	36	172	104	240	19	155	87	223	53	189	121	257	137
83	219	49	185	117	253	32	168	100	236	66	202	134	270	15	287	151
41	177	109	245	24	160	92	228	58	194	126	262	7	279	143	75	211
113	249	28	164	96	232	62	198	130	266	11	283	147	79	215	45	181
20	156	88	224	54	190	122	258	255	275	139	71	207	37	173	105	241
98	234	64	200	132	268	13	285	149	81	217	47	183	115	251	30	166
56	192	124	260	5	277	141	73	209	39	175	107	243	22	158	90	226
128	264	9	281	145	77	213	43	179	111	247	26	162	94	230	60	196
0	272	136	68	204	34	170	102	238	17	153	85	221	51	187	119	3

Table H.7 — 19 x 19 data

2	82	234	44	348	196	120	272	25	329	177	101	253	63	215	139	291	6	1
239	49	353	201	125	277	30	334	182	106	258	68	220	144	296	11	315	163	87
343	191	115	267	20	324	172	96	248	58	210	134	286	310	305	153	77	229	39
132	284	37	341	189	113	265	75	227	151	303	18	322	170	94	246	56	360	208
28	332	180	104	256	66	218	142	294	9	313	161	85	237	47	351	199	123	275
185	109	261	71	223	147	299	14	318	166	90	242	52	356	204	128	280	33	337
251	61	213	137	289	4	308	156	80	232	42	346	194	118	270	23	327	175	99
225	149	301	16	320	168	92	244	54	358	206	130	282	35	339	187	111	263	73
292	7	311	159	83	235	45	349	197	121	273	26	330	178	102	254	64	216	140
316	164	88	240	50	354	202	126	278	31	335	183	107	259	69	221	145	297	12
78	230	40	344	192	116	268	21	325	173	97	249	59	211	135	287	158	306	154
55	359	207	131	283	36	340	188	112	264	74	226	150	302	17	321	169	93	245
198	122	274	27	331	179	103	255	65	217	141	293	8	312	160	84	236	46	350
279	32	336	184	108	260	70	222	146	298	13	317	165	89	241	51	355	203	127
326	174	98	250	60	212	136	288	285	307	155	79	231	41	345	193	117	269	22
110	262	72	224	148	300	15	319	167	91	243	53	357	205	129	281	34	338	186
62	214	138	290	5	309	157	81	233	43	347	195	119	271	24	328	176	100	252
143	295	10	314	162	86	238	48	352	200	124	276	29	333	181	105	257	67	219
0	304	152	76	228	38	342	190	114	266	19	323	171	95	247	57	209	133	3

Table H.8 — 21 x 21 data

2	88	424	256	46	382	214	130	298	25	361	193	109	277	67	403	235	151	319	4	1
437	269	59	395	227	143	311	38	374	206	122	290	80	416	248	164	332	17	353	185	101
49	385	217	133	301	28	364	196	112	280	70	406	238	154	322	7	343	175	91	427	259
222	138	306	33	369	201	117	285	75	411	243	159	327	12	348	180	96	432	264	54	390
295	22	358	190	106	274	64	400	232	148	316	340	337	169	85	421	253	43	379	211	127
377	209	125	293	83	419	251	167	335	20	356	188	104	440	272	62	398	230	146	314	41
115	283	73	409	241	157	325	10	346	178	94	430	262	52	388	220	136	304	31	367	199
78	414	246	162	330	15	351	183	99	435	267	57	393	225	141	309	36	372	204	120	288
236	152	320	5	341	173	89	425	257	47	383	215	131	299	26	362	194	110	278	68	404
333	18	354	186	102	438	270	60	396	228	144	312	39	375	207	123	291	81	417	249	165
344	176	92	428	260	50	386	218	134	302	29	365	197	113	281	71	407	239	155	323	8
97	433	265	55	391	223	139	307	34	370	202	118	286	76	412	244	160	328	13	349	181
254	44	380	212	128	296	23	359	191	107	275	65	401	233	149	317	172	338	170	86	422
397	229	145	313	40	376	208	124	292	82	418	250	166	334	19	355	187	103	439	271	61
135	303	30	366	198	114	282	72	408	240	156	324	9	345	177	93	429	261	51	387	219
35	371	203	119	287	77	413	245	161	329	14	350	182	98	434	266	56	392	224	140	308
192	108	276	66	402	234	150	318	315	339	171	87	423	255	45	381	213	129	297	24	360
289	79	415	247	163	331	16	352	184	100	436	268	58	394	226	142	310	37	373	205	121
405	237	153	321	6	342	174	90	426	258	48	384	216	132	300	27	363	195	111	279	69
158	326	11	347	179	95	431	263	53	389	221	137	305	32	368	200	116	284	74	410	242
0	336	168	84	420	252	42	378	210	126	294	21	357	189	105	273	63	399	231	147	3

Table H.9 — 23 x 23 data

2	102	470	286	56	424	240	148	516	332	33	401	217	125	493	309	79	447	263	171	355	10	1
476	292	62	430	246	154	522	338	39	407	223	131	499	315	85	453	269	177	361	16	384	200	108
50	418	234	142	510	326	27	395	211	119	487	303	73	441	257	165	349	4	372	188	96	464	280
249	157	525	341	42	410	226	134	502	318	88	456	272	180	364	19	387	203	111	479	295	65	433
513	329	30	398	214	122	490	306	76	444	260	168	352	7	375	191	99	467	283	53	421	237	145
36	404	220	128	496	312	82	450	266	174	358	13	381	197	105	473	289	59	427	243	151	519	335
208	116	484	300	70	438	254	162	346	378	369	185	93	461	277	47	415	231	139	507	323	24	392
505	321	91	459	275	183	367	22	390	206	114	482	298	68	436	252	160	528	344	45	413	229	137
80	448	264	172	356	11	379	195	103	471	287	57	425	241	149	517	333	34	402	218	126	494	310
270	178	362	17	385	201	109	477	293	63	431	247	155	523	339	40	408	224	132	500	316	86	454
350	5	373	189	97	465	281	51	419	235	143	511	327	28	396	212	120	488	304	74	442	258	166
388	204	112	480	296	66	434	250	158	526	342	43	411	227	135	503	319	89	457	273	181	365	20
100	468	284	54	422	238	146	514	330	31	399	215	123	491	307	77	445	261	169	353	8	376	192
290	60	428	244	152	520	336	37	405	221	129	497	313	83	451	267	175	359	14	382	198	106	474
416	232	140	508	324	25	393	209	117	485	301	71	439	255	163	347	194	370	186	94	462	278	48
159	527	343	44	412	228	136	504	320	90	458	274	182	366	21	389	205	113	481	297	67	435	251
331	32	400	216	124	492	308	78	446	262	170	354	9	377	193	101	469	285	55	423	239	147	515
406	222	130	498	314	84	452	268	176	360	15	383	199	107	475	291	61	429	245	153	521	337	38
118	486	302	72	440	256	164	348	345	371	187	95	463	279	49	417	233	141	509	325	26	394	210
317	87	455	271	179	363	18	386	202	110	478	294	64	432	248	156	524	340	41	409	225	133	501
443	259	167	351	6	374	190	98	466	282	52	420	236	144	512	328	29	397	213	121	489	305	75
173	357	12	380	196	104	472	288	58	426	242	150	518	334	35	403	219	127	495	311	81	449	265
0	368	184	92	460	276	46	414	230	138	506	322	23	391	207	115	483	299	69	437	253	161	3

Table H.10 — 25 x 25 data

2	603	103	503	303	53	453	253	153	553	353	28	428	228	128	528	328	78	478	278	178	578	378	375	1
123	523	323	73	473	273	173	573	373	48	448	248	148	548	348	98	498	298	198	598	398	23	423	223	623
311	61	461	261	161	561	361	36	436	236	136	536	336	86	486	286	186	586	386	11	411	211	611	111	511
467	267	167	567	367	42	442	242	142	542	342	92	492	292	192	592	392	17	417	217	617	117	517	317	67
155	555	355	30	430	230	130	530	330	80	480	280	180	580	380	5	405	205	605	105	505	305	55	455	255
370	45	445	245	145	545	345	95	495	295	195	595	395	20	420	220	620	120	520	320	70	470	270	170	570
433	233	133	533	333	83	483	283	183	583	383	8	408	208	608	108	508	308	58	458	258	158	558	358	33
139	539	339	89	489	289	189	589	389	14	414	214	614	114	514	314	64	464	264	164	564	364	39	439	239
326	76	476	276	176	576	376	403	401	201	601	101	501	301	51	451	251	151	551	351	26	426	226	126	526
499	299	199	599	399	24	424	224	624	124	524	324	74	474	274	174	574	374	49	449	249	149	549	349	99
187	587	387	12	412	212	612	112	512	312	62	462	262	162	562	362	37	437	237	137	537	337	87	487	287
393	18	418	218	618	118	518	318	68	468	268	168	568	368	43	443	243	143	543	343	93	493	293	193	593
406	206	606	106	506	306	56	456	256	156	556	356	31	431	231	131	531	331	81	481	281	181	581	381	6
621	121	521	321	71	471	271	171	571	371	46	446	246	146	546	346	96	496	296	196	596	396	21	421	221
509	309	59	459	259	159	559	359	34	434	234	134	534	334	84	484	284	184	584	384	9	409	209	609	109
65	465	265	165	565	365	40	440	240	140	540	340	90	490	290	190	590	390	15	415	215	615	115	515	315
252	152	552	352	27	427	227	127	527	327	77	477	277	177	577	377	203	402	202	602	102	502	302	52	452
572	372	47	447	247	147	547	347	97	497	297	197	597	397	22	422	222	622	122	522	322	72	472	272	172
35	435	235	135	535	335	85	485	285	185	585	385	10	410	210	610	110	510	310	60	460	260	160	560	360
241	141	541	341	91	491	291	191	591	391	16	416	216	616	116	516	316	66	466	266	166	566	366	41	441
529	329	79	479	279	179	579	379	4	404	204	604	104	504	304	54	454	254	154	554	354	29	429	229	129
94	494	294	194	594	394	19	419	219	619	119	519	319	69	469	269	169	569	369	44	444	244	144	544	344
282	182	582	382	7	407	207	607	107	507	307	57	457	257	157	557	357	32	432	232	132	532	332	82	482
588	388	13	413	213	613	113	513	313	63	463	263	163	563	363	38	438	238	138	538	338	88	488	288	188
0	400	200	600	100	500	300	50	450	250	150	550	350	25	425	225	125	525	325	75	475	275	175	575	3

Table H.11 — 27 x 27 data

2	658	118	550	334	64	496	280	712	172	604	388	37	469	253	685	145	577	361	91	523	307	199	631	415	10	1
125	557	341	71	503	287	719	179	611	395	44	476	260	692	152	584	368	98	530	314	206	638	422	17	449	233	665
327	57	489	273	705	165	597	381	30	462	246	678	138	570	354	84	516	300	192	624	408	405	435	219	651	111	543
511	295	727	187	619	403	52	484	268	700	160	592	376	106	538	322	214	646	430	25	457	241	673	133	565	349	79
714	174	606	390	39	471	255	687	147	579	363	93	525	309	201	633	417	12	444	228	660	120	552	336	66	498	282
613	397	46	478	262	694	154	586	370	100	532	316	208	640	424	19	451	235	667	127	559	343	73	505	289	721	181
32	464	248	680	140	572	356	86	518	302	194	626	410	5	437	221	653	113	545	329	59	491	275	707	167	599	383
265	697	157	589	373	103	535	319	211	643	427	22	454	238	670	130	562	346	76	508	292	724	184	616	400	49	481
143	575	359	89	521	305	197	629	413	8	440	224	656	116	548	332	62	494	278	710	170	602	386	35	467	251	683
366	96	528	312	204	636	420	15	447	231	663	123	555	339	69	501	285	717	177	609	393	42	474	258	690	150	582
514	298	190	622	406	442	433	217	649	109	541	325	55	487	271	703	163	595	379	28	460	244	676	136	568	352	82
215	647	431	26	458	242	674	134	566	350	80	512	296	728	188	620	404	53	485	269	701	161	593	377	107	539	323
418	13	445	229	661	121	553	337	67	499	283	715	175	607	391	40	472	256	688	148	580	364	94	526	310	202	634
452	236	668	128	560	344	74	506	290	722	182	614	398	47	479	263	695	155	587	371	101	533	317	209	641	425	20
654	114	546	330	60	492	276	708	168	600	384	33	465	249	681	141	573	357	87	519	303	195	627	411	6	438	222
563	347	77	509	293	725	185	617	401	50	482	266	698	158	590	374	104	536	320	212	644	428	23	455	239	671	131
63	495	279	711	171	603	387	36	468	252	684	144	576	360	90	522	306	198	630	414	9	441	225	657	117	549	333
286	718	178	610	394	43	475	259	691	151	583	367	97	529	313	205	637	421	16	448	232	664	124	556	340	70	502
164	596	380	29	461	245	677	137	569	353	83	515	299	191	623	407	226	434	218	650	110	542	326	56	488	272	704
402	51	483	267	699	159	591	375	105	537	321	213	645	429	24	456	240	672	132	564	348	78	510	294	726	186	618
470	254	686	146	578	362	92	524	308	200	632	416	11	443	227	659	119	551	335	65	497	281	713	173	605	389	38
693	153	585	369	99	531	315	207	639	423	18	450	234	666	126	558	342	72	504	288	720	180	612	396	45	477	261
571	355	85	517	301	193	625	409	4	436	220	652	112	544	328	58	490	274	706	166	598	382	31	463	247	679	139
102	534	318	210	642	426	21	453	237	669	129	561	345	75	507	291	723	183	615	399	48	480	264	696	156	588	372
304	196	628	412	7	439	223	655	115	547	331	61	493	277	709	169	601	385	34	466	250	682	142	574	358	88	520
635	419	14	446	230	662	122	554	338	68	500	284	716	176	608	392	41	473	257	689	149	581	365	95	527	311	203
0	432	216	648	108	540	324	54	486	270	702	162	594	378	27	459	243	675	135	567	351	81	513	297	189	621	3

Table H.12 — 29 x 29 data

2	703	123	587	355	819	65	529	297	761	181	645	413	36	500	268	732	152	616	384	94	558	326	790	210	674	442	7	1
141	605	373	837	83	547	315	779	199	663	431	54	518	286	750	170	634	402	112	576	344	808	228	692	460	25	489	257	721
359	823	69	533	301	765	185	649	417	40	504	272	736	156	620	388	98	562	330	794	214	678	446	11	475	243	707	127	591
76	540	308	772	192	656	424	47	511	279	743	163	627	395	105	569	337	801	221	685	453	18	482	250	714	134	598	366	830
293	757	177	641	409	32	496	264	728	148	612	380	90	554	322	786	206	670	438	435	467	235	699	119	583	351	815	61	525
201	665	433	56	520	288	752	172	636	404	114	578	346	810	230	694	462	27	491	259	723	143	607	375	839	85	549	317	781
419	42	506	274	738	158	622	390	100	564	332	796	216	680	448	13	477	245	709	129	593	361	825	71	535	303	767	187	651
513	281	745	165	629	397	107	571	339	803	223	687	455	20	484	252	716	136	600	368	832	78	542	310	774	194	658	426	49
730	150	614	382	92	556	324	788	208	672	440	5	469	237	701	121	585	353	817	63	527	295	759	179	643	411	34	498	266
632	400	110	574	342	806	226	690	458	23	487	255	719	139	603	371	835	81	545	313	777	197	661	429	52	516	284	748	168
96	560	328	792	212	676	444	9	473	241	705	125	589	357	821	67	531	299	763	183	647	415	38	502	270	734	154	618	386
335	799	219	683	451	16	480	248	712	132	596	364	828	74	538	306	770	190	654	422	45	509	277	741	161	625	393	103	567
204	668	436	471	465	233	697	117	581	349	813	59	523	291	755	175	639	407	30	494	262	726	146	610	378	88	552	320	784
463	28	492	260	724	144	608	376	840	86	550	318	782	202	666	434	57	521	289	753	173	637	405	115	579	347	811	231	695
478	246	710	130	594	362	826	72	536	304	768	188	652	420	43	507	275	739	159	623	391	101	565	333	797	217	681	449	14
717	137	601	369	833	79	543	311	775	195	659	427	50	514	282	746	166	630	398	108	572	340	804	224	688	456	21	485	253
586	354	818	64	528	296	760	180	644	412	35	499	267	731	151	615	383	93	557	325	789	209	673	441	6	470	238	702	122
836	82	546	314	778	198	662	430	53	517	285	749	169	633	401	111	575	343	807	227	691	459	24	488	256	720	140	604	372
532	300	764	184	648	416	39	503	271	735	155	619	387	97	561	329	793	213	677	445	10	474	242	706	126	590	358	822	68
771	191	655	423	46	510	278	742	162	626	394	104	568	336	800	220	684	452	17	481	249	713	133	597	365	829	75	539	307
640	408	31	495	263	727	147	611	379	89	553	321	785	205	669	437	239	466	234	698	118	582	350	814	60	524	292	756	176
55	519	287	751	171	635	403	113	577	345	809	229	693	461	26	490	258	722	142	606	374	838	84	548	316	780	200	664	432
273	737	157	621	389	99	563	331	795	215	679	447	12	476	244	708	128	592	360	824	70	534	302	766	186	650	418	41	505
164	628	396	106	570	338	802	222	686	454	19	483	251	715	135	599	367	831	77	541	309	773	193	657	425	48	512	280	744
381	91	555	323	787	207	671	439	4	468	236	700	120	584	352	816	62	526	294	758	178	642	410	33	497	265	729	149	613
573	341	805	225	689	457	22	486	254	718	138	602	370	834	80	544	312	776	196	660	428	51	515	283	747	167	631	399	109
791	211	675	443	8	472	240	704	124	588	356	820	66	530	298	762	182	646	414	37	501	269	733	153	617	385	95	559	327
682	450	15	479	247	711	131	595	363	827	73	537	305	769	189	653	421	44	508	276	740	160	624	392	102	566	334	798	218
0	464	232	696	116	580	348	812	58	522	290	754	174	638	406	29	493	261	725	145	609	377	87	551	319	783	203	667	3

Table H.13 — 31 x 31 data

2	759	139	635	387	883	77	573	325	821	201	697	449	945	46	542	294	790	170	666	418	914	108	604	356	852	232	728	480	15	1
147	643	395	891	85	581	333	829	209	705	457	953	54	550	302	798	178	674	426	922	116	612	364	860	240	736	488	23	519	271	767
379	875	69	565	317	813	193	689	441	937	38	534	286	782	162	658	410	906	100	596	348	844	224	720	472	7	503	255	751	131	627
89	585	337	833	213	709	461	957	58	554	306	802	182	678	430	926	120	616	368	864	244	740	492	27	523	275	771	151	647	399	895
321	817	197	693	445	941	42	538	290	786	166	662	414	910	104	600	352	848	228	724	476	11	507	259	755	135	631	383	879	73	569
205	701	453	949	50	546	298	794	174	670	422	918	112	608	360	856	236	732	484	19	515	267	763	143	639	391	887	81	577	329	825
437	933	34	530	282	778	158	654	406	902	96	592	344	840	220	716	468	465	499	251	747	127	623	375	871	65	561	313	809	189	685
60	556	308	804	184	680	432	928	122	618	370	866	246	742	494	29	525	277	773	153	649	401	897	91	587	339	835	215	711	463	959
292	788	168	664	416	912	106	602	354	850	230	726	478	13	509	261	757	137	633	385	881	75	571	323	819	199	695	447	943	44	540
176	672	424	920	114	610	362	858	238	734	486	21	517	269	765	145	641	393	889	83	579	331	827	207	703	455	951	52	548	300	796
408	904	98	594	346	842	222	718	470	5	501	253	749	129	625	377	873	67	563	315	811	191	687	439	935	36	532	284	780	160	656
118	614	366	862	242	738	490	25	521	273	769	149	645	397	893	87	583	335	831	211	707	459	955	56	552	304	800	180	676	428	924
350	846	226	722	474	9	505	257	753	133	629	381	877	71	567	319	815	195	691	443	939	40	536	288	784	164	660	412	908	102	598
234	730	482	17	513	265	761	141	637	389	885	79	575	327	823	203	699	451	947	48	544	296	792	172	668	420	916	110	606	358	854
466	511	497	249	745	125	621	373	869	63	559	311	807	187	683	435	931	32	528	280	776	156	652	404	900	94	590	342	838	218	714
526	278	774	154	650	402	898	92	588	340	836	216	712	464	960	61	557	309	805	185	681	433	929	123	619	371	867	247	743	495	30
758	138	634	386	882	76	572	324	820	200	696	448	944	45	541	293	789	169	665	417	913	107	603	355	851	231	727	479	14	510	262
642	394	890	84	580	332	828	208	704	456	952	53	549	301	797	177	673	425	921	115	611	363	859	239	735	487	22	518	270	766	146
874	68	564	316	812	192	688	440	936	37	533	285	781	161	657	409	905	99	595	347	843	223	719	471	6	502	254	750	130	626	378
584	336	832	212	708	460	956	57	553	305	801	181	677	429	925	119	615	367	863	243	739	491	26	522	274	770	150	646	398	894	88
816	196	692	444	940	41	537	289	785	165	661	413	909	103	599	351	847	227	723	475	10	506	258	754	134	630	382	878	72	568	320
700	452	948	49	545	297	793	173	669	421	917	111	607	359	855	235	731	483	18	514	266	762	142	638	390	886	80	576	328	824	204
932	33	529	281	777	157	653	405	901	95	591	343	839	219	715	467	263	498	250	746	126	622	374	870	64	560	312	808	188	684	436
555	307	803	183	679	431	927	121	617	369	865	245	741	493	28	524	276	772	152	648	400	896	90	586	338	834	214	710	462	958	59
787	167	663	415	911	105	601	353	849	229	725	477	12	508	260	756	136	632	384	880	74	570	322	818	198	694	446	942	43	539	291
671	423	919	113	609	361	857	237	733	485	20	516	268	764	144	640	392	888	82	578	330	826	206	702	454	950	51	547	299	795	175
903	97	593	345	841	221	717	469	4	500	252	748	128	624	376	872	66	562	314	810	190	686	438	934	35	531	283	779	159	655	407
613	365	861	241	737	489	24	520	272	768	148	644	396	892	86	582	334	830	210	706	458	954	55	551	303	799	179	675	427	923	117
845	225	721	473	8	504	256	752	132	628	380	876	70	566	318	814	194	690	442	938	39	535	287	783	163	659	411	907	101	597	349
729	481	16	512	264	760	140	636	388	884	78	574	326	822	202	698	450	946	47	543	295	791	171	667	419	915	109	605	357	853	233
0	496	248	744	124	620	372	868	62	558	310	806	186	682	434	930	31	527	279	775	155	651	403	899	93	589	341	837	217	713	3

Table H.14 — 33 x 33 data

2	265	793	133	661	397	925	67	595	331	859	199	727	463	991	34	562	298	826	166	694	430	958	100	628	364	892	232	760	496	1024	1057	1
824	164	692	428	956	98	626	362	890	230	758	494	1022	65	593	329	857	197	725	461	989	131	659	395	923	263	791	527	1055	32	1088	560	296
676	412	940	82	610	346	874	214	742	478	1006	49	577	313	841	181	709	445	973	115	643	379	907	247	775	511	1039	16	1072	544	280	808	148
948	90	618	354	882	222	750	486	1014	57	585	321	849	189	717	453	981	123	651	387	915	255	783	519	1047	24	1080	552	288	816	156	684	420
602	338	866	206	734	470	998	41	569	305	833	173	701	437	965	107	635	371	899	239	767	503	1031	8	1064	536	272	800	140	668	404	932	74
886	226	754	490	1018	61	589	325	853	193	721	457	985	127	655	391	919	259	787	523	1051	28	1084	556	292	820	160	688	424	952	94	622	358
738	474	1002	45	573	309	837	177	705	441	969	111	639	375	903	243	771	507	1035	12	1068	540	276	804	144	672	408	936	78	606	342	870	210
1010	53	581	317	845	185	713	449	977	119	647	383	911	251	779	515	1043	20	1076	548	284	812	152	680	416	944	86	614	350	878	218	746	482
565	301	829	169	697	433	961	103	631	367	895	235	763	499	1027	4	1060	532	268	796	136	664	400	928	70	598	334	862	202	730	466	994	37
855	195	723	459	987	129	657	393	921	261	789	525	1053	30	1086	558	294	822	162	690	426	954	96	624	360	888	228	756	492	1020	63	591	327
707	443	971	113	641	377	905	245	773	509	1037	14	1070	542	278	806	146	674	410	938	80	608	344	872	212	740	476	1004	47	575	311	839	179
979	121	649	385	913	253	781	517	1045	22	1078	550	286	814	154	682	418	946	88	616	352	880	220	748	484	1012	55	583	319	847	187	715	451
633	369	897	237	765	501	1029	6	1062	534	270	798	138	666	402	930	72	600	336	864	204	732	468	996	39	567	303	831	171	699	435	963	105
917	257	785	521	1049	26	1082	554	290	818	158	686	422	950	92	620	356	884	224	752	488	1016	59	587	323	851	191	719	455	983	125	653	389
769	505	1033	10	1066	538	274	802	142	670	406	934	76	604	340	868	208	736	472	1000	43	571	307	835	175	703	439	967	109	637	373	901	241
1041	18	1074	546	282	810	150	678	414	942	84	612	348	876	216	744	480	1008	51	579	315	843	183	711	447	975	117	645	381	909	249	777	513
1058	530	266	794	134	662	398	926	68	596	332	860	200	728	464	992	35	563	299	827	167	695	431	959	101	629	365	893	233	761	497	1025	529
295	823	163	691	427	955	97	625	361	889	229	757	493	1021	64	592	328	856	196	724	460	988	130	658	394	922	262	790	526	1054	31	1087	559
147	675	411	939	81	609	345	873	213	741	477	1005	48	576	312	840	180	708	444	972	114	642	378	906	246	774	510	1038	15	1071	543	279	807
419	947	89	617	353	881	221	749	485	1013	56	584	320	848	188	716	452	980	122	650	386	914	254	782	518	1046	23	1079	551	287	815	155	683
73	601	337	865	205	733	469	997	40	568	304	832	172	700	436	964	106	634	370	898	238	766	502	1030	7	1063	535	271	799	139	667	403	931
357	885	225	753	489	1017	60	588	324	852	192	720	456	984	126	654	390	918	258	786	522	1050	27	1083	555	291	819	159	687	423	951	93	621
209	737	473	1001	44	572	308	836	176	704	440	968	110	638	374	902	242	770	506	1034	11	1067	539	275	803	143	671	407	935	77	605	341	869
481	1009	52	580	316	844	184	712	448	976	118	646	382	910	250	778	514	1042	19	1075	547	283	811	151	679	415	943	85	613	349	877	217	745
36	564	300	828	168	696	432	960	102	630	366	894	234	762	498	1026	1023	1059	531	267	795	135	663	399	927	69	597	333	861	201	729	465	993
326	854	194	722	458	986	128	656	392	920	260	788	524	1052	29	1085	557	293	821	161	689	425	953	95	623	359	887	227	755	491	1019	62	590
178	706	442	970	112	640	376	904	244	772	508	1036	13	1069	541	277	805	145	673	409	937	79	607	343	871	211	739	475	1003	46	574	310	838
450	978	120	648	384	912	252	780	516	1044	21	1077	549	285	813	153	681	417	945	87	615	351	879	219	747	483	1011	54	582	318	846	186	714
104	632	368	896	236	764	500	1028	5	1061	533	269	797	137	665	401	929	71	599	335	863	203	731	467	995	38	566	302	830	170	698	434	962
388	916	256	784	520	1048	25	1081	553	289	817	157	685	421	949	91	619	355	883	223	751	487	1015	58	586	322	850	190	718	454	982	124	652
240	768	504	1032	9	1065	537	273	801	141	669	405	933	75	603	339	867	207	735	471	999	42	570	306	834	174	702	438	966	108	636	372	900
512	1040	17	1073	545	281	809	149	677	413	941	83	611	347	875	215	743	479	1007	50	578	314	842	182	710	446	974	116	644	380	908	248	776
0	1056	528	264	792	132	660	396	924	66	594	330	858	198	726	462	990	33	561	297	825	165	693	429	957	99	627	363	891	231	759	495	3

Table H.15 — 35 x 35 data

2	290	850	150	710	430	990	80	1200	640	360	920	220	780	500	1060	45	1165	605	325	885	185	745	465	1025	115	675	395	955	255	815	535	1095	10	1
859	159	719	439	999	89	1209	649	369	929	229	789	509	1069	54	1174	614	334	894	194	754	474	1034	124	684	404	964	264	824	544	1104	19	1139	579	299
701	421	981	71	1191	631	351	911	211	771	491	1051	36	1156	596	316	876	176	736	456	1016	106	666	386	946	246	806	526	1086	1130	1121	561	281	841	141
1014	104	1224	664	384	944	244	804	524	1084	69	1189	629	349	909	209	769	489	1049	139	699	419	979	279	839	559	1119	34	1154	594	314	874	174	734	454
1207	647	367	927	227	787	507	1067	52	1172	612	332	892	192	752	472	1032	122	682	402	962	262	822	542	1102	17	1137	577	297	857	157	717	437	997	87
376	936	236	796	516	1076	61	1181	621	341	901	201	761	481	1041	131	691	411	971	271	831	551	1111	26	1146	586	306	866	166	726	446	1006	96	1216	656
218	778	498	1058	43	1163	603	323	883	183	743	463	1023	113	673	393	953	253	813	533	1093	8	1128	568	288	848	148	708	428	988	78	1198	638	358	918
520	1080	65	1185	625	345	905	205	765	485	1045	135	695	415	975	275	835	555	1115	30	1150	590	310	870	170	730	450	1010	100	1220	660	380	940	240	800
48	1168	608	328	888	188	748	468	1028	118	678	398	958	258	818	538	1098	13	1133	573	293	853	153	713	433	993	83	1203	643	363	923	223	783	503	1063
617	337	897	197	757	477	1037	127	687	407	967	267	827	547	1107	22	1142	582	302	862	162	722	442	1002	92	1212	652	372	932	232	792	512	1072	57	1177
879	179	739	459	1019	109	669	389	949	249	809	529	1089	4	1124	564	284	844	144	704	424	984	74	1194	634	354	914	214	774	494	1054	39	1159	599	319
767	487	1047	137	697	417	977	277	837	557	1117	32	1152	592	312	872	172	732	452	1012	102	1222	662	382	942	242	802	522	1082	67	1187	627	347	907	207
1030	120	680	400	960	260	820	540	1100	15	1135	575	295	855	155	715	435	995	85	1205	645	365	925	225	785	505	1065	50	1170	610	330	890	190	750	470
689	409	969	269	829	549	1109	24	1144	584	304	864	164	724	444	1004	94	1214	654	374	934	234	794	514	1074	59	1179	619	339	899	199	759	479	1039	129
951	251	811	531	1091	6	1126	566	286	846	146	706	426	986	76	1196	636	356	916	216	776	496	1056	41	1161	601	321	881	181	741	461	1021	111	671	391
833	553	1113	28	1148	588	308	868	168	728	448	1008	98	1218	658	378	938	238	798	518	1078	63	1183	623	343	903	203	763	483	1043	133	693	413	973	273
1096	11	1131	571	291	851	151	711	431	991	81	1201	641	361	921	221	781	501	1061	46	1166	606	326	886	186	746	466	1026	116	676	396	956	256	816	536
1140	580	300	860	160	720	440	1000	90	1210	650	370	930	230	790	510	1070	55	1175	615	335	895	195	755	475	1035	125	685	405	965	265	825	545	1105	20
282	842	142	702	422	982	72	1192	632	352	912	212	772	492	1052	37	1157	597	317	877	177	737	457	1017	107	667	387	947	247	807	527	1087	570	1122	562
173	733	453	1013	103	1223	663	383	943	243	803	523	1083	68	1188	628	348	908	208	768	488	1048	138	698	418	978	278	838	558	1118	33	1153	593	313	873
436	996	86	1206	646	366	926	226	786	506	1066	51	1171	611	331	891	191	751	471	1031	121	681	401	961	261	821	541	1101	16	1136	576	296	856	156	716
95	1215	655	375	935	235	795	515	1075	60	1180	620	340	900	200	760	480	1040	130	690	410	970	270	830	550	1110	25	1145	585	305	865	165	725	445	1005
637	357	917	217	777	497	1057	42	1162	602	322	882	182	742	462	1022	112	672	392	952	252	812	532	1092	7	1127	567	287	847	147	707	427	987	77	1197
939	239	799	519	1079	64	1184	624	344	904	204	764	484	1044	134	694	414	974	274	834	554	1114	29	1149	589	309	869	169	729	449	1009	99	1219	659	379
782	502	1062	47	1167	607	327	887	187	747	467	1027	117	677	397	957	257	817	537	1097	12	1132	572	292	852	152	712	432	992	82	1202	642	362	922	222
1071	56	1176	616	336	896	196	756	476	1036	126	686	406	966	266	826	546	1106	21	1141	581	301	861	161	721	441	1001	91	1211	651	371	931	231	791	511
1158	598	318	878	178	738	458	1018	108	668	388	948	248	808	528	1088	1085	1123	563	283	843	143	703	423	983	73	1193	633	353	913	213	773	493	1053	38
346	906	206	766	486	1046	136	696	416	976	276	836	556	1116	31	1151	591	311	871	171	731	451	1011	101	1221	661	381	941	241	801	521	1081	66	1186	626
189	749	469	1029	119	679	399	959	259	819	539	1099	14	1134	574	294	854	154	714	434	994	84	1204	644	364	924	224	784	504	1064	49	1169	609	329	889
478	1038	128	688	408	968	268	828	548	1108	23	1143	583	303	863	163	723	443	1003	93	1213	653	373	933	233	793	513	1073	58	1178	618	338	898	198	758
110	670	390	950	250	810	530	1090	5	1125	565	285	845	145	705	425	985	75	1195	635	355	915	215	775	495	1055	40	1160	600	320	880	180	740	460	1020
412	972	272	832	552	1112	27	1147	587	307	867	167	727	447	1007	97	1217	657	377	937	237	797	517	1077	62	1182	622	342	902	202	762	482	1042	132	692
254	814	534	1094	9	1129	569	289	849	149	709	429	989	79	1199	639	359	919	219	779	499	1059	44	1164	604	324	884	184	744	464	1024	114	674	394	954
543	1103	18	1138	578	298	858	158	718	438	998	88	1208	648	368	928	228	788	508	1068	53	1173	613	333	893	193	753	473	1033	123	683	403	963	263	823
0	1120	560	280	840	140	700	420	980	70	1190	630	350	910	210	770	490	1050	35	1155	595	315	875	175	735	455	1015	105	665	385	945	245	805	525	3

Table H.16 — 37 x 37 data

2	302	894	154	1338	746	450	1042	80	1264	672	376	968	228	820	524	1116	43	1227	635	339	931	191	783	487	1079	117	1301	709	413	1005	265	857	561	1153	6	1
917	177	1361	769	473	1065	103	1287	695	399	991	251	843	547	1139	66	1250	658	362	954	214	806	510	1102	140	1324	732	436	1028	288	880	584	1176	29	1213	621	325
1343	751	455	1047	85	1269	677	381	973	233	825	529	1121	48	1232	640	344	936	196	788	492	1084	122	1306	714	418	1010	270	862	566	1158	11	1195	603	307	899	159
464	1056	94	1278	686	390	982	242	834	538	1130	57	1241	649	353	945	205	797	501	1093	131	1315	723	427	1019	279	871	575	1167	20	1204	612	316	908	168	1352	760
75	1259	667	371	963	223	815	519	1111	38	1222	630	334	926	186	778	482	1074	112	1296	704	408	1000	260	852	556	1148	1190	1185	593	297	889	149	1333	741	445	1037
702	406	998	258	850	554	1146	73	1257	665	369	961	221	813	517	1109	147	1331	739	443	1035	295	887	591	1183	36	1220	628	332	924	184	1368	776	480	1072	110	1294
980	240	832	536	1128	55	1239	647	351	943	203	795	499	1091	129	1313	721	425	1017	277	869	573	1165	18	1202	610	314	906	166	1350	758	462	1054	92	1276	684	388
841	545	1137	64	1248	656	360	952	212	804	508	1100	138	1322	730	434	1026	286	878	582	1174	27	1211	619	323	915	175	1359	767	471	1063	101	1285	693	397	989	249
1119	46	1230	638	342	934	194	786	490	1082	120	1304	712	416	1008	268	860	564	1156	9	1193	601	305	897	157	1341	749	453	1045	83	1267	675	379	971	231	823	527
1253	661	365	957	217	809	513	1105	143	1327	735	439	1031	291	883	587	1179	32	1216	624	328	920	180	1364	772	476	1068	106	1290	698	402	994	254	846	550	1142	69
347	939	199	791	495	1087	125	1309	717	421	1013	273	865	569	1161	14	1198	606	310	902	162	1346	754	458	1050	88	1272	680	384	976	236	828	532	1124	51	1235	643
208	800	504	1096	134	1318	726	430	1022	282	874	578	1170	23	1207	615	319	911	171	1355	763	467	1059	97	1281	689	393	985	245	837	541	1133	60	1244	652	356	948
485	1077	115	1299	707	411	1003	263	855	559	1151	4	1188	596	300	892	152	1336	744	448	1040	78	1262	670	374	966	226	818	522	1114	41	1225	633	337	929	189	781
145	1329	737	441	1033	293	885	589	1181	34	1218	626	330	922	182	1366	774	478	1070	108	1292	700	404	996	256	848	552	1144	71	1255	663	367	959	219	811	515	1107
719	423	1015	275	867	571	1163	16	1200	608	312	904	164	1348	756	460	1052	90	1274	682	386	978	238	830	534	1126	53	1237	645	349	941	201	793	497	1089	127	1311
1024	284	876	580	1172	25	1209	617	321	913	173	1357	765	469	1061	99	1283	691	395	987	247	839	543	1135	62	1246	654	358	950	210	802	506	1098	136	1320	728	432
858	562	1154	7	1191	599	303	895	155	1339	747	451	1043	81	1265	673	377	969	229	821	525	1117	44	1228	636	340	932	192	784	488	1080	118	1302	710	414	1006	266
1177	30	1214	622	326	918	178	1362	770	474	1066	104	1288	696	400	992	252	844	548	1140	67	1251	659	363	955	215	807	511	1103	141	1325	733	437	1029	289	881	585
1196	604	308	900	160	1344	752	456	1048	86	1270	678	382	974	234	826	530	1122	49	1233	641	345	937	197	789	493	1085	123	1307	715	419	1011	271	863	567	1159	12
317	909	169	1353	761	465	1057	95	1279	687	391	983	243	835	539	1131	58	1242	650	354	946	206	798	502	1094	132	1316	724	428	1020	280	872	576	1168	21	1205	613
150	1334	742	446	1038	76	1260	668	372	964	224	816	520	1112	39	1223	631	335	927	187	779	483	1075	113	1297	705	409	1001	261	853	557	1149	598	1186	594	298	890
775	479	1071	109	1293	701	405	997	257	849	553	1145	72	1256	664	368	960	220	812	516	1108	146	1330	738	442	1034	294	886	590	1182	35	1219	627	331	923	183	1367
1053	91	1275	683	387	979	239	831	535	1127	54	1238	646	350	942	202	794	498	1090	128	1312	720	424	1016	276	868	572	1164	17	1201	609	313	905	165	1349	757	461
1284	692	396	988	248	840	544	1136	63	1247	655	359	951	211	803	507	1099	137	1321	729	433	1025	285	877	581	1173	26	1210	618	322	914	174	1358	766	470	1062	100
378	970	230	822	526	1118	45	1229	637	341	933	193	785	489	1081	119	1303	711	415	1007	267	859	563	1155	8	1192	600	304	896	156	1340	748	452	1044	82	1266	674
253	845	549	1141	68	1252	660	364	956	216	808	512	1104	142	1326	734	438	1030	290	882	586	1178	31	1215	623	327	919	179	1363	771	475	1067	105	1289	697	401	993
531	1123	50	1234	642	346	938	198	790	494	1086	124	1308	716	420	1012	272	864	568	1160	13	1197	605	309	901	161	1345	753	457	1049	87	1271	679	383	975	235	827
59	1243	651	355	947	207	799	503	1095	133	1317	725	429	1021	281	873	577	1169	22	1206	614	318	910	170	1354	762	466	1058	96	1280	688	392	984	244	836	540	1132
632	336	928	188	780	484	1076	114	1298	706	410	1002	262	854	558	1150	1147	1187	595	299	891	151	1335	743	447	1039	77	1261	669	373	965	225	817	521	1113	40	1224
958	218	810	514	1106	144	1328	736	440	1032	292	884	588	1180	33	1217	625	329	921	181	1365	773	477	1069	107	1291	699	403	995	255	847	551	1143	70	1254	662	366
792	496	1088	126	1310	718	422	1014	274	866	570	1162	15	1199	607	311	903	163	1347	755	459	1051	89	1273	681	385	977	237	829	533	1125	52	1236	644	348	940	200
1097	135	1319	727	431	1023	283	875	579	1171	24	1208	616	320	912	172	1356	764	468	1060	98	1282	690	394	986	246	838	542	1134	61	1245	653	357	949	209	801	505
1300	708	412	1004	264	856	560	1152	5	1189	597	301	893	153	1337	745	449	1041	79	1263	671	375	967	227	819	523	1115	42	1226	634	338	930	190	782	486	1078	116
435	1027	287	879	583	1175	28	1212	620	324	916	176	1360	768	472	1064	102	1286	694	398	990	250	842	546	1138	65	1249	657	361	953	213	805	509	1101	139	1323	731
269	861	565	1157	10	1194	602	306	898	158	1342	750	454	1046	84	1268	676	380	972	232	824	528	1120	47	1231	639	343	935	195	787	491	1083	121	1305	713	417	1009
574	1166	19	1203	611	315	907	167	1351	759	463	1055	93	1277	685	389	981	241	833	537	1129	56	1240	648	352	944	204	796	500	1092	130	1314	722	426	1018	278	870
0	1184	592	296	888	148	1332	740	444	1036	74	1258	666	370	962	222	814	518	1110	37	1221	629	333	925	185	777	481	1073	111	1295	703	407	999	259	851	555	3

Table H.17 — 39 x 39 data

2	328	952	172	1420	796	484	1108	94	1342	718	406	1030	250	1498	874	562	1186	55	1303	679	367	991	211	1459	835	523	1147	133	1381	757	445	1069	289	913	601	1225	16	1
962	182	1430	806	494	1118	104	1352	728	416	1040	260	1508	884	572	1196	65	1313	689	377	1001	221	1469	845	533	1157	143	1391	767	455	1079	299	923	611	1235	26	1274	650	338
1410	786	474	1098	84	1332	708	396	1020	240	1488	864	552	1176	45	1293	669	357	981	201	1449	825	513	1137	123	1371	747	435	1059	279	903	591	1215	6	1254	630	318	942	162
499	1123	109	1357	733	421	1045	265	1513	889	577	1201	70	1318	694	382	1006	226	1474	850	538	1162	148	1396	772	460	1084	304	928	616	1240	31	1279	655	343	967	187	1435	811
89	1337	713	401	1025	245	1493	869	557	1181	50	1298	674	362	986	206	1454	830	518	1142	128	1376	752	440	1064	284	908	596	1220	11	1259	635	323	947	167	1415	791	479	1103
723	411	1035	255	1503	879	567	1191	60	1308	684	372	996	216	1464	840	528	1152	138	1386	762	450	1074	294	918	606	1230	21	1269	645	333	957	177	1425	801	489	1113	99	1347
1015	235	1483	859	547	1171	40	1288	664	352	976	196	1444	820	508	1132	118	1366	742	430	1054	274	898	586	1210	1264	1249	625	313	937	157	1405	781	469	1093	79	1327	703	391
1520	896	584	1208	77	1325	701	389	1013	233	1481	857	545	1169	155	1403	779	467	1091	311	935	623	1247	38	1286	662	350	974	194	1442	818	506	1130	116	1364	740	428	1052	272
565	1189	58	1306	682	370	994	214	1462	838	526	1150	136	1384	760	448	1072	292	916	604	1228	19	1267	643	331	955	175	1423	799	487	1111	97	1345	721	409	1033	253	1501	877
68	1316	692	380	1004	224	1472	848	536	1160	146	1394	770	458	1082	302	926	614	1238	29	1277	653	341	965	185	1433	809	497	1121	107	1355	731	419	1043	263	1511	887	575	1199
672	360	984	204	1452	828	516	1140	126	1374	750	438	1062	282	906	594	1218	9	1257	633	321	945	165	1413	789	477	1101	87	1335	711	399	1023	243	1491	867	555	1179	48	1296
1009	229	1477	853	541	1165	151	1399	775	463	1087	307	931	619	1243	34	1282	658	346	970	190	1438	814	502	1126	112	1360	736	424	1048	268	1516	892	580	1204	73	1321	697	385
1457	833	521	1145	131	1379	755	443	1067	287	911	599	1223	14	1262	638	326	950	170	1418	794	482	1106	92	1340	716	404	1028	248	1496	872	560	1184	53	1301	677	365	989	209
531	1155	141	1389	765	453	1077	297	921	609	1233	24	1272	648	336	960	180	1428	804	492	1116	102	1350	726	414	1038	258	1506	882	570	1194	63	1311	687	375	999	219	1467	843
121	1369	745	433	1057	277	901	589	1213	4	1252	628	316	940	160	1408	784	472	1096	82	1330	706	394	1018	238	1486	862	550	1174	43	1291	667	355	979	199	1447	823	511	1135
777	465	1089	309	933	621	1245	36	1284	660	348	972	192	1440	816	504	1128	114	1362	738	426	1050	270	1518	894	582	1206	75	1323	699	387	1011	231	1479	855	543	1167	153	1401
1070	290	914	602	1226	17	1265	641	329	953	173	1421	797	485	1109	95	1343	719	407	1031	251	1499	875	563	1187	56	1304	680	368	992	212	1460	836	524	1148	134	1382	758	446
924	612	1236	27	1275	651	339	963	183	1431	807	495	1119	105	1353	729	417	1041	261	1509	885	573	1197	66	1314	690	378	1002	222	1470	846	534	1158	144	1392	768	456	1080	300
1216	7	1255	631	319	943	163	1411	787	475	1099	85	1333	709	397	1021	241	1489	865	553	1177	46	1294	670	358	982	202	1450	826	514	1138	124	1372	748	436	1060	280	904	592
1280	656	344	968	188	1436	812	500	1124	110	1358	734	422	1046	266	1514	890	578	1202	71	1319	695	383	1007	227	1475	851	539	1163	149	1397	773	461	1085	305	929	617	1241	32
324	948	168	1416	792	480	1104	90	1338	714	402	1026	246	1494	870	558	1182	51	1299	675	363	987	207	1455	831	519	1143	129	1377	753	441	1065	285	909	597	1221	12	1260	636
178	1426	802	490	1114	100	1348	724	412	1036	256	1504	880	568	1192	61	1309	685	373	997	217	1465	841	529	1153	139	1387	763	451	1075	295	919	607	1231	22	1270	646	334	958
782	470	1094	80	1328	704	392	1016	236	1484	860	548	1172	41	1289	665	353	977	197	1445	821	509	1133	119	1367	743	431	1055	275	899	587	1211	640	1250	626	314	938	158	1406
1129	115	1363	739	427	1051	271	1519	895	583	1207	76	1324	700	388	1012	232	1480	856	544	1168	154	1402	778	466	1090	310	934	622	1246	37	1285	661	349	973	193	1441	817	505
1344	720	408	1032	252	1500	876	564	1188	57	1305	681	369	993	213	1461	837	525	1149	135	1383	759	447	1071	291	915	603	1227	18	1266	642	330	954	174	1422	798	486	1110	96
418	1042	262	1510	886	574	1198	67	1315	691	379	1003	223	1471	847	535	1159	145	1393	769	457	1081	301	925	613	1237	28	1276	652	340	964	184	1432	808	496	1120	106	1354	730
242	1490	866	554	1178	47	1295	671	359	983	203	1451	827	515	1139	125	1373	749	437	1061	281	905	593	1217	8	1256	632	320	944	164	1412	788	476	1100	86	1334	710	398	1022
891	579	1203	72	1320	696	384	1008	228	1476	852	540	1164	150	1398	774	462	1086	306	930	618	1242	33	1281	657	345	969	189	1437	813	501	1125	111	1359	735	423	1047	267	1515
1183	52	1300	676	364	988	208	1456	832	520	1144	130	1378	754	442	1066	286	910	598	1222	13	1261	637	325	949	169	1417	793	481	1105	91	1339	715	403	1027	247	1495	871	559
1310	686	374	998	218	1466	842	530	1154	140	1388	764	452	1076	296	920	608	1232	23	1271	647	335	959	179	1427	803	491	1115	101	1349	725	413	1037	257	1505	881	569	1193	62
354	978	198	1446	822	510	1134	120	1368	744	432	1056	276	900	588	1212	1209	1251	627	315	939	159	1407	783	471	1095	81	1329	705	393	1017	237	1485	861	549	1173	42	1290	666
230	1478	854	542	1166	152	1400	776	464	1088	308	932	620	1244	35	1283	659	347	971	191	1439	815	503	1127	113	1361	737	425	1049	269	1517	893	581	1205	74	1322	698	386	1010
834	522	1146	132	1380	756	444	1068	288	912	600	1224	15	1263	639	327	951	171	1419	795	483	1107	93	1341	717	405	1029	249	1497	873	561	1185	54	1302	678	366	990	210	1458
1156	142	1390	766	454	1078	298	922	610	1234	25	1273	649	337	961	181	1429	805	493	1117	103	1351	727	415	1039	259	1507	883	571	1195	64	1312	688	376	1000	220	1468	844	532
1370	746	434	1058	278	902	590	1214	5	1253	629	317	941	161	1409	785	473	1097	83	1331	707	395	1019	239	1487	863	551	1175	44	1292	668	356	980	200	1448	824	512	1136	122
459	1083	303	927	615	1239	30	1278	654	342	966	186	1434	810	498	1122	108	1356	732	420	1044	264	1512	888	576	1200	69	1317	693	381	1005	225	1473	849	537	1161	147	1395	771
283	907	595	1219	10	1258	634	322	946	166	1414	790	478	1102	88	1336	712	400	1024	244	1492	868	556	1180	49	1297	673	361	985	205	1453	829	517	1141	127	1375	751	439	1063
605	1229	20	1268	644	332	956	176	1424	800	488	1112	98	1346	722	410	1034	254	1502	878	566	1190	59	1307	683	371	995	215	1463	839	527	1151	137	1385	761	449	1073	293	917
0	1248	624	312	936	156	1404	780	468	1092	78	1326	702	390	1014	234	1482	858	546	1170	39	1287	663	351	975	195	1443	819	507	1131	117	1365	741	429	1053	273	897	585	3

Table H.18 — 41 x 41 data

2	332	1644	988	168	1480	824	496	1152	86	1398	742	414	1070	250	1562	906	578	1234	45	1357	701	373	1029	209	1521	865	537	1193	127	1439	783	455	1111	291	1603	947	619	1275	4	1
1677	1021	201	1513	857	529	1185	119	1431	775	447	1103	283	1595	939	611	1267	78	1390	734	406	1062	242	1554	898	570	1226	160	1472	816	488	1144	324	1636	980	652	1308	37	1349	693	365
181	1493	837	509	1165	99	1411	755	427	1083	263	1575	919	591	1247	58	1370	714	386	1042	222	1534	878	550	1206	140	1452	796	468	1124	304	1616	960	632	1288	17	1329	673	345	1657	1001
847	519	1175	109	1421	765	437	1093	273	1585	929	601	1257	68	1380	724	396	1052	232	1544	888	560	1216	150	1462	806	478	1134	314	1626	970	642	1298	27	1339	683	355	1667	1011	191	1503
1155	89	1401	745	417	1073	253	1565	909	581	1237	48	1360	704	376	1032	212	1524	868	540	1196	130	1442	786	458	1114	294	1606	950	622	1278	7	1319	663	335	1647	991	171	1483	827	499
1426	770	442	1098	278	1590	934	606	1262	73	1385	729	401	1057	237	1549	893	565	1221	155	1467	811	483	1139	319	1631	975	647	1303	32	1344	688	360	1672	1016	196	1508	852	524	1180	114
422	1078	258	1570	914	586	1242	53	1365	709	381	1037	217	1529	873	545	1201	135	1447	791	463	1119	299	1611	955	627	1283	12	1324	668	340	1652	996	176	1488	832	504	1160	94	1406	750
268	1580	924	596	1252	63	1375	719	391	1047	227	1539	883	555	1211	145	1457	801	473	1129	309	1621	965	637	1293	22	1334	678	350	1662	1006	186	1498	842	514	1170	104	1416	760	432	1088
903	575	1231	42	1354	698	370	1026	206	1518	862	534	1190	124	1436	780	452	1108	288	1600	944	616	1272	1316	1313	657	329	1641	985	165	1477	821	493	1149	83	1395	739	411	1067	247	1559
1270	81	1393	737	409	1065	245	1557	901	573	1229	163	1475	819	491	1147	327	1639	983	655	1311	40	1352	696	368	1680	1024	204	1516	860	532	1188	122	1434	778	450	1106	286	1598	942	614
1373	717	389	1045	225	1537	881	553	1209	143	1455	799	471	1127	307	1619	963	635	1291	20	1332	676	348	1660	1004	184	1496	840	512	1168	102	1414	758	430	1086	266	1578	922	594	1250	61
399	1055	235	1547	891	563	1219	153	1465	809	481	1137	317	1629	973	645	1301	30	1342	686	358	1670	1014	194	1506	850	522	1178	112	1424	768	440	1096	276	1588	932	604	1260	71	1383	727
215	1527	871	543	1199	133	1445	789	461	1117	297	1609	953	625	1281	10	1322	666	338	1650	994	174	1486	830	502	1158	92	1404	748	420	1076	256	1568	912	584	1240	51	1363	707	379	1035
896	568	1224	158	1470	814	486	1142	322	1634	978	650	1306	35	1347	691	363	1675	1019	199	1511	855	527	1183	117	1429	773	445	1101	281	1593	937	609	1265	76	1388	732	404	1060	240	1552
1204	138	1450	794	466	1122	302	1614	958	630	1286	15	1327	671	343	1655	999	179	1491	835	507	1163	97	1409	753	425	1081	261	1573	917	589	1245	56	1368	712	384	1040	220	1532	876	548
1460	804	476	1132	312	1624	968	640	1296	25	1337	681	353	1665	1009	189	1501	845	517	1173	107	1419	763	435	1091	271	1583	927	599	1255	66	1378	722	394	1050	230	1542	886	558	1214	148
456	1112	292	1604	948	620	1276	5	1317	661	333	1645	989	169	1481	825	497	1153	87	1399	743	415	1071	251	1563	907	579	1235	46	1358	702	374	1030	210	1522	866	538	1194	128	1440	784
325	1637	981	653	1309	38	1350	694	366	1678	1022	202	1514	858	530	1186	120	1432	776	448	1104	284	1596	940	612	1268	79	1391	735	407	1063	243	1555	899	571	1227	161	1473	817	489	1145
961	633	1289	18	1330	674	346	1658	1002	182	1494	838	510	1166	100	1412	756	428	1084	264	1576	920	592	1248	59	1371	715	387	1043	223	1535	879	551	1207	141	1453	797	469	1125	305	1617
1299	28	1340	684	356	1668	1012	192	1504	848	520	1176	110	1422	766	438	1094	274	1586	930	602	1258	69	1381	725	397	1053	233	1545	889	561	1217	151	1463	807	479	1135	315	1627	971	643
1320	664	336	1648	992	172	1484	828	500	1156	90	1402	746	418	1074	254	1566	910	582	1238	49	1361	705	377	1033	213	1525	869	541	1197	131	1443	787	459	1115	295	1607	951	623	1279	8
361	1673	1017	197	1509	853	525	1181	115	1427	771	443	1099	279	1591	935	607	1263	74	1386	730	402	1058	238	1550	894	566	1222	156	1468	812	484	1140	320	1632	976	648	1304	33	1345	689
997	177	1489	833	505	1161	95	1407	751	423	1079	259	1571	915	587	1243	54	1366	710	382	1038	218	1530	874	546	1202	136	1448	792	464	1120	300	1612	956	628	1284	13	1325	669	341	1653
1499	843	515	1171	105	1417	761	433	1089	269	1581	925	597	1253	64	1376	720	392	1048	228	1540	884	556	1212	146	1458	802	474	1130	310	1622	966	638	1294	23	1335	679	351	1663	1007	187
494	1150	84	1396	740	412	1068	248	1560	904	576	1232	43	1355	699	371	1027	207	1519	863	535	1191	125	1437	781	453	1109	289	1601	945	617	1273	660	1314	658	330	1642	986	166	1478	822
121	1433	777	449	1105	285	1597	941	613	1269	80	1392	736	408	1064	244	1556	900	572	1228	162	1474	818	490	1146	326	1638	982	654	1310	39	1351	695	367	1679	1023	203	1515	859	531	1187
757	429	1085	265	1577	921	593	1249	60	1372	716	388	1044	224	1536	880	552	1208	142	1454	798	470	1126	306	1618	962	634	1290	19	1331	675	347	1659	1003	183	1495	839	511	1167	101	1413
1095	275	1587	931	603	1259	70	1382	726	398	1054	234	1546	890	562	1218	152	1464	808	480	1136	316	1628	972	644	1300	29	1341	685	357	1669	1013	193	1505	849	521	1177	111	1423	767	439
1567	911	583	1239	50	1362	706	378	1034	214	1526	870	542	1198	132	1444	788	460	1116	296	1608	952	624	1280	9	1321	665	337	1649	993	173	1485	829	501	1157	91	1403	747	419	1075	255
608	1264	75	1387	731	403	1059	239	1551	895	567	1223	157	1469	813	485	1141	321	1633	977	649	1305	34	1346	690	362	1674	1018	198	1510	854	526	1182	116	1428	772	444	1100	280	1592	936
55	1367	711	383	1039	219	1531	875	547	1203	137	1449	793	465	1121	301	1613	957	629	1285	14	1326	670	342	1654	998	178	1490	834	506	1162	96	1408	752	424	1080	260	1572	916	588	1244
721	393	1049	229	1541	885	557	1213	147	1459	803	475	1131	311	1623	967	639	1295	24	1336	680	352	1664	1008	188	1500	844	516	1172	106	1418	762	434	1090	270	1582	926	598	1254	65	1377
1028	208	1520	864	536	1192	126	1438	782	454	1110	290	1602	946	618	1274	1271	1315	659	331	1643	987	167	1479	823	495	1151	85	1397	741	413	1069	249	1561	905	577	1233	44	1356	700	372
1553	897	569	1225	159	1471	815	487	1143	323	1635	979	651	1307	36	1348	692	364	1676	1020	200	1512	856	528	1184	118	1430	774	446	1102	282	1594	938	610	1266	77	1389	733	405	1061	241
549	1205	139	1451	795	467	1123	303	1615	959	631	1287	16	1328	672	344	1656	1000	180	1492	836	508	1164	98	1410	754	426	1082	262	1574	918	590	1246	57	1369	713	385	1041	221	1533	877
149	1461	805	477	1133	313	1625	969	641	1297	26	1338	682	354	1666	1010	190	1502	846	518	1174	108	1420	764	436	1092	272	1584	928	600	1256	67	1379	723	395	1051	231	1543	887	559	1215
785	457	1113	293	1605	949	621	1277	6	1318	662	334	1646	990	170	1482	826	498	1154	88	1400	744	416	1072	252	1564	908	580	1236	47	1359	703	375	1031	211	1523	867	539	1195	129	1441
1138	318	1630	974	646	1302	31	1343	687	359	1671	1015	195	1507	851	523	1179	113	1425	769	441	1097	277	1589	933	605	1261	72	1384	728	400	1056	236	1548	892	564	1220	154	1466	810	482
1610	954	626	1282	11	1323	667	339	1651	995	175	1487	831	503	1159	93	1405	749	421	1077	257	1569	913	585	1241	52	1364	708	380	1036	216	1528	872	544	1200	134	1446	790	462	1118	298
636	1292	21	1333	677	349	1661	1005	185	1497	841	513	1169	103	1415	759	431	1087	267	1579	923	595	1251	62	1374	718	390	1046	226	1538	882	554	1210	144	1456	800	472	1128	308	1620	964
0	1312	656	328	1640	984	164	1476	820	492	1148	82	1394	738	410	1066	246	1558	902	574	1230	41	1353	697	369	1025	205	1517	861	533	1189	123	1435	779	451	1107	287	1599	943	615	3

Table H.19 — 43 x 43 data

2	359	1735	1047	187	1563	875	531	1219	101	1477	789	445	1821	1133	273	1649	961	617	1305	58	1434	746	402	1778	1090	230	1606	918	574	1262	144	1520	832	488	1176	316	1692	1004	660	1348	15	1
1746	1058	198	1574	886	542	1230	112	1488	800	456	1832	1144	284	1660	972	628	1316	69	1445	757	413	1789	1101	241	1617	929	585	1273	155	1531	843	499	1187	327	1703	1015	671	1359	26	1402	714	370
176	1552	864	520	1208	90	1466	778	434	1810	1122	262	1638	950	606	1294	47	1423	735	391	1767	1079	219	1595	907	563	1251	133	1509	821	477	1165	305	1681	993	649	1337	4	1380	692	348	1724	1036
899	555	1243	125	1501	813	469	1845	1157	297	1673	985	641	1329	82	1458	770	426	1802	1114	254	1630	942	598	1286	168	1544	856	512	1200	340	1716	1028	684	1372	39	1415	727	383	1759	1071	211	1587
1222	104	1480	792	448	1824	1136	276	1652	964	620	1308	61	1437	749	405	1781	1093	233	1609	921	577	1265	147	1523	835	491	1179	319	1695	1007	663	1351	18	1394	706	362	1738	1050	190	1566	878	534
1491	803	459	1835	1147	287	1663	975	631	1319	72	1448	760	416	1792	1104	244	1620	932	588	1276	158	1534	846	502	1190	330	1706	1018	674	1362	29	1405	717	373	1749	1061	201	1577	889	545	1233	115
437	1813	1125	265	1641	953	609	1297	50	1426	738	394	1770	1082	222	1598	910	566	1254	136	1512	824	480	1168	308	1684	996	652	1340	7	1383	695	351	1727	1039	179	1555	867	523	1211	93	1469	781
1152	292	1668	980	636	1324	77	1453	765	421	1797	1109	249	1625	937	593	1281	163	1539	851	507	1195	335	1711	1023	679	1367	34	1410	722	378	1754	1066	206	1582	894	550	1238	120	1496	808	464	1840
1646	958	614	1302	55	1431	743	399	1775	1087	227	1603	915	571	1259	141	1517	829	485	1173	313	1689	1001	657	1345	12	1388	700	356	1732	1044	184	1560	872	528	1216	98	1474	786	442	1818	1130	270
625	1313	66	1442	754	410	1786	1098	238	1614	926	582	1270	152	1528	840	496	1184	324	1700	1012	668	1356	23	1399	711	367	1743	1055	195	1571	883	539	1227	109	1485	797	453	1829	1141	281	1657	969
44	1420	732	388	1764	1076	216	1592	904	560	1248	130	1506	818	474	1162	302	1678	990	646	1334	1391	1377	689	345	1721	1033	173	1549	861	517	1205	87	1463	775	431	1807	1119	259	1635	947	603	1291
773	429	1805	1117	257	1633	945	601	1289	171	1547	859	515	1203	343	1719	1031	687	1375	42	1418	730	386	1762	1074	214	1590	902	558	1246	128	1504	816	472	1848	1160	300	1676	988	644	1332	85	1461
1784	1096	236	1612	924	580	1268	150	1526	838	494	1182	322	1698	1010	666	1354	21	1397	709	365	1741	1053	193	1569	881	537	1225	107	1483	795	451	1827	1139	279	1655	967	623	1311	64	1440	752	408
247	1623	935	591	1279	161	1537	849	505	1193	333	1709	1021	677	1365	32	1408	720	376	1752	1064	204	1580	892	548	1236	118	1494	806	462	1838	1150	290	1666	978	634	1322	75	1451	763	419	1795	1107
913	569	1257	139	1515	827	483	1171	311	1687	999	655	1343	10	1386	698	354	1730	1042	182	1558	870	526	1214	96	1472	784	440	1816	1128	268	1644	956	612	1300	53	1429	741	397	1773	1085	225	1601
1284	166	1542	854	510	1198	338	1714	1026	682	1370	37	1413	725	381	1757	1069	209	1585	897	553	1241	123	1499	811	467	1843	1155	295	1671	983	639	1327	80	1456	768	424	1800	1112	252	1628	940	596
1521	833	489	1177	317	1693	1005	661	1349	16	1392	704	360	1736	1048	188	1564	876	532	1220	102	1478	790	446	1822	1134	274	1650	962	618	1306	59	1435	747	403	1779	1091	231	1607	919	575	1263	145
500	1188	328	1704	1016	672	1360	27	1403	715	371	1747	1059	199	1575	887	543	1231	113	1489	801	457	1833	1145	285	1661	973	629	1317	70	1446	758	414	1790	1102	242	1618	930	586	1274	156	1532	844
306	1682	994	650	1338	5	1381	693	349	1725	1037	177	1553	865	521	1209	91	1467	779	435	1811	1123	263	1639	951	607	1295	48	1424	736	392	1768	1080	220	1596	908	564	1252	134	1510	822	478	1166
1029	685	1373	40	1416	728	384	1760	1072	212	1588	900	556	1244	126	1502	814	470	1846	1158	298	1674	986	642	1330	83	1459	771	427	1803	1115	255	1631	943	599	1287	169	1545	857	513	1201	341	1717
1352	19	1395	707	363	1739	1051	191	1567	879	535	1223	105	1481	793	449	1825	1137	277	1653	965	621	1309	62	1438	750	406	1782	1094	234	1610	922	578	1266	148	1524	836	492	1180	320	1696	1008	664
1406	718	374	1750	1062	202	1578	890	546	1234	116	1492	804	460	1836	1148	288	1664	976	632	1320	73	1449	761	417	1793	1105	245	1621	933	589	1277	159	1535	847	503	1191	331	1707	1019	675	1363	30
352	1728	1040	180	1556	868	524	1212	94	1470	782	438	1814	1126	266	1642	954	610	1298	51	1427	739	395	1771	1083	223	1599	911	567	1255	137	1513	825	481	1169	309	1685	997	653	1341	8	1384	696
1067	207	1583	895	551	1239	121	1497	809	465	1841	1153	293	1669	981	637	1325	78	1454	766	422	1798	1110	250	1626	938	594	1282	164	1540	852	508	1196	336	1712	1024	680	1368	35	1411	723	379	1755
1561	873	529	1217	99	1475	787	443	1819	1131	271	1647	959	615	1303	56	1432	744	400	1776	1088	228	1604	916	572	1260	142	1518	830	486	1174	314	1690	1002	658	1346	13	1389	701	357	1733	1045	185
540	1228	110	1486	798	454	1830	1142	282	1658	970	626	1314	67	1443	755	411	1787	1099	239	1615	927	583	1271	153	1529	841	497	1185	325	1701	1013	669	1357	24	1400	712	368	1744	1056	196	1572	884
88	1464	776	432	1808	1120	260	1636	948	604	1292	45	1421	733	389	1765	1077	217	1593	905	561	1249	131	1507	819	475	1163	303	1679	991	647	1335	703	1378	690	346	1722	1034	174	1550	862	518	1206
815	471	1847	1159	299	1675	987	643	1331	84	1460	772	428	1804	1116	256	1632	944	600	1288	170	1546	858	514	1202	342	1718	1030	686	1374	41	1417	729	385	1761	1073	213	1589	901	557	1245	127	1503
1826	1138	278	1654	966	622	1310	63	1439	751	407	1783	1095	235	1611	923	579	1267	149	1525	837	493	1181	321	1697	1009	665	1353	20	1396	708	364	1740	1052	192	1568	880	536	1224	106	1482	794	450
289	1665	977	633	1321	74	1450	762	418	1794	1106	246	1622	934	590	1278	160	1536	848	504	1192	332	1708	1020	676	1364	31	1407	719	375	1751	1063	203	1579	891	547	1235	117	1493	805	461	1837	1149
955	611	1299	52	1428	740	396	1772	1084	224	1600	912	568	1256	138	1514	826	482	1170	310	1686	998	654	1342	9	1385	697	353	1729	1041	181	1557	869	525	1213	95	1471	783	439	1815	1127	267	1643
1326	79	1455	767	423	1799	1111	251	1627	939	595	1283	165	1541	853	509	1197	337	1713	1025	681	1369	36	1412	724	380	1756	1068	208	1584	896	552	1240	122	1498	810	466	1842	1154	294	1670	982	638
1433	745	401	1777	1089	229	1605	917	573	1261	143	1519	831	487	1175	315	1691	1003	659	1347	14	1390	702	358	1734	1046	186	1562	874	530	1218	100	1476	788	444	1820	1132	272	1648	960	616	1304	57
412	1788	1100	240	1616	928	584	1272	154	1530	842	498	1186	326	1702	1014	670	1358	25	1401	713	369	1745	1057	197	1573	885	541	1229	111	1487	799	455	1831	1143	283	1659	971	627	1315	68	1444	756
1078	218	1594	906	562	1250	132	1508	820	476	1164	304	1680	992	648	1336	1333	1379	691	347	1723	1035	175	1551	863	519	1207	89	1465	777	433	1809	1121	261	1637	949	605	1293	46	1422	734	390	1766
1629	941	597	1285	167	1543	855	511	1199	339	1715	1027	683	1371	38	1414	726	382	1758	1070	210	1586	898	554	1242	124	1500	812	468	1844	1156	296	1672	984	640	1328	81	1457	769	425	1801	1113	253
576	1264	146	1522	834	490	1178	318	1694	1006	662	1350	17	1393	705	361	1737	1049	189	1565	877	533	1221	103	1479	791	447	1823	1135	275	1651	963	619	1307	60	1436	748	404	1780	1092	232	1608	920
157	1533	845	501	1189	329	1705	1017	673	1361	28	1404	716	372	1748	1060	200	1576	888	544	1232	114	1490	802	458	1834	1146	286	1662	974	630	1318	71	1447	759	415	1791	1103	243	1619	931	587	1275
823	479	1167	307	1683	995	651	1339	6	1382	694	350	1726	1038	178	1554	866	522	1210	92	1468	780	436	1812	1124	264	1640	952	608	1296	49	1425	737	393	1769	1081	221	1597	909	565	1253	135	1511
1194	334	1710	1022	678	1366	33	1409	721	377	1753	1065	205	1581	893	549	1237	119	1495	807	463	1839	1151	291	1667	979	635	1323	76	1452	764	420	1796	1108	248	1624	936	592	1280	162	1538	850	506
1688	1000	656	1344	11	1387	699	355	1731	1043	183	1559	871	527	1215	97	1473	785	441	1817	1129	269	1645	957	613	1301	54	1430	742	398	1774	1086	226	1602	914	570	1258	140	1516	828	484	1172	312
667	1355	22	1398	710	366	1742	1054	194	1570	882	538	1226	108	1484	796	452	1828	1140	280	1656	968	624	1312	65	1441	753	409	1785	1097	237	1613	925	581	1269	151	1527	839	495	1183	323	1699	1011
0	1376	688	344	1720	1032	172	1548	860	516	1204	86	1462	774	430	1806	1118	258	1634	946	602	1290	43	1419	731	387	1763	1075	215	1591	903	559	1247	129	1505	817	473	1161	301	1677	989	645	3

Table H.20 — 45 x 45 data

2	370	1810	1090	190	1630	910	550	1990	1270	100	1540	820	460	1900	1180	280	1720	1000	640	1360	55	1495	775	415	1855	1135	235	1675	955	595	1315	145	1585	865	505	1945	1225	325	1765	1045	685	1405	10	1
1838	1118	218	1658	938	578	2018	1298	128	1568	848	488	1928	1208	308	1748	1028	668	1388	83	1523	803	443	1883	1163	263	1703	983	623	1343	173	1613	893	533	1973	1253	353	1793	1073	713	1433	38	1478	758	398
196	1636	916	556	1996	1276	106	1546	826	466	1906	1186	286	1726	1006	646	1366	61	1501	781	421	1861	1141	241	1681	961	601	1321	151	1591	871	511	1951	1231	331	1771	1051	691	1411	16	1456	736	376	1816	1096
927	567	2007	1287	117	1557	837	477	1917	1197	297	1737	1017	657	1377	72	1512	792	432	1872	1152	252	1692	972	612	1332	162	1602	882	522	1962	1242	342	1782	1062	702	1422	27	1467	747	387	1827	1107	207	1647
1984	1264	94	1534	814	454	1894	1174	274	1714	994	634	1354	49	1489	769	409	1849	1129	229	1669	949	589	1309	139	1579	859	499	1939	1219	319	1759	1039	679	1399	4	1444	724	364	1804	1084	184	1624	904	544
131	1571	851	491	1931	1211	311	1751	1031	671	1391	86	1526	806	446	1886	1166	266	1706	986	626	1346	176	1616	896	536	1976	1256	356	1796	1076	716	1436	41	1481	761	401	1841	1121	221	1661	941	581	2021	1301
829	469	1909	1189	289	1729	1009	649	1369	64	1504	784	424	1864	1144	244	1684	964	604	1324	154	1594	874	514	1954	1234	334	1774	1054	694	1414	19	1459	739	379	1819	1099	199	1639	919	559	1999	1279	109	1549
1920	1200	300	1740	1020	660	1380	75	1515	795	435	1875	1155	255	1695	975	615	1335	165	1605	885	525	1965	1245	345	1785	1065	705	1425	30	1470	750	390	1830	1110	210	1650	930	570	2010	1290	120	1560	840	480
277	1717	997	637	1357	52	1492	772	412	1852	1132	232	1672	952	592	1312	142	1582	862	502	1942	1222	322	1762	1042	682	1402	7	1447	727	367	1807	1087	187	1627	907	547	1987	1267	97	1537	817	457	1897	1177
1025	665	1385	80	1520	800	440	1880	1160	260	1700	980	620	1340	170	1610	890	530	1970	1250	350	1790	1070	710	1430	35	1475	755	395	1835	1115	215	1655	935	575	2015	1295	125	1565	845	485	1925	1205	305	1745
1363	58	1498	778	418	1858	1138	238	1678	958	598	1318	148	1588	868	508	1948	1228	328	1768	1048	688	1408	13	1453	733	373	1813	1093	193	1633	913	553	1993	1273	103	1543	823	463	1903	1183	283	1723	1003	643
1509	789	429	1869	1149	249	1689	969	609	1329	159	1599	879	519	1959	1239	339	1779	1059	699	1419	24	1464	744	384	1824	1104	204	1644	924	564	2004	1284	114	1554	834	474	1914	1194	294	1734	1014	654	1374	69
406	1846	1126	226	1666	946	586	1306	136	1576	856	496	1936	1216	316	1756	1036	676	1396	1450	1441	721	361	1801	1081	181	1621	901	541	1981	1261	91	1531	811	451	1891	1171	271	1711	991	631	1351	46	1486	766
1169	269	1709	989	629	1349	179	1619	899	539	1979	1259	359	1799	1079	719	1439	44	1484	764	404	1844	1124	224	1664	944	584	2024	1304	134	1574	854	494	1934	1214	314	1754	1034	674	1394	89	1529	809	449	1889
1687	967	607	1327	157	1597	877	517	1957	1237	337	1777	1057	697	1417	22	1462	742	382	1822	1102	202	1642	922	562	2002	1282	112	1552	832	472	1912	1192	292	1732	1012	652	1372	67	1507	787	427	1867	1147	247
618	1338	168	1608	888	528	1968	1248	348	1788	1068	708	1428	33	1473	753	393	1833	1113	213	1653	933	573	2013	1293	123	1563	843	483	1923	1203	303	1743	1023	663	1383	78	1518	798	438	1878	1158	258	1698	978
146	1586	866	506	1946	1226	326	1766	1046	686	1406	11	1451	731	371	1811	1091	191	1631	911	551	1991	1271	101	1541	821	461	1901	1181	281	1721	1001	641	1361	56	1496	776	416	1856	1136	236	1676	956	596	1316
894	534	1974	1254	354	1794	1074	714	1434	39	1479	759	399	1839	1119	219	1659	939	579	2019	1299	129	1569	849	489	1929	1209	309	1749	1029	669	1389	84	1524	804	444	1884	1164	264	1704	984	624	1344	174	1614
1952	1232	332	1772	1052	692	1412	17	1457	737	377	1817	1097	197	1637	917	557	1997	1277	107	1547	827	467	1907	1187	287	1727	1007	647	1367	62	1502	782	422	1862	1142	242	1682	962	602	1322	152	1592	872	512
343	1783	1063	703	1423	28	1468	748	388	1828	1108	208	1648	928	568	2008	1288	118	1558	838	478	1918	1198	298	1738	1018	658	1378	73	1513	793	433	1873	1153	253	1693	973	613	1333	163	1603	883	523	1963	1243
1040	680	1400	5	1445	725	365	1805	1085	185	1625	905	545	1985	1265	95	1535	815	455	1895	1175	275	1715	995	635	1355	50	1490	770	410	1850	1130	230	1670	950	590	1310	140	1580	860	500	1940	1220	320	1760
1437	42	1482	762	402	1842	1122	222	1662	942	582	2022	1302	132	1572	852	492	1932	1212	312	1752	1032	672	1392	87	1527	807	447	1887	1167	267	1707	987	627	1347	177	1617	897	537	1977	1257	357	1797	1077	717
1460	740	380	1820	1100	200	1640	920	560	2000	1280	110	1550	830	470	1910	1190	290	1730	1010	650	1370	65	1505	785	425	1865	1145	245	1685	965	605	1325	155	1595	875	515	1955	1235	335	1775	1055	695	1415	20
391	1831	1111	211	1651	931	571	2011	1291	121	1561	841	481	1921	1201	301	1741	1021	661	1381	76	1516	796	436	1876	1156	256	1696	976	616	1336	166	1606	886	526	1966	1246	346	1786	1066	706	1426	31	1471	751
1088	188	1628	908	548	1988	1268	98	1538	818	458	1898	1178	278	1718	998	638	1358	53	1493	773	413	1853	1133	233	1673	953	593	1313	143	1583	863	503	1943	1223	323	1763	1043	683	1403	8	1448	728	368	1808
1656	936	576	2016	1296	126	1566	846	486	1926	1206	306	1746	1026	666	1386	81	1521	801	441	1881	1161	261	1701	981	621	1341	171	1611	891	531	1971	1251	351	1791	1071	711	1431	36	1476	756	396	1836	1116	216
554	1994	1274	104	1544	824	464	1904	1184	284	1724	1004	644	1364	59	1499	779	419	1859	1139	239	1679	959	599	1319	149	1589	869	509	1949	1229	329	1769	1049	689	1409	14	1454	734	374	1814	1094	194	1634	914
1285	115	1555	835	475	1915	1195	295	1735	1015	655	1375	70	1510	790	430	1870	1150	250	1690	970	610	1330	160	1600	880	520	1960	1240	340	1780	1060	700	1420	25	1465	745	385	1825	1105	205	1645	925	565	2005
1532	812	452	1892	1172	272	1712	992	632	1352	47	1487	767	407	1847	1127	227	1667	947	587	1307	137	1577	857	497	1937	1217	317	1757	1037	677	1397	730	1442	722	362	1802	1082	182	1622	902	542	1982	1262	92
493	1933	1213	313	1753	1033	673	1393	88	1528	808	448	1888	1168	268	1708	988	628	1348	178	1618	898	538	1978	1258	358	1798	1078	718	1438	43	1483	763	403	1843	1123	223	1663	943	583	2023	1303	133	1573	853
1191	291	1731	1011	651	1371	66	1506	786	426	1866	1146	246	1686	966	606	1326	156	1596	876	516	1956	1236	336	1776	1056	696	1416	21	1461	741	381	1821	1101	201	1641	921	561	2001	1281	111	1551	831	471	1911
1742	1022	662	1382	77	1517	797	437	1877	1157	257	1697	977	617	1337	167	1607	887	527	1967	1247	347	1787	1067	707	1427	32	1472	752	392	1832	1112	212	1652	932	572	2012	1292	122	1562	842	482	1922	1202	302
639	1359	54	1494	774	414	1854	1134	234	1674	954	594	1314	144	1584	864	504	1944	1224	324	1764	1044	684	1404	9	1449	729	369	1809	1089	189	1629	909	549	1989	1269	99	1539	819	459	1899	1179	279	1719	999
82	1522	802	442	1882	1162	262	1702	982	622	1342	172	1612	892	532	1972	1252	352	1792	1072	712	1432	37	1477	757	397	1837	1117	217	1657	937	577	2017	1297	127	1567	847	487	1927	1207	307	1747	1027	667	1387
780	420	1860	1140	240	1680	960	600	1320	150	1590	870	510	1950	1230	330	1770	1050	690	1410	15	1455	735	375	1815	1095	195	1635	915	555	1995	1275	105	1545	825	465	1905	1185	285	1725	1005	645	1365	60	1500
1871	1151	251	1691	971	611	1331	161	1601	881	521	1961	1241	341	1781	1061	701	1421	26	1466	746	386	1826	1106	206	1646	926	566	2006	1286	116	1556	836	476	1916	1196	296	1736	1016	656	1376	71	1511	791	431
228	1668	948	588	1308	138	1578	858	498	1938	1218	318	1758	1038	678	1398	1395	1443	723	363	1803	1083	183	1623	903	543	1983	1263	93	1533	813	453	1893	1173	273	1713	993	633	1353	48	1488	768	408	1848	1128
985	625	1345	175	1615	895	535	1975	1255	355	1795	1075	715	1435	40	1480	760	400	1840	1120	220	1660	940	580	2020	1300	130	1570	850	490	1930	1210	310	1750	1030	670	1390	85	1525	805	445	1885	1165	265	1705
1323	153	1593	873	513	1953	1233	333	1773	1053	693	1413	18	1458	738	378	1818	1098	198	1638	918	558	1998	1278	108	1548	828	468	1908	1188	288	1728	1008	648	1368	63	1503	783	423	1863	1143	243	1683	963	603
1604	884	524	1964	1244	344	1784	1064	704	1424	29	1469	749	389	1829	1109	209	1649	929	569	2009	1289	119	1559	839	479	1919	1199	299	1739	1019	659	1379	74	1514	794	434	1874	1154	254	1694	974	614	1334	164
501	1941	1221	321	1761	1041	681	1401	6	1446	726	366	1806	1086	186	1626	906	546	1986	1266	96	1536	816	456	1896	1176	276	1716	996	636	1356	51	1491	771	411	1851	1131	231	1671	951	591	1311	141	1581	861
1249	349	1789	1069	709	1429	34	1474	754	394	1834	1114	214	1654	934	574	2014	1294	124	1564	844	484	1924	1204	304	1744	1024	664	1384	79	1519	799	439	1879	1159	259	1699	979	619	1339	169	1609	889	529	1969
1767	1047	687	1407	12	1452	732	372	1812	1092	192	1632	912	552	1992	1272	102	1542	822	462	1902	1182	282	1722	1002	642	1362	57	1497	777	417	1857	1137	237	1677	957	597	1317	147	1587	867	507	1947	1227	327
698	1418	23	1463	743	383	1823	1103	203	1643	923	563	2003	1283	113	1553	833	473	1913	1193	293	1733	1013	653	1373	68	1508	788	428	1868	1148	248	1688	968	608	1328	158	1598	878	518	1958	1238	338	1778	1058
0	1440	720	360	1800	1080	180	1620	900	540	1980	1260	90	1530	810	450	1890	1170	270	1710	990	630	1350	45	1485	765	405	1845	1125	225	1665	945	585	1305	135	1575	855	495	1935	1215	315	1755	1035	675	3

Table H.21 — 47 x 47 data

2	398	1902	1150	210	1714	962	586	2090	1338	116	1620	868	492	1996	1244	304	1808	1056	680	2184	1432	69	1573	821	445	1949	1197	257	1761	1009	633	2137	1385	163	1667	915	539	2043	1291	351	1855	1103	727	1479	22	1
1914	1162	222	1726	974	598	2102	1350	128	1632	880	504	2008	1256	316	1820	1068	692	2196	1444	81	1585	833	457	1961	1209	269	1773	1021	645	2149	1397	175	1679	927	551	2055	1303	363	1867	1115	739	1491	34	1538	786	410
198	1702	950	574	2078	1326	104	1608	856	480	1984	1232	292	1796	1044	668	2172	1420	57	1561	809	433	1937	1185	245	1749	997	621	2125	1373	151	1655	903	527	2031	1279	339	1843	1091	715	1467	10	1514	762	386	1890	1138
980	604	2108	1356	134	1638	886	510	2014	1262	322	1826	1074	698	2202	1450	87	1591	839	463	1967	1215	275	1779	1027	651	2155	1403	181	1685	933	557	2061	1309	369	1873	1121	745	1497	40	1544	792	416	1920	1168	228	1732
2084	1332	110	1614	862	486	1990	1238	298	1802	1050	674	2178	1426	63	1567	815	439	1943	1191	251	1755	1003	627	2131	1379	157	1661	909	533	2037	1285	345	1849	1097	721	1473	16	1520	768	392	1896	1144	204	1708	956	580
122	1626	874	498	2002	1250	310	1814	1062	686	2190	1438	75	1579	827	451	1955	1203	263	1767	1015	639	2143	1391	169	1673	921	545	2049	1297	357	1861	1109	733	1485	28	1532	780	404	1908	1156	216	1720	968	592	2096	1344
850	474	1978	1226	286	1790	1038	662	2166	1414	51	1555	803	427	1931	1179	239	1743	991	615	2119	1367	145	1649	897	521	2025	1273	333	1837	1085	709	1461	4	1508	756	380	1884	1132	192	1696	944	568	2072	1320	98	1602
2017	1265	325	1829	1077	701	2205	1453	90	1594	842	466	1970	1218	278	1782	1030	654	2158	1406	184	1688	936	560	2064	1312	372	1876	1124	748	1500	43	1547	795	419	1923	1171	231	1735	983	607	2111	1359	137	1641	889	513
301	1805	1053	677	2181	1429	66	1570	818	442	1946	1194	254	1758	1006	630	2134	1382	160	1664	912	536	2040	1288	348	1852	1100	724	1476	19	1523	771	395	1899	1147	207	1711	959	583	2087	1335	113	1617	865	489	1993	1241
1065	689	2193	1441	78	1582	830	454	1958	1206	266	1770	1018	642	2146	1394	172	1676	924	548	2052	1300	360	1864	1112	736	1488	31	1535	783	407	1911	1159	219	1723	971	595	2099	1347	125	1629	877	501	2005	1253	313	1817
2169	1417	54	1558	806	430	1934	1182	242	1746	994	618	2122	1370	148	1652	900	524	2028	1276	336	1840	1088	712	1464	7	1511	759	383	1887	1135	195	1699	947	571	2075	1323	101	1605	853	477	1981	1229	289	1793	1041	665
84	1588	836	460	1964	1212	272	1776	1024	648	2152	1400	178	1682	930	554	2058	1306	366	1870	1118	742	1494	37	1541	789	413	1917	1165	225	1729	977	601	2105	1353	131	1635	883	507	2011	1259	319	1823	1071	695	2199	1447
812	436	1940	1188	248	1752	1000	624	2128	1376	154	1658	906	530	2034	1282	342	1846	1094	718	1470	13	1517	765	389	1893	1141	201	1705	953	577	2081	1329	107	1611	859	483	1987	1235	295	1799	1047	671	2175	1423	60	1564
1952	1200	260	1764	1012	636	2140	1388	166	1670	918	542	2046	1294	354	1858	1106	730	1482	25	1529	777	401	1905	1153	213	1717	965	589	2093	1341	119	1623	871	495	1999	1247	307	1811	1059	683	2187	1435	72	1576	824	448
236	1740	988	612	2116	1364	142	1646	894	518	2022	1270	330	1834	1082	706	1458	1526	1505	753	377	1881	1129	189	1693	941	565	2069	1317	95	1599	847	471	1975	1223	283	1787	1035	659	2163	1411	48	1552	800	424	1928	1176
1033	657	2161	1409	187	1691	939	563	2067	1315	375	1879	1127	751	1503	46	1550	798	422	1926	1174	234	1738	986	610	2114	1362	140	1644	892	516	2020	1268	328	1832	1080	704	2208	1456	93	1597	845	469	1973	1221	281	1785
2138	1386	164	1668	916	540	2044	1292	352	1856	1104	728	1480	23	1527	775	399	1903	1151	211	1715	963	587	2091	1339	117	1621	869	493	1997	1245	305	1809	1057	681	2185	1433	70	1574	822	446	1950	1198	258	1762	1010	634
176	1680	928	552	2056	1304	364	1868	1116	740	1492	35	1539	787	411	1915	1163	223	1727	975	599	2103	1351	129	1633	881	505	2009	1257	317	1821	1069	693	2197	1445	82	1586	834	458	1962	1210	270	1774	1022	646	2150	1398
904	528	2032	1280	340	1844	1092	716	1468	11	1515	763	387	1891	1139	199	1703	951	575	2079	1327	105	1609	857	481	1985	1233	293	1797	1045	669	2173	1421	58	1562	810	434	1938	1186	246	1750	998	622	2126	1374	152	1656
2062	1310	370	1874	1122	746	1498	41	1545	793	417	1921	1169	229	1733	981	605	2109	1357	135	1639	887	511	2015	1263	323	1827	1075	699	2203	1451	88	1592	840	464	1968	1216	276	1780	1028	652	2156	1404	182	1686	934	558
346	1850	1098	722	1474	17	1521	769	393	1897	1145	205	1709	957	581	2085	1333	111	1615	863	487	1991	1239	299	1803	1051	675	2179	1427	64	1568	816	440	1944	1192	252	1756	1004	628	2132	1380	158	1662	910	534	2038	1286
1110	734	1486	29	1533	781	405	1909	1157	217	1721	969	593	2097	1345	123	1627	875	499	2003	1251	311	1815	1063	687	2191	1439	76	1580	828	452	1956	1204	264	1768	1016	640	2144	1392	170	1674	922	546	2050	1298	358	1862
1462	5	1509	757	381	1885	1133	193	1697	945	569	2073	1321	99	1603	851	475	1979	1227	287	1791	1039	663	2167	1415	52	1556	804	428	1932	1180	240	1744	992	616	2120	1368	146	1650	898	522	2026	1274	334	1838	1086	710
1548	796	420	1924	1172	232	1736	984	608	2112	1360	138	1642	890	514	2018	1266	326	1830	1078	702	2206	1454	91	1595	843	467	1971	1219	279	1783	1031	655	2159	1407	185	1689	937	561	2065	1313	373	1877	1125	749	1501	44
396	1900	1148	208	1712	960	584	2088	1336	114	1618	866	490	1994	1242	302	1806	1054	678	2182	1430	67	1571	819	443	1947	1195	255	1759	1007	631	2135	1383	161	1665	913	537	2041	1289	349	1853	1101	725	1477	20	1524	772
1160	220	1724	972	596	2100	1348	126	1630	878	502	2006	1254	314	1818	1066	690	2194	1442	79	1583	831	455	1959	1207	267	1771	1019	643	2147	1395	173	1677	925	549	2053	1301	361	1865	1113	737	1489	32	1536	784	408	1912
1700	948	572	2076	1324	102	1606	854	478	1982	1230	290	1794	1042	666	2170	1418	55	1559	807	431	1935	1183	243	1747	995	619	2123	1371	149	1653	901	525	2029	1277	337	1841	1089	713	1465	8	1512	760	384	1888	1136	196
602	2106	1354	132	1636	884	508	2012	1260	320	1824	1072	696	2200	1448	85	1589	837	461	1965	1213	273	1777	1025	649	2153	1401	179	1683	931	555	2059	1307	367	1871	1119	743	1495	38	1542	790	414	1918	1166	226	1730	978
1330	108	1612	860	484	1988	1236	296	1800	1048	672	2176	1424	61	1565	813	437	1941	1189	249	1753	1001	625	2129	1377	155	1659	907	531	2035	1283	343	1847	1095	719	1471	14	1518	766	390	1894	1142	202	1706	954	578	2082
1624	872	496	2000	1248	308	1812	1060	684	2188	1436	73	1577	825	449	1953	1201	261	1765	1013	637	2141	1389	167	1671	919	543	2047	1295	355	1859	1107	731	1483	26	1530	778	402	1906	1154	214	1718	966	590	2094	1342	120
472	1976	1224	284	1788	1036	660	2164	1412	49	1553	801	425	1929	1177	237	1741	989	613	2117	1365	143	1647	895	519	2023	1271	331	1835	1083	707	1459	774	1506	754	378	1882	1130	190	1694	942	566	2070	1318	96	1600	848
1267	327	1831	1079	703	2207	1455	92	1596	844	468	1972	1220	280	1784	1032	656	2160	1408	186	1690	938	562	2066	1314	374	1878	1126	750	1502	45	1549	797	421	1925	1173	233	1737	985	609	2113	1361	139	1643	891	515	2019
1807	1055	679	2183	1431	68	1572	820	444	1948	1196	256	1760	1008	632	2136	1384	162	1666	914	538	2042	1290	350	1854	1102	726	1478	21	1525	773	397	1901	1149	209	1713	961	585	2089	1337	115	1619	867	491	1995	1243	303
691	2195	1443	80	1584	832	456	1960	1208	268	1772	1020	644	2148	1396	174	1678	926	550	2054	1302	362	1866	1114	738	1490	33	1537	785	409	1913	1161	221	1725	973	597	2101	1349	127	1631	879	503	2007	1255	315	1819	1067
1419	56	1560	808	432	1936	1184	244	1748	996	620	2124	1372	150	1654	902	526	2030	1278	338	1842	1090	714	1466	9	1513	761	385	1889	1137	197	1701	949	573	2077	1325	103	1607	855	479	1983	1231	291	1795	1043	667	2171
1590	838	462	1966	1214	274	1778	1026	650	2154	1402	180	1684	932	556	2060	1308	368	1872	1120	744	1496	39	1543	791	415	1919	1167	227	1731	979	603	2107	1355	133	1637	885	509	2013	1261	321	1825	1073	697	2201	1449	86
438	1942	1190	250	1754	1002	626	2130	1378	156	1660	908	532	2036	1284	344	1848	1096	720	1472	15	1519	767	391	1895	1143	203	1707	955	579	2083	1331	109	1613	861	485	1989	1237	297	1801	1049	673	2177	1425	62	1566	814
1202	262	1766	1014	638	2142	1390	168	1672	920	544	2048	1296	356	1860	1108	732	1484	27	1531	779	403	1907	1155	215	1719	967	591	2095	1343	121	1625	873	497	2001	1249	309	1813	1061	685	2189	1437	74	1578	826	450	1954
1742	990	614	2118	1366	144	1648	896	520	2024	1272	332	1836	1084	708	1460	1457	1507	755	379	1883	1131	191	1695	943	567	2071	1319	97	1601	849	473	1977	1225	285	1789	1037	661	2165	1413	50	1554	802	426	1930	1178	238
653	2157	1405	183	1687	935	559	2063	1311	371	1875	1123	747	1499	42	1546	794	418	1922	1170	230	1734	982	606	2110	1358	136	1640	888	512	2016	1264	324	1828	1076	700	2204	1452	89	1593	841	465	1969	1217	277	1781	1029
1381	159	1663	911	535	2039	1287	347	1851	1099	723	1475	18	1522	770	394	1898	1146	206	1710	958	582	2086	1334	112	1616	864	488	1992	1240	300	1804	1052	676	2180	1428	65	1569	817	441	1945	1193	253	1757	1005	629	2133
1675	923	547	2051	1299	359	1863	1111	735	1487	30	1534	782	406	1910	1158	218	1722	970	594	2098	1346	124	1628	876	500	2004	1252	312	1816	1064	688	2192	1440	77	1581	829	453	1957	1205	265	1769	1017	641	2145	1393	171
523	2027	1275	335	1839	1087	711	1463	6	1510	758	382	1886	1134	194	1698	946	570	2074	1322	100	1604	852	476	1980	1228	288	1792	1040	664	2168	1416	53	1557	805	429	1933	1181	241	1745	993	617	2121	1369	147	1651	899
1305	365	1869	1117	741	1493	36	1540	788	412	1916	1164	224	1728	976	600	2104	1352	130	1634	882	506	2010	1258	318	1822	1070	694	2198	1446	83	1587	835	459	1963	1211	271	1775	1023	647	2151	1399	177	1681	929	553	2057
1845	1093	717	1469	12	1516	764	388	1892	1140	200	1704	952	576	2080	1328	106	1610	858	482	1986	1234	294	1798	1046	670	2174	1422	59	1563	811	435	1939	1187	247	1751	999	623	2127	1375	153	1657	905	529	2033	1281	341
729	1481	24	1528	776	400	1904	1152	212	1716	964	588	2092	1340	118	1622	870	494	1998	1246	306	1810	1058	682	2186	1434	71	1575	823	447	1951	1199	259	1763	1011	635	2139	1387	165	1669	917	541	2045	1293	353	1857	1105
0	1504	752	376	1880	1128	188	1692	940	564	2068	1316	94	1598	846	470	1974	1222	282	1786	1034	658	2162	1410	47	1551	799	423	1927	1175	235	1739	987	611	2115	1363	141	1645	893	517	2021	1269	329	1833	1081	705	3

Annex I
(normative)

ECC 000 - 140 character encodation schemes

This Annex provides details of the ASCII character set (ISO/IEC 646) used for one of the ECC 000 - 140 encodation schemes, and the four encodation schemes showing the mapping of the data character to the encodation scheme code value.

Table I.1 — Mapping of data character value to encodation scheme value

ASCII SET		ENCODATION SCHEME			
Character	**Decimal value**	**Base 11 code value**	**Base 27 code value**	**Base 37 code value**	**Base 41 code value**
NUL	0				
SOH	1				
STX	2				
ETX	3				
EOT	4				
ENQ	5				
ACK	6				
BEL	7				
BS	8				
HT	9				
LF	10				
VT	11				
FF	12				
CR	13				
SO	14				
SI	15				
DLE	16				
DC1	17				
DC2	18				
DC3	19				
DC4	20				
NAK	21				
SYN	22				
ETB	23				
CAN	24				
EM	25				

ASCII SET		ENCODATION SCHEME			
Character	**Decimal value**	**Base 11 code value**	**Base 27 code value**	**Base 37 code value**	**Base 41 code value**
SUB	26				
ESC	27				
FS	28				
GS	29				
RS	30				
US	31				
space	32	0	0	0	0
!	33				
“	34				
#	35				
$	36				
%	37				
&	38				
‘	39				
(	40				
)	41				
*	42				
+	43				
,	44				38
-	45				39
.	46				37
/	47				40
0	48	1		27	27
1	49	2		28	28
2	50	3		29	29
3	51	4		30	30
4	52	5		31	31
5	53	6		32	32
6	54	7		33	33
7	55	8		34	34
8	56	9		35	35
9	57	10		36	36
:	58				
;	59				
<	60				

ASCII SET		ENCODATION SCHEME			
Character	Decimal value	Base 11 code value	Base 27 code value	Base 37 code value	Base 41 code value
=	61				
>	62				
?	63				
@	64				
A	65		1	1	1
B	66		2	2	2
C	67		3	3	3
D	68		4	4	4
E	69		5	5	5
F	70		6	6	6
G	71		7	7	7
H	72		8	8	8
I	73		9	9	9
J	74		10	10	10
K	75		11	11	11
L	76		12	12	12
M	77		13	13	13
N	78		14	14	14
O	79		15	15	15
P	80		16	16	16
Q	81		17	17	17
R	82		18	18	18
S	83		19	19	19
T	84		20	20	20
U	85		21	21	21
V	86		22	22	22
W	87		23	23	23
X	88		24	24	24
Y	89		25	25	25
Z	90		26	26	26
[	91				
\	92				
]	93				
^	94				
-	95				

ASCII SET		ENCODATION SCHEME			
Character	Decimal value	Base 11 code value	Base 27 code value	Base 37 code value	Base 41 code value
‘	96				
a	97				
b	98				
c	99				
d	100				
e	101				
f	102				
g	103				
h	104				
i	105				
j	106				
k	107				
l	108				
m	109				
n	110				
o	111				
p	112				
q	113				
r	114				
s	115				
t	116				
u	117				
v	118				
w	119				
x	120				
y	121				
z	122				
{	123				
|	124				
}	125				
~	126				
DEL	127				

I.1 Base 11 encodation scheme

I.1.1 First stage procedure

The data characters shall be converted to their Base 11 code values using Table I.1 as the conversion table.

I.1.2 Second stage procedure

The following procedure shall be used to compact the Base 11 code values to a binary string.

a) Sub-divide the number of Base 11 characters into a sequence of six characters, from left to right. If less than six characters go to Step 5.

b) Assign the code values of the six Base 11 characters as C_1 to C_6 , where C_1 is the first character.

c) Carry out a Base 11 to Base 2 conversion to produce a sequence of 21 bits, using equation 6 of Table I.2.

d) Repeat from step a) as necessary.

e) When there are less than six characters, carry out a Base 11 to Base 2 conversion using the appropriate equation of Table I.2 which corresponds to the number of remaining Base 11 characters.

Table I.2 — Base 11 (Numeric) encodation equations

Number of data characters	Encodation equation	Bit length
1	C_1	4
2	$C_1 + C_2 * 11$	7
3	$C_1 + C_2 * 11 + C_3 * 11^2$	11
4	$C_1 + C_2 * 11 + C_3 * 11^2 + C_4 * 11^3$	14
5	$C_1 + C_2 * 11 + C_3 * 11^2 + C_4 * 11^3 + C_5 * 11^4$	18
6	$C_1 + C_2 * 11 + C_3 * 11^2 + C_4 * 11^3 + C_5 * 11^4 + C_6 * 11^5$	21

I.1.3 Example

Using the data character string: 123<space>45678 the complete Base 11 encodation process is shown in Figure I.1.

Data	**1**	**2**	**3**	**<space>**	**4**	**5**	**6**	**7**	**8**
Base 11 code value	2	3	4	0	5	6	7	8	9
Character position	C_1	C_2	C_3	C_4	C_5	C_6	C_1	C_2	C_3
Weight	1	11	121	1331	14641	161051	1	11	121
Product	2	33	484	0	73205	966306	7	88	1089
Decimal value	1040030						1184		
Binary string	011111101111010011110						10010100000		

Figure I.1 — Base 11 example

I.2 Base 27 encodation scheme

I.2.1 First stage procedure

The data characters shall be converted to their Base 27 code values using Table I.1 as the conversion table.

I.2.2 Second stage procedure

The following procedure shall be used to compact the Base 27 code values to a binary string.

a) Sub-divide the number of Base 27 characters into a sequence of five characters, from left to right. If less than five characters go to Step 5.

b) Assign the code values of the five Base 27 characters as C_1 to C_5, where C_1 is the first character.

c) Carry out a Base 27 to Base 2 conversion to produce a sequence of 24 bits, using equation 5 of Table I.3.

d) Repeat from step a) as necessary.

e) When there are less than five characters, carry out a Base 27 to Base 2 conversion using the appropriate equation of Table I.3 which corresponds to the number of remaining Base 27 characters.

Table I.3 — Base 27 (Upper-case Alphabetic) encodation equations

Number of data characters	Encodation equation	Bit length
1	C_1	5
2	$C_1 + C_2 * 27$	10
3	$C_1 + C_2 * 27 + C_3 * 27^2$	15
4	$C_1 + C_2 * 27 + C_3 * 27^2 + C_4 * 27^3$	20
5	$C_1 + C_2 * 27 + C_3 * 27^2 + C_4 * 27^3 + C_5 * 27^4$	24

I.2.3 Example

Using the data character string: DATA<space>MATRIX the complete Base 27 encodation process is shown in Figure I.2.

Data	**D**	**A**	**T**	**A**	**<space>**	**M**	**A**	**T**	**R**	**I**	**X**
Base 27 code value	4	1	20	1	0	13	1	20	18	9	24
Character position	C_1	C_2	C_3	C_4	C_5	C_1	C_2	C_3	C_4	C_5	C_1
Weight	1	27	729	19683	531441	1	27	729	19683	531441	1
Product	4	27	14580	19683	0	13	27	14580	354294	4782969	24
Decimal Value	34294					5151883					24
Binary String	000000001000010111110110					010011101001110010001011					11000

Figure I.2 — Base 27 example

I.3 Base 37 encodation scheme

I.3.1 First stage procedure

The data characters shall be converted to their Base 37 code values using Table I.1 as the conversion table.

I.3.2 Second stage procedure

The following procedure shall be used to compact the Base 37 code values to a binary string.

a) Sub-divide the number of Base 37 characters into a sequence of four characters, from left to right. If less than four characters go to Step 5.

b) Assign the code values of the four Base 37 characters as C_1 to C_4, where C_1 is the first character.

c) Carry out a Base 37 to Base 2 conversion to produce a sequence of 21 bits, using equation 4 of Table I.4.

d) Repeat from step a) as necessary.

e) When there are less than four characters, carry out a Base 37 to Base 2 conversion using the equation (1 to 3) of Table I.4 which corresponds to the number of remaining Base 37 characters.

Table I.4 — Base 37 (Upper-case Alphanumeric) encodation equations

Number of data characters	Encodation equation	Bit length
1	C_1	6
2	$C_1 + C_2 * 37$	11
3	$C_1 + C_2 * 37 + C_3 * 37^2$	16
4	$C_1 + C_2 * 37 + C_3 * 37^2 + C_4 * 37^3$	21

I.3.3 Example

Using the data character string:

123ABCD89

the complete Base 37 encodation process is shown in Figure I.3.

Data	**1**	**2**	**3**	**A**	**B**	**C**	**D**	**8**	**9**
Base 37 code value	28	29	30	1	2	3	4	35	36
Character position	C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4	C_1
Weight	1	37	1369	50653	1	37	1369	50653	1
Product	28	1073	41070	50653	2	111	5476	1772855	36
Decimal value	92824				1778444				36
Binary string	000010110101010011000				110110010001100001100				100100

Figure I.3 — Base 37 example

I.4 Base 41 encodation scheme

I.4.1 First stage procedure

The data characters shall be converted to their Base 41 code values using Table I.1 as the conversion table.

I.4.2 Second stage procedure

The following procedure shall be used to compact the Base 41 code values to a binary string.

a) Sub-divide the number of Base 41 characters into a sequence of four characters, from left to right. If less than four characters go to Step 5.

b) Assign the code values of the four Base 41 characters as C_1 to C_4, where C_1 is the first character.

c) Carry out a Base 41 to Base 2 conversion to produce a sequence of 22 bits, using equation 4 of Table I.5.

d) Repeat from step a) as necessary.

e) When there are less than four characters, carry out a Base 41 to Base 2 conversion using the appropriate equation of Table I.5 which corresponds to the number of remaining Base 41 characters.

Table I.5 — Base 41 (Upper-case alphanumeric + punctuation) encodation equations

Number of data characters	Encodation equation	Bit length
1	C_1	6
2	$C_1 + C_2 * 41$	11
3	$C_1 + C_2 * 41 + C_3 * 41^2$	17
4	$C_1 + C_2 * 41 + C_3 * 41^2 + C_4 * 41^3$	22

I.4.3 Example

Using the data character string:

AB/C123-X

the complete Base 41 encodation process is shown in Figure I.4.

Data	A	B	/	C	1	2	3	-	X
Base 41 code value	1	2	40	3	28	29	30	39	24
Character position	C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4	C_1
Weight	1	41	1681	68921	1	41	1681	68921	1
Product	1	82	67240	206763	28	1189	50430	2687919	24
Decimal value	274086				2739566				24
Binary string	0001000010111010100110				1010011100110101101110				011000

Figure I.4 — Base 41 example

Annex J
(normative)

ECC 000 - 140 CRC algorithm

Following are two implementations for representing CRC.

J.1 CRC state machine

The CRC may be represented as a schematic, as illustrated in Figure J.1. After the data bits have been shifted through the state machine the resulting CRC is read out of the 16 memory registers (m) in the diagram (left most register is the MSB).

J.2 CRC polynomial

The CRC algorithm shall be the CCITT standard polynomial:

$$X^{16} + X^{12} + X^{5} + 1$$

With $X = 2$, the value of the polynomial shown as a 17 bit value is:

$$10001000000100001_{\text{base 2}}$$

The CRC is the remainder after dividing the data string by this value.

J.3 CRC 2-byte header

The CRC calculation headers, as defined in Table J.1, are used in the CRC operation as a prefix to the 8-bit byte values of the data characters. The CRC 2-byte header is shifted into the state machine prior to the calculation of the CRC.

Table J.1 — CRC calculation header

Format ID	Encodation scheme	CRC calculation header		
		MS Byte	LS Byte	Hex
1	Base 11	00000001	00000000	01 00
2	Base 27	00000010	00000000	02 00
3	Base 41	00000011	00000000	03 00
4	Base 37	00000100	00000000	04 00
5	ASCII	00000101	00000000	05 00
6	8-bit Byte	00000110	00000000	06 00

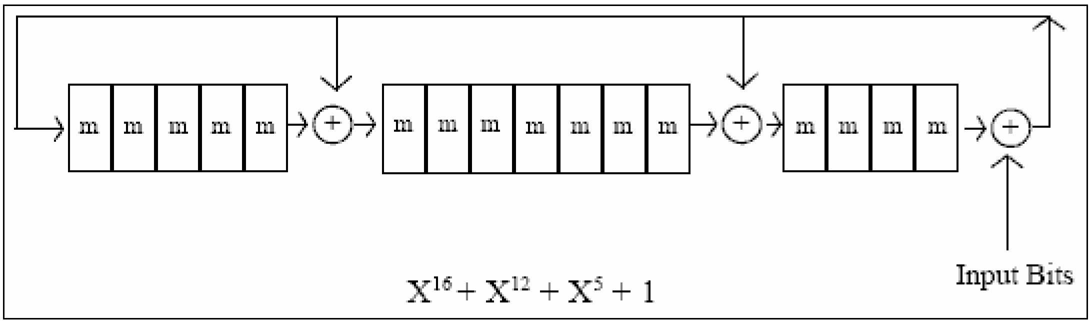


Figure J.1 — CRC algorithm schematic

Annex K
(normative)

ECC 000 - 140 error checking and correcting algorithms

K.1 ECC 000

This provides no error correction.

K.2 ECC 050

The error correction bit stream 'v' for ECC 050 shall be created by processing the unprotected bit stream 'u' through a state machine suitable for a convolutional code of the structure 4-3-3, as illustrated in Figure K.1.

K.3 ECC 080

The error correction bit stream 'v' for ECC 080 shall be created by processing the unprotected bit stream 'u' through a state machine suitable for a convolutional code of the structure 3-2-11, as illustrated in Figure K.2.

K.4 ECC 100

The error correction bit stream 'v' for ECC 100 shall be created by processing the unprotected bit stream 'u' through a state machine suitable for a convolutional code of the structure 2-1-15, as illustrated in Figure K.3.

K.5 ECC 140

The error correction bit stream 'v' for ECC 140 shall be created by processing the unprotected bit stream 'u' through a state machine suitable for a convolutional code of the structure 4-1-13, as illustrated in Figure K.4.

K.6 Processing the convolutional code

In the state machine circuit diagrams, the following notation is used:

$\boxed{m}$ represents a single bit storage register

$\boxed{+}$ represents a one bit binary adder which outputs the lowest bit. It is equivalent to an odd parity generator.

or such adjoining lines are connected

such intersecting lines are not connected

The state machine is operated as follows:

a) The memory storage registers (m) are filled with a zero value before starting the process.

b) An input cycle is performed, consisting of passing a user data bit through the input switch to a memory storage register (m) for each possible input switch position, i.e. for *k* bits.

c) Once a complete set of *k* input bits has been entered, an output cycle is performed. An output cycle consists of reading out an error corrected bit for each possible output switch position, i.e. for *n* bits. At each position, the output bit is computed by performing an XOR operation on the connected memory storage register values.

d) After one input and output cycle, a shift operation is performed by shifting all memory storage register values to the right by one position.

e) Steps b) through d) are repeated until all raw data bits have been input. At the end:

   1) Some zero bits may need to be added to the end of the last segment of input bits to ensure that *k* bits are input.

   2) Sufficient additional zero bits shall be input to ensure that the *m* memory storage registers shall all return to zero values. The output from steps e) 1) and e) 2) is part of the encoded data. The process is complete when all true data bits have passed through the last (rightmost) memory storage register.

K.7 Convolutional codes reference decode algorithm

The Fano algorithm can be used for error correction of data protected by convolutional codes. A basic description of the operation of the Fano algorithm is given in Lin and Costello (see Bibliography). The following guidelines should be used in constructing a convolutional coding decoder.

The start-up variable values must be as follows:

Backward Metric = maximum negative number

Current Metric = 0

Forward Metric = 0

Threshold = 0

The metric is computed by determining the number of bits that are different between the damaged block and the candidate match block:

Metric = (1 * correct bits) - (penalty * incorrect)

Table K1 presents values for the Single Bit Penalty and Delta which should be used when decoding each of the ECC levels.

Table K.1 — Fano algorithm coefficients

ECC level	Single bit penalty	Delta
050	31	20
080	16	11
100	8	6
140	4	1

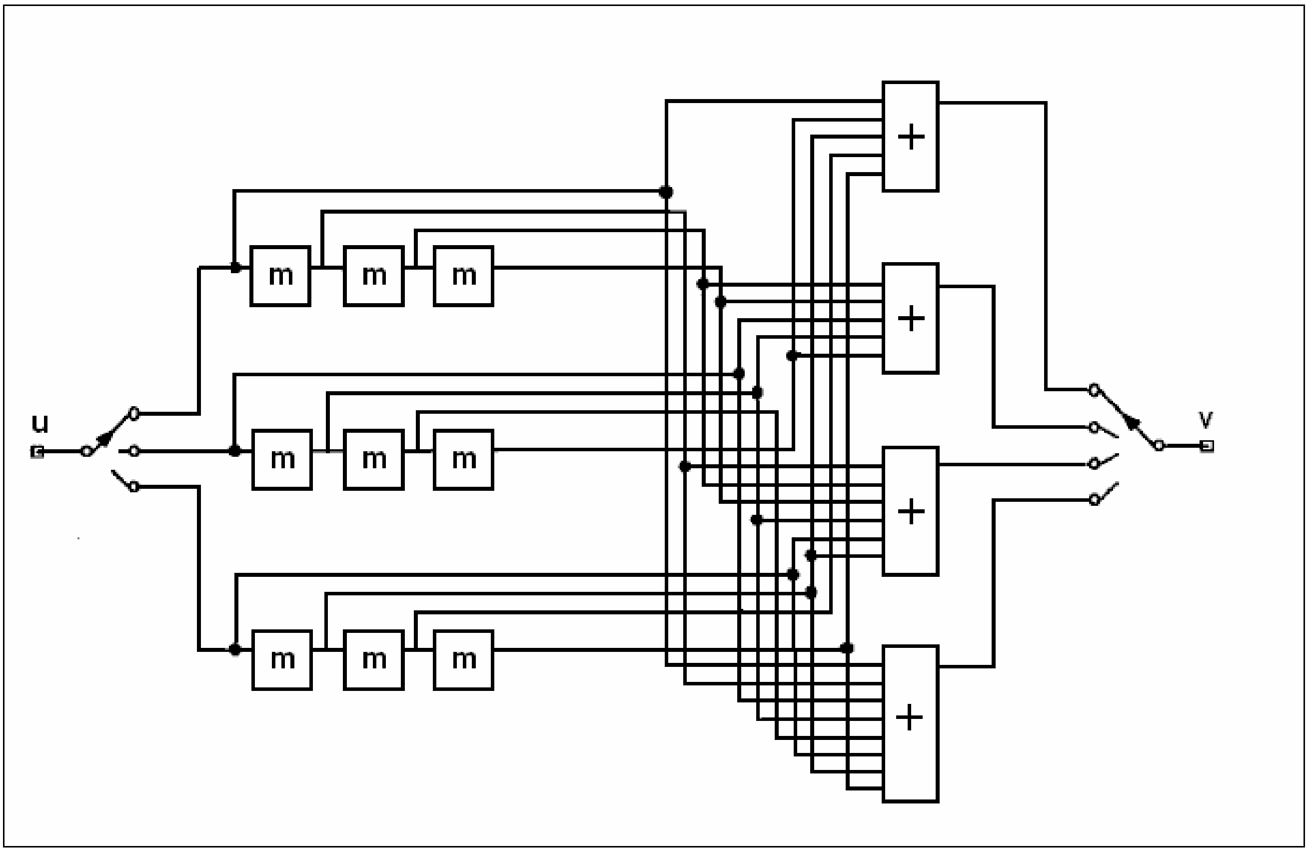


Figure K.1 — ECC 050; 4-3-3

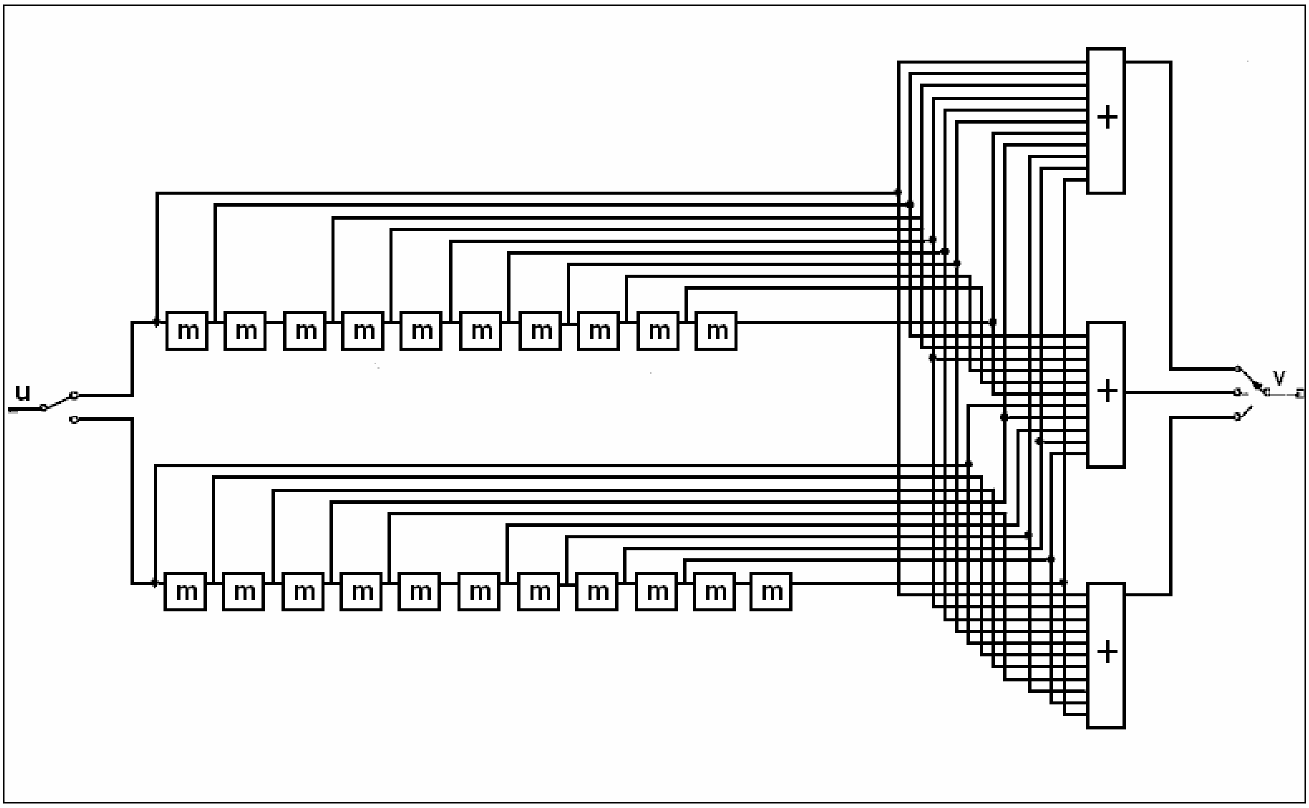


Figure K.2 — ECC 080; 3-2-11

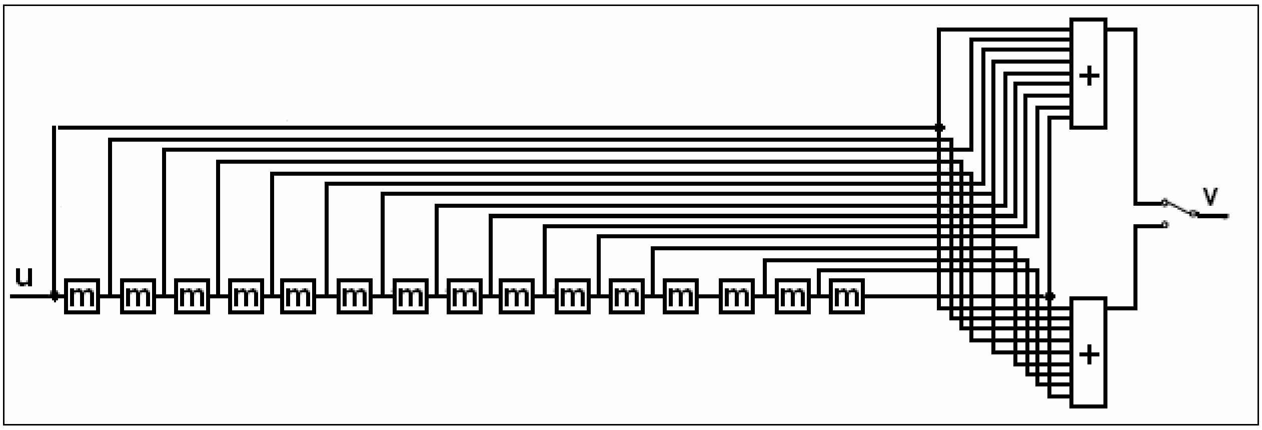


Figure K.3 — ECC 100; 2-1-15

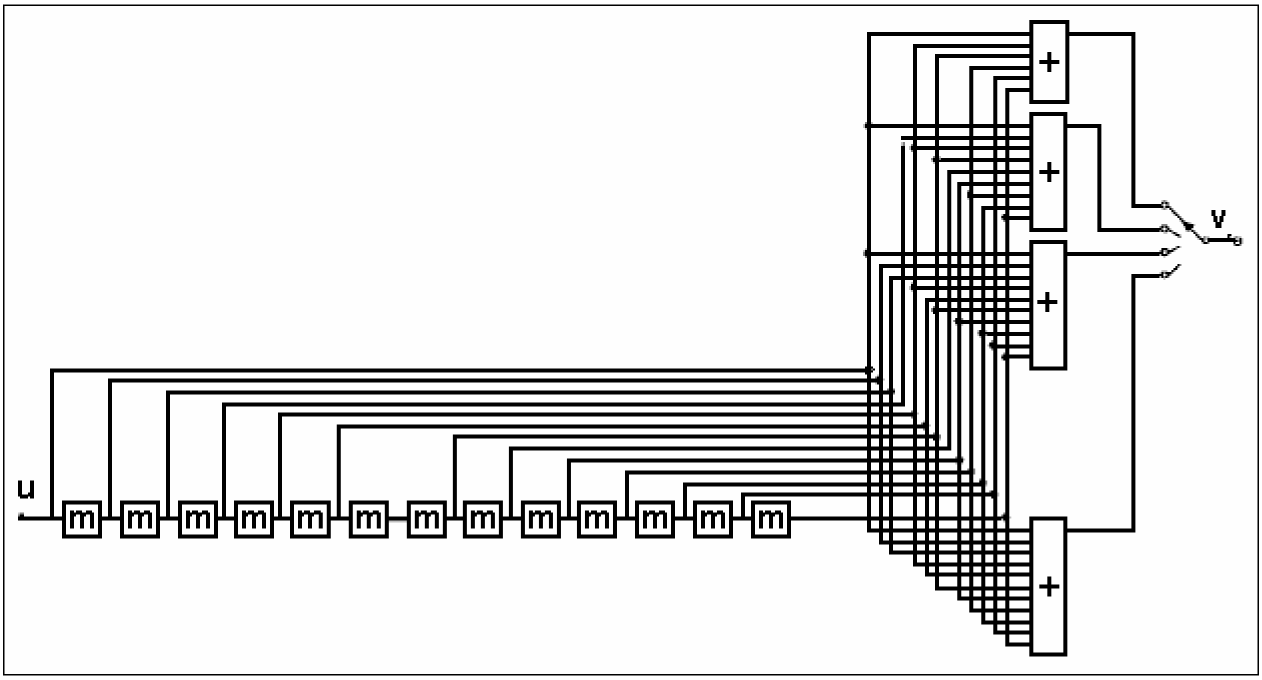


Figure K.4 — ECC 140; 4-1-13

Annex L
(normative)

ECC 000 - 140 Master Random Bit Stream (in hexadecimal)

```
(MSB)

05 ff c7 31 88 a8 83 9c 64 87 9f 64 b3 e0 4d 9c 80 29 3a 90

b3 8b 9e 90 45 bf f5 68 4b 08 cf 44 b8 d4 4c 5b a0 ab 72 52

1c e4 d2 74 a4 da 8a 08 fa a7 c7 dd 00 30 a9 e6 64 ab d5 8b

ed 9c 79 f8 08 d1 8b c6 22 64 0b 33 43 d0 80 d4 44 95 2e 6f

5e 13 8d 47 62 06 eb 80 82 c9 41 d5 73 8a 30 23 24 e3 7f b2

a8 0b ed 38 42 4c d7 b0 ce 98 bd e1 d5 e4 c3 1d 15 4a cf d1

1f 39 26 18 93 fc 19 b2 2d ab f2 6e a1 9f af d0 8a 2b a0 56

b0 41 6d 43 a4 63 f3 aa 7d af 35 57 c2 94 4a 65 0b 41 de b8

e2 30 12 27 9b 66 2b 34 5b b8 99 e8 28 71 d0 95 6b 07 4d 3c

7a b3 e5 29 b3 ba 8c cc 2d e0 c9 c0 22 ec 4c de f8 58 07 fc

19 f2 64 e2 c3 e2 d8 b9 fd 67 a0 bc f5 2e c9 49 75 62 82 27

10 f4 19 6f 49 f7 b3 84 14 ea eb e1 2a 31 ab 47 7d 08 29 ac

bb 72 fa fa 62 b8 c8 d3 86 89 95 fd df cc 9c ad f1 d4 6c 64

23 24 2a 56 1f 36 eb b7 d6 ff da 57 f4 50 79 08 0 (LSB)
```

Annex M
(normative)

Data Matrix print quality – symbology-specific aspects

Because of differences in symbology structures and reference decode algorithms, the effect of certain parameters on a symbol's reading performance may vary from one symbology to another. ISO/IEC 15415 provides for symbology specifications to define the grading of certain symbology-specific attributes. This annex therefore defines the method of grading Fixed Pattern Damage to be used in the application of ISO/IEC 15415 to Data Matrix.

M.1 Data Matrix Fixed Pattern Damage

M.1.1 Features to be assessed

The fixed pattern features to be assessed are contained in the one-module wide perimeter of the symbol and the quiet zone of a minimum of one module width (or more if specified by the application) surrounding the symbol. In larger symbols (square symbols 32 x 32 modules or larger, or rectangular symbols 8 x 32 or 12 x 36 or larger) with internal alignment patterns, the alignment pattern is also part of the fixed pattern. The left and lower side of the symbol should form a one-module wide solid "L" shape and the right and upper sides should consist of alternating dark and light single modules (known as the clock track). The alignment bars and internal clock track of the alignment pattern should similarly be a one-module wide solid bar or a series of alternating dark and light single modules respectively. The grading of Fixed Pattern Damage takes account not only of the total number of damaged modules but also of concentrations of damage.

M.1.2 Grading of the outside L of the fixed pattern

Damage to each side of the L shall be graded based on the modulation of the individual modules that compose it. These measurements are applied to the full length of the L sides and to the associated quiet zones.

Figure M.1 below indicates the four segments L1, L2, QZL1 and QZL2. Segment L1 is the vertical portion of the L and extends to the module in the quiet zone adjacent to the L corner. Segment L2 is the horizontal portion of the L and extends to the module in the quiet zone adjacent to the L corner. Segments QZL1 and QZL2 are the portions of the quiet zone adjacent to L1 and L2 respectively and extend one module beyond the end of L1 and L2 respectively, furthest from the corner and are shown shaded in Figure M.1. The corner module at the intersection of L1 and L2 is included in both segments, as is that at the intersection of QZL1 and QZL2.

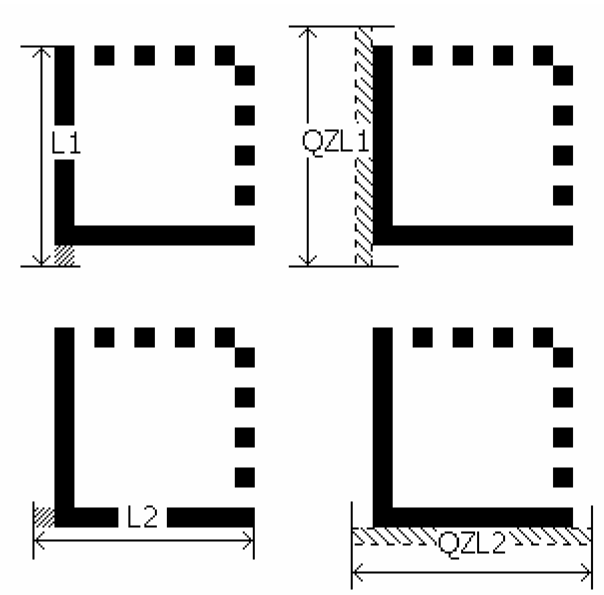


Figure M.1 — Outside L and corresponding quiet zone segments of fixed pattern

The procedure described below shall be applied to each segment in turn.

a) Find the modulation grade for each module based on the values in ISO/IEC 15415. Since the intended light or dark nature of the module is known, any module intended to be dark but the reflectance of which is above the global threshold, and any module intended to be light but the reflectance of which is below the global threshold shall be given modulation grade 0.

b) For each modulation grade level apply the parameter grade overlay technique described in ISO/IEC 15415:

c) For each side of the L (L1 and L2 in Figure M.1) and each quiet zone area (QZL1 and QZL2, adjacent to L1 and L2 respectively in Figure M.1), assume that all modules not achieving that grade or a higher grade are module errors, and derive a notional damage grade based on the grade thresholds shown in Table M.1. Take the lower of the modulation grade level and the notional damage grade.

d) Additionally, for symbols with more than one data region, repeat step a) above where L1 and L2 start with the module in the quiet zone and extend to include the module in the solid interior region of the next data region and QZL1 and QZL2 consist of the quiet zone adjacent to these L1 and L2 segments. In other words treat the lower left data region as if it were a symbol with a single data region. If this grade is lower than that obtained in step a) replace the grade obtained in step a) with this grade.

e) Additionally, for segments L1 and L2, verify that all gaps are separated by at least 4 correct modules and that no gaps are wider than three modules; if this test fails, the grade from step a) shall be reduced to 0 at that modulation grade level.

Table M.1 — Grade thresholds for notional damage

Percentage of modules damaged	Grade
0%	4
≤ 9%	3
≤ 13%	2
≤ 17%	1
> 17%	0

f) The grade for Fixed Pattern Damage for the segment shall be the highest resulting grade for all modulation grade levels.

M.1.3 Grading of the clock track and adjacent solid area segments

This section defines the measurement of damage to the internal alignment patterns (when present) and also external clock tracks and associated quiet zone areas. These tests are applied separately to each segment of the internal alignment patterns, the clock tracks, and associated quiet zone areas that bound the data region, or individual data regions of larger symbols. Each segment consists of a clock track portion and a solid area portion (which is part either of the quiet zone or of an internal alignment bar). A clock track portion commences with a dark module in the L side or internal alignment bar perpendicular to it and continues to the light module preceding either the quiet zone or the next internal alignment bar. A solid area portion commences with the module adjacent to the first module of the associated clock track portion and continues to the module one past the last module of the associated clock track portion. Figure M.2 illustrates the structures of these segments.

NOTE In a symbol without internal alignment patterns, the external clock track segments extend for the full width or height of the symbol.

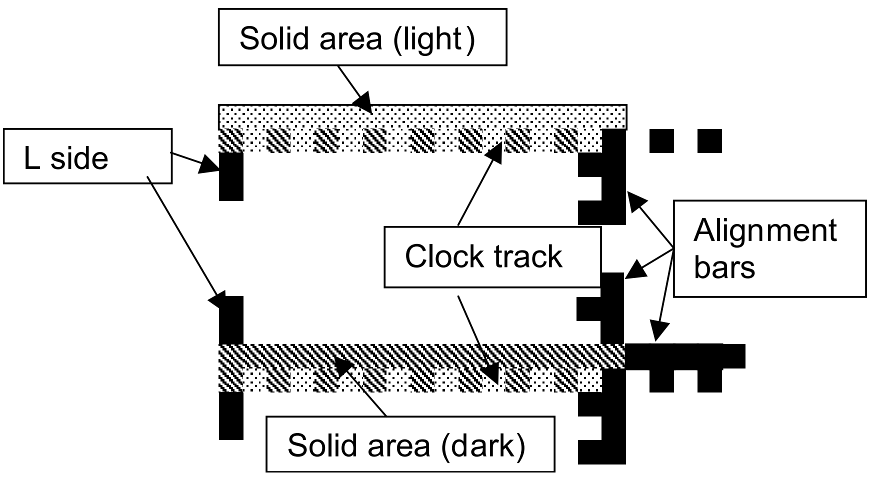


Figure M.2 — Structure of external clock track segment and internal alignment pattern segment

a) For each external clock track segment or internal alignment pattern segment of a symbol (for multi-segment symbols), damage is measured according to the following procedure.

b) Transition ratio test.

On every clock track segment in the binarised image, both external (adjacent to the quiet zone) and internal (adjacent to the solid internal alignment bar), count the number of transitions in the clock track side, *Tc*, and the solid line side, *Ts*, and compute and grade the transition ratio *TR* as follows:

$$Ts' = \mathrm{Max}\,(0,\ Ts - 1)$$

$$TR = Ts' / Tc$$

Table M.2 — Grading of Transition ratio

TR	Grade
$TR < 0{,}06$	4
$0{,}06 \leq TR < 0{,}08$	3
$0{,}08 \leq TR < 0{,}10$	2
$0{,}10 \leq TR < 0{,}12$	1
$TR \geq 0{,}12$	0

NOTE The end points between which transitions are counted are the intersections of grid lines plotted by the reference decode algorithm in the first and last modules of the clock track or solid area. See Figure M.3.

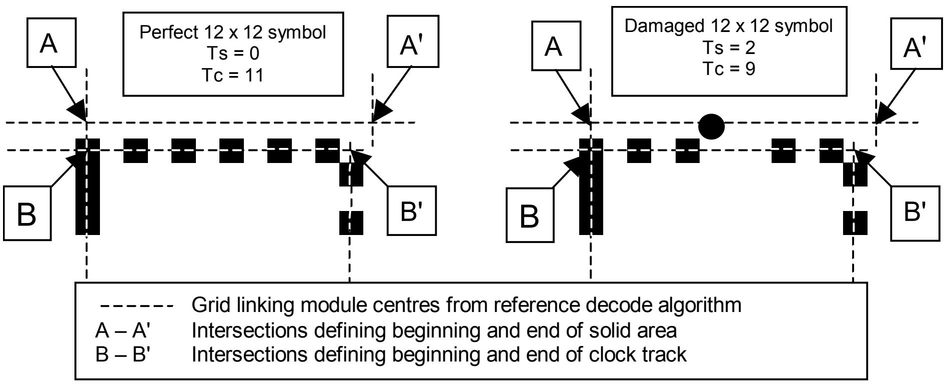


Figure M.3 — Transitions in perfect symbol (left) and damaged symbol (right)

c) Notional damage grade

Find the modulation grade for each module based on the values in ISO/IEC 15415. Since the intended light or dark nature of the module is known, any module intended to be dark but the reflectance of which is above the global threshold, and any module intended to be light but the reflectance of which is below the global threshold shall be given modulation grade 0.

d) For each modulation grade level:

Assume that all modules not achieving that grade or a higher grade are module errors, and derive a notional damage grade based on the following three assessments:

e) Clock track regularity test

For each segment of clock track, taking groups of five adjacent modules and progressing along the segment in steps of one module, verify that in any group of five adjacent modules no more than two are module errors; if this condition is met, the clock track regularity grade shall be 4, otherwise it shall be 0.

f) Clock track damage test

For each segment, count the number of incorrect modules in the clock track for the segment; the percentage *P* of incorrect modules over the length of the area shall result in the percentage damage grades shown in Table M.3.

g) Solid fixed pattern test

For each segment, count the number of incorrect modules in the solid area (internal alignment bar or external quiet zone area) adjacent to the clock track; the percentage *P* of incorrect modules over the length of the area shall result in the percentage damage grades shown in Table M.3.

Table M.3 — Grading of percentage damage to clock track segments and solid area segments

P	Grade
$P < 10\%$	4
$10\% \le P < 15\%$	3
$15\% \le P < 20\%$	2
$20\% \le P < 25\%$	1
$P \ge 25\%$	0

h) At each grade level take the lowest of the modulation grade level, the clock track regularity grade, the clock track percentage damage grade, and the solid fixed pattern percentage damage grade.

i) The notional damage grade for the segment shall be the highest resulting grade for all modulation grade levels.

j) The Fixed Pattern Damage grade for the segment shall be the lower of the transition ratio grade and the notional damage grade.

k) The overall Fixed Pattern Damage grade for the clock track and adjacent solid area segments is the lowest of the grades obtained for each of the individual segments.

The shaded areas in Figure M.4 below show an example of an internal alignment pattern segment, which includes the clock track portion and solid area portion to which the transition ratio, regularity and solid fixed pattern tests are applied.

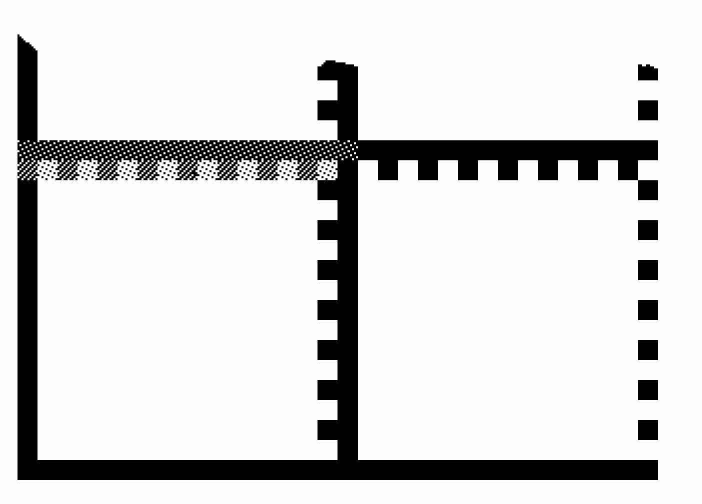


Figure M.4 — Internal alignment pattern segment

The shaded areas in Figure M.5 below show an example of a segment of the external clock track and associated quiet zone to which the transition ratio, regularity and solid fixed pattern tests are applied.

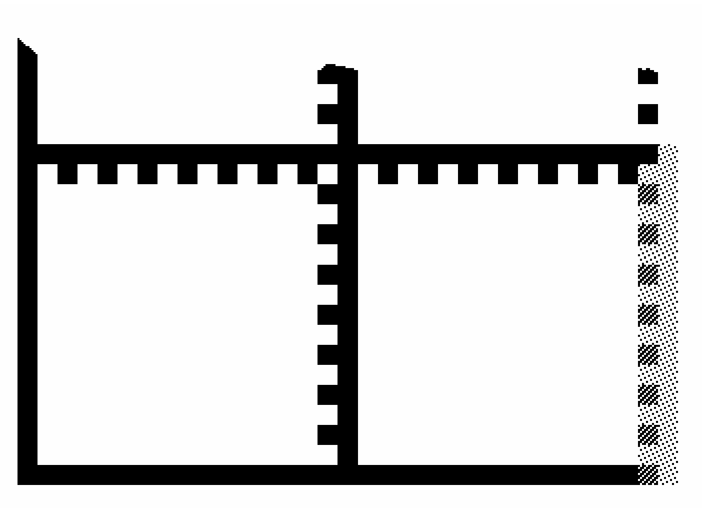


Figure M.5 — External clock track segment

EXAMPLE Figure M.6 shows an example based on grading the L1 segment of a 36 x 36 symbol, with *SC* = 89% and *GT* = 51%. The reflectance and modulation values, and modulation grade, are shown in Table M.4 for each of the 36 modules in the segment.

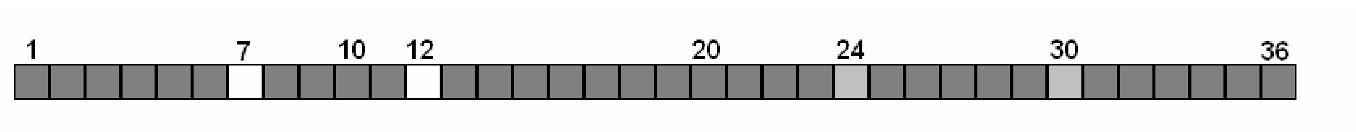


Figure M.6 — Example of L1 segment to show modulation effects

Table M.4 — Example of modulation grading of 36-module segment

Module	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Reflectance (%)	15	13	13	13	9	11	84	11	10
MOD	80	86	86	86	94	90	(74)	90	92
MOD Grade	**4**	**4**	**4**	**4**	**4**	**4**	**0**	**4**	**4**
Module	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Reflectance (%)	9	11	70	13	12	15	11	11	11
MOD	94	90	(42)	86	88	80	90	90	90
MOD Grade	**4**	**4**	**0**	**4**	**4**	**4**	**4**	**4**	**4**
Module	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
Reflectance (%)	27	11	14	10	12	50	12	11	14
MOD	54	90	83	92	88	2	88	90	83
MOD Grade	**4**	**4**	**4**	**4**	**4**	**0**	**4**	**4**	**4**
Module	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
Reflectance (%)	13	12	37	13	12	13	11	13	12
MOD	86	88	31	86	88	86	90	86	88
MOD Grade	**4**	**4**	**2**	**4**	**4**	**4**	**4**	**4**	**4**

Note that modules 7 and 12 are clearly light and module 24 and to a lesser extent module 30 suffer from low modulation.

Based upon these values, the segment grading would be as shown below:

Table M.5 — Example of grading of segment

MOD grade level	No. of modules	Cum. no. of modules	Remainder "damaged" modules	Damaged modules %	Notional damage grade	Lower of grades
4	32	32	4	11,1	2	2
3	0	32	4	11,1	2	2
2	1	33	3	8,3	3	2
1	0	33	3	8,3	3	1
0	3	36	0	0	4	0
Final Grade for segment - highest of last column						2

M.1.4 Calculation and grading of average grade

In addition to the assessment of the individual segments, a calculation of AG (average grade) is also made to take account of the cumulative effect of damage that is of relatively minor significance in individual segments but that affects several segments. This is based on averaging the grades for L1, L2, QZL1, QZL2 and the overall clock track and adjacent solid area segment grade

Once all segments have been graded, calculate the average grade AG:

AG = (Sum of the segment grades) / 5

Assign a grade to *AG* in accordance with Table M.6.

The Fixed Pattern Damage grade for the symbol shall be the lowest of the five segment grades and the grade for *AG*.

Table M.6 — Grading of *AG*

Mean of five segment grades	Grade
4	4
≥ 3,5	3
≥ 3,0	2
≥ 2,5	1
< 2,5	0

EXAMPLE 1

Assume that four of the five segments are graded 4, and one is graded 1. Then

(4 x 4) + (1 x 1) = 17

So *AG* = 17 / 5 = 3,4

From Table M.6, a mean of 3,4 will be graded 2. The lowest of the 6 grades is 1, and the symbol Fixed Pattern Damage grade, is therefore 1.

EXAMPLE 2

Assume that three of the five segments are graded 4, one is graded 3 and one is graded 1. Then

(3 x 4) + (1 x 3) + (1 x 1) = 16

So AG = 16 / 5 = 3,2

From Table M.6, a mean of 3,2 will be graded 2. The lowest of the 6 grades is1, and the symbol Fixed Pattern Damage grade is therefore1.

EXAMPLE 3

Assume that all of the five segments are graded 3. Then

5 x 3 = 15

So AG = 15 / 5 = 3,0

From Table M.6, a mean of 3,0 will be graded 2. The lowest of the 6 grades is 2, and the symbol Fixed Pattern Damage grade is therefore 2.

M.2 Scan grade

The scan grade shall be the lowest of the grades for the standard parameters evaluated according to ISO/IEC 15415 together with the grade for Fixed Pattern Damage evaluated in accordance with this Annex.

Annex N
(normative)

Symbology identifier

ISO/IEC 15424 provides a uniform methodology for reporting the symbology read, options set in the reader and any special features of the symbology encountered.

The symbology identifier for Data Matrix is:

]dm

where:

] is the symbology identifier flag (ASCII value 93)

d is the code character for the Data Matrix symbology

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

Table N.1 — Symbology Identifier option values for Data Matrix

Option value	Option
0	ECC 000 - 140
1	ECC 200
2	ECC 200, FNC1 in 1st or 5th position
3	ECC 200, FNC1 in 2nd or 6th position
4	ECC 200 supporting ECI protocol
5	ECC 200, FNC1 in 1st or 5th position plus supporting ECI protocol
6	ECC 200, FNC1 in 2nd or 6th position plus supporting ECI protocol
NOTE (Permissible values of m: 0, 1, 2, 3, 4, 5, 6)	

Annex O
(informative)

ECC 200 encode example

In this example the user data to be encoded is “123456” (length of 6).

Step 1: Data encodation

The ASCII representation is:

data character:	‘1’	‘2’	‘3’	‘4’	‘5’	‘6’
decimal:	49	50	51	52	53	54

ASCII encodation converts the above 6 characters to 3 bytes. This is done using the following formula for digit pairs.

Codeword = (numerical value of digit pairs) + 130

The details of this calculation are as follows.

“12” = 12 + 130 = 142

“34” = 34 + 130 = 164

“56” = 56 + 130 = 186

The data stream after data encodation is:

decimal: 142 164 186

Consulting Table 7, three data codewords fit exactly into a 10 x 10 symbol, and five error correction codewords need to be added. If the encoded data did not exactly fill a data region, then additional pads would have to be encoded.

Step 2: Error checking and correction

Error correction codewords are generated using the Reed-Solomon algorithm and appended to the encodation data stream. The resulting data stream is:

codeword:	1	2	3	4	5	6	7	8
decimal:	142	164	186	114	25	5	88	102
hex:	8E	A4	BA	72	19	05	58	66
	____data____/			_______check_______/				

Annex E describes the error correction process for ECC 200 and E.3 gives an example of a routine to perform the calculation of the error correction codewords.

Step 3: Module placement in matrix:

The final codewords from Step 2 are placed in the binary matrix as symbol characters according to the algorithm described in 5.8.1 (also see Figure F.1):

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

Figure O.1 — Module positioning in matrix

Step 4: Actual symbol

The final Data Matrix symbol is produced by adding the finder pattern modules and converting the binary ones to black and binary zeroes to white.

Figure O.2 — Final Data Matrix symbol encoding "123456"

Annex P
(informative)

Encoding data using the minimum symbol data characters for ECC 200

The same data may be represented by different Data Matrix symbols through the use of different code sets.

The following algorithm will usually produce the shortest codeword stream.

a) Start in ASCII encodation.

b) While in ASCII encodation:

   1) If the next data sequence is at least 2 consecutive digits, encode the next two digits as a double digit in ASCII mode.

   2) If the look-ahead test (starting at step j) indicates another mode, switch to that mode.

   3) If the Base 256 encodation mode has been indicated, encode the Latch to Base 256 encodation mode character followed by a currently undefined length byte; step G or step I will fill in the length field (this may require adding a second length byte).

   4) If the next data character is extended ASCII (greater than 127) encode it in ASCII mode first using the Upper Shift (value 235) character.

   5) Otherwise process the next data character in ASCII encodation.

c) While in C40 encodation:

   1) If the C40 encoding is at the point of starting a new double symbol character and if the look-ahead test (starting at step J) indicates another mode, switch to that mode.

   2) Otherwise process the next character in C40 encodation.

d) While in Text encodation:

   1) If the Text encoding is at the point of starting a new double symbol character and if the look-ahead test (starting at step J) indicates another mode, switch to that mode.

   2) Otherwise process the next character in Text encodation.

e) While in X12 encodation:

   1) If the X12 encoding is at the point of starting a new double symbol character and if the look-ahead test (starting at step J) indicates another mode, switch to that mode.

   2) Otherwise process the next character in X12 encodation.

f) While in EDIFACT (EDF) encodation:

   1) If the EDIFACT encoding is at the point of starting a new triple symbol character and if the look-ahead test (starting at step J) indicates another mode, switch to that mode.

   2) Otherwise process the next character in EDIFACT encodation.

g) While in Base 256 (B256) encodation:

   1) If the look-ahead test (starting at step J) indicates another mode, switch to that mode.

   2) Otherwise, process the next character in Base 256 encodation.

h) Repeat from step B until end of data.

i) At the end of data, if in Base 256 encodation, set the length to 0 (0 indicates that Base 256 encodation terminates the symbol).

The look-ahead test (Steps J through S):

The look-ahead test scans the data to be encoded to find the best mode.

j) Initialise the symbol character count for each mode:

   1) If the current mode is ASCII, initialise:

      ASCII count = 0,

      C40 count = 1,

      Text count = 1,

      X12 count = 1,

      EDF count = 1,

      B256 count = 1,25,

   otherwise initialise:

      ASCII count = 1,

      C40 count = 2,

      Text count = 2,

      X12 count = 2,

      EDF count = 2,

      B256 count = 2,25.

   2) If the current mode is C40 encodation, the C40 count = 0.

   3) If the current mode is Text encodation, the Text count = 0.

   4) If the current mode is X12 encodation, the X12 count = 0.

   5) If the current mode is EDIFACT encodation, the EDF count = 0.

   6) If the current mode is Base 256 encodation, the B256 count = 0.

k) If at the end of data:

   1) Round up all the counts to whole numbers.

   2) If the ASCII count is less than or equal to all the other counts, return from the test indicating ASCII encodation.

3) If the B256 count is less than all the other counts, return from the test indicating Base 256 encodation.

4) If the EDF count is less than all the other counts, return from the test indicating EDIFACT encodation.

5) If the Text count is less than all the other counts, return from the test indicating Text encodation.

6) If the X12 count is less than all the other counts, return from the test indicating X12 encodation.

7) Return from the test indicating C40 encodation.

l) Process the ASCII count:

1) If the data character is a digit, add 1/2 to the ASCII count.

2) If the data character is extended ASCII (greater than 127), round up and add 2 to the ASCII count.

3) Otherwise round up and add 1 to the ASCII count.

m) Process the C40 count:

1) If the data character is a native C40 character, add 2/3 to the C40 count.

2) If the data character is extended ASCII (greater than 127), add 8/3 to the C40 count.

3) Otherwise add 4/3 to the C40 count.

n) Process the Text count:

1) If the data character is a native Text character, add 2/3 to the Text count.

2) If the data character is extended ASCII (greater than 127), add 8/3 to the Text count.

3) Otherwise add 4/3 to the Text count.

o) Process the X12 count:

1) If the data character is a native X12 character, add 2/3 to the X12 count.

2) If the data character is extended ASCII (greater than 127), add 13/3 to the X12 count.

3) Otherwise add 10/3 to the X12 count.

p) Process the EDF count:

1) If the data character is a native EDF character, add 3/4 to the X12 count.

2) If the data character is extended ASCII (greater than 127), add 17/4 to the X12 count.

3) Otherwise add 13/4 to the X12 count.

q) Process the B256 count:

1) If the character is a Function character (FNC1, Structured Append, Reader Program, or Code Page), add 4 to the B256 count.

2) Otherwise add 1 to the B256 count.

r) If at least 4 data characters have been processed in this test loop:

   1) If the ASCII count plus 1 is less than or equal to all the other counts, return from the test indicating ASCII encodation.

   2) If the B256 count plus 1 is less than or equal to the ASCII count or less than the other counts, return from the test indicating Base 256 encodation.

   3) If the EDF count plus 1 is less than all the other counts, return from the test indicating EDIFACT encodation.

   4) If the Text count plus 1 is less than all the other counts, return from the test indicating Text encodation.

   5) If the X12 count plus 1 is less than all the other counts, return from the test indicating X12 encodation.

   6) If the C40 count plus 1 is less than the ASCII, B256, EDF, and Text counts:

      i) If the C40 count is less than the X12 count, return from the test indicating C40 count.

      ii) If the C40 count equals the X12 count:

         I) If one of the three X12 terminator/ separator characters first occurs in the yet to be processed data before a non-X12 character, return from the test indicating X12 encodation.

         II) Otherwise return with the C40 encodation.

s) Repeat from step k) until a return condition occurs.

Annex Q
(informative)

ECC 000 - 140 encode example using ECC 050

Q.1 Encode example

User data to be encoded: “AB12-X”. This will be encoded in base 41 (format ID3)

Step 1: Data encodation

	sequence 1	sequence 2
a) break user data into 4-character sequences:		
	A B 1 2	- X
b) convert to Base 41 code values:		
	1 2 28 29	39 24
c) apply conversion equations:		
	2045860	1023
d) convert to binary bit stream:		
	0111110011011110100100	01111111111
e) reverse each sequence to create the final Encoded Data Bit Stream:		
	0010010111101100111110	11111111110

Step 2: Data prefix construction

a) The format ID field for base 41 is given from Table 11 (Section 5.4.1):

00010

b) The CRC field is computed as shown in Q.2, then it receives an MSB/LSB reversal to result in:

1001 1010 1010 1110

c) The length field must be 6 in binary with MSB/LSB reversal:

011000000

d) The final Unprotected Bit Stream is shown in Figure Q.1.

Step 3: Error checking and correction:

The Unprotected Bit Stream is broken into 3-bit input blocks in preparation for input to the ECC 050 state machine. Three extra input blocks of all zeros have been added to the input block list; this gives a total of 24 input blocks (see Figure Q.1). The number of extra zero blocks added is equal to the longest shift register path through the state machine for the ECC being used; for ECC 050, 3 zero blocks are added. The basic flow of all ECC state machines is as follows:

a) Zero the state machine registers

b) Switch in a new input block (MSB goes to position 1)

c) Compute the output values from all XOR gates

d) Switch out an output block (MSB comes from position 1)

Table Q.1 shows the values of all state machine elements during the process of performing convolutional coding on the 24 input blocks.

The final Protected Bit Stream (length = 96 bits) is:

0000 1010 1011 1111 1010 1010 1010 0000 0100 0011 0110 1000 0101 0001 1000 0000 1110 1010 1001 1010 1001 1000 0100 1010

Unprotected Bit Stream (Step 2)																							
00010	1001101010101110	011000000	0010010111101100111110 11111111110																				
fmt3	CRC-16	len	encoded data																				
Unprotected Bit Stream broken into 3-bit blocks with three extra input blocks (Step 3)																							
000	101	001	101	010	101	110	011	000	000	001	001	011	110	110	011	111	011	111	111	110	000	000	000

Figure Q.1 — Unprotected Bit Stream from steps 2 and 3

Table Q.1 — Values of all registers during convolutional encoding

state machine cycle	Input 1 2 3	register 1A 1B 1C 2A 2B 2C 3A 3B 3C	output 1 2 3 4	state machine cycle	input 1 2 3	register 1A 1B 1C 2A 2B 2C 3A 3B 3C	output 1 2 3 4
1	0	000	0	13	0	000	0
	0	000	0		1	000	1
	0	000	0		1	110	0
			0				1
2	1	000	1	14	1	000	0
	0	000	0		1	100	0
	1	000	1		0	111	0
			0				1
3	0	100	1	15	1	100	1
	0	000	0		1	110	0
	1	100	1		0	011	0
			1				0
4	1	010	1	16	0	110	0
	0	000	1		1	111	0
	1	110	1		1	001	0
			1				0
5	0	101	1	17	1	011	1
	1	000	0		1	111	1
	0	111	1		1	100	1
			0				0
6	1	010	1	18	0	101	1
	0	100	0		1	111	0
	1	011	1		1	110	1
			0				0
7	1	101	1	19	1	010	1
	1	010	0		1	111	0
	0	101	1		1	111	0
			0				1
8	0	110	0	20	1	101	1
	1	101	0		1	111	0
	1	010	0		1	111	1
			0				0
9	0	011	0	21	1	110	1
	0	110	1		1	111	0
	0	101	0		0	111	0
			0				1
10	0	001	0	22	0	111	1
	0	011	0		0	111	0
	0	010	1		0	011	0
			1				0
11	0	000	0	23	0	011	0
	0	001	1		0	011	1
	1	001	1		0	001	0
			0				0
12	0	000	1	24	0	001	1
	0	000	0		0	001	0
	1	100	0		0	000	1
			0				0

Step 4: Header and trailer construction

The header contains the ECC bit field for 050 from Table 12 (6.6.1) with the bits reversed (MSB/LSB):

0111000000000111000 (length = 19 bits)

The trailer contains enough pad bits to make the Unrandomised Bit Stream fit exactly into a square matrix of the smallest size. There are 96 bits in the Protected Bit Stream and 19 bits in the header for a total of 115 bits.

A 13 x 13 data matrix has 11 x 11 information bits available (121 bits); this is the smallest matrix size able to contain 115 bits. There are 6 bits (121 - 115) that are set to zero. Therefore, the trailer is:

000000

The final Unrandomised Bit Stream is shown in Figure Q.2.

0111000000000111000	
header	
000010101011111110101010101000000100001101101000010100011000000011101010100110101001100001001010	000000
protected bit stream	trailer

Figure Q.2 — Final Unrandomised Bit Stream

Step 5: Pattern randomising

Partition the Unrandomised Bit Stream into 4-bit nibbles for easy XORing:

0111 0000 0000 0111 0000 0001 0101 0111 1111 0101 0101 0100 0000 1000 0110 1101 0000 1010 0011 0000 0001 1101 0101 0011 0101 0011 0000 1001 0100 0000 0

Get the required number (121) of random bits from the Master Random Bit Stream (Annex L):

(05, FF, C7, 31, 88, A8, 83, 9C, 64, 87, 9F, 64, B3, E0, 4D, first bit of 9C) =

0000 0101 1111 1111 1100 0111 0011 0001 1000 1000 1010 1000 1000 0011 1001 1100 0110 0100 1000 0111 1001 1111 0110 0100 1011 0011 1110 0000 0100 1101 1

Produce the Randomised Bit Stream by XORing the input with the random bits:

0111 0101 1111 1000 1100 0110 0110 0110 0111 1101 1111 1100 1000 1011 1111 0001 0110 1110 1011 0111 1000 0010 0011 0111 1110 0000 1110 1001 0000 1101 1

Step 6: Module placement in matrix

Using the Data Module Placement Grid for this matrix size, the data modules are placed into the binary matrix data area:

1 1 0 1 0 0 1 1 0 0 1

1 0 0 1 0 1 0 1 1 0 1

1 0 1 1 1 0 0 1 0 1 0

1 1 0 1 1 1 0 1 0 1 0

0 1 1 0 0 0 0 1 1 0 0

1 1 1 0 1 0 0 1 1 0 1

0 0 1 0 0 1 1 1 1 1 0

1 0 1 0 1 1 1 1 0 0 1

0 1 1 1 1 1 0 1 0 1 0

1 0 0 1 0 0 1 1 1 1 0

0 0 1 1 0 1 1 0 1 1 1

After adding the finder pattern modules, the final binary matrix is produced:

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

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

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

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

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

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

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

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

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

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

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

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

1 1 1 1 1 1 1 1 1 1 1 1 1

Q.2 CRC calculation for example

Construct the bit stream for input to the CRC algorithm. This consists of the CRC 2-byte header followed by the original user data. The CRC 2-byte header from Annex J, Table J.1 for format 3 is:

0000 0011 0000 0000.

The original user data is:

A B 1 2 - X

0100 0001, 0100 0010, 0011 0001, 0011 0010, 0010 1101, 0101 1000

The complete pre-CRC bit stream before byte reversal is:

0000 0011, 0000 0000, 0100 0001, 0100 0010, 0011 0001, 0011 0010, 0010 1101, 0101 1000

The complete pre-CRC bit stream after byte reversal is: (64 bits)

1100 0000, 0000 0000, 1000 0010, 0100 0010, 1000 1100, 0100 1100, 1011 0100, 0001 1010

This bit stream is input to the CRC state machine shown in Table Q.2. The CRC MSB is in the left-most shift register, so the final computed CRC value is 01110 1010101 1001 when read directly from the state machine. Parsing into 4-bit nibbles yields: 0111, 0101, 0101, 1001 which is the CRC field value used in Annex Q, Step 2b.

Table Q.2 — Value of all registers during CRC calculation

Machine cycle	XOR register bits 1-5	gate out3	XOR register bits 6-12	gate out2	register bits 13-16	input bit	XOR gate out1
start-up	00000	1	0000000	1	0000	1	1
1	10000	1	1000000	1	1000	1	1
2	11000	0	1100000	0	1100	0	0
3	01100	0	0110000	0	0110	0	0
4	00110	1	0011000	1	0011	0	1
5	10011	0	1001100	1	1001	0	1
6	11001	1	0100110	0	1100	0	0
7	01100	0	1010011	1	0110	0	0
8	00110	1	0101001	0	1011	0	1
9	10011	0	1010100	1	0101	0	1
10	11001	1	0101010	0	1010	0	0
11	01100	1	1010101	0	0101	0	1
12	10110	0	1101010	0	0010	0	0
13	01011	0	0110101	0	0001	0	1
14	10101	1	0011010	0	0000	0	0
15	01010	0	1001101	1	0000	0	0
16	00101	0	0100110	1	1000	1	1
17	10010	0	0010011	1	1100	0	0

Machine cycle	XOR register bits 1-5	gate out3	XOR register bits 6-12	gate out2	register bits 13-16	input bit	XOR gate out1
18	01001	1	0001001	1	1110	0	0
19	00100	1	1000100	1	1111	0	1
20	10010	1	1100010	1	1111	0	1
21	11001	0	1110001	0	1111	0	1
22	11100	0	0111000	0	0111	1	0
23	01110	1	0011100	1	0011	0	1
24	10111	0	1001110	1	1001	0	1
25	11011	0	0100111	0	1100	1	1
26	11101	1	0010011	1	0110	0	0
27	01110	1	1001001	0	1011	0	1
28	10111	0	1100100	1	0101	0	1
29	11011	1	0110010	0	1010	0	0
30	01101	1	1011001	1	0101	1	0
31	00110	0	1101100	0	1010	0	0
32	00011	1	0110110	0	0101	1	0
33	00001	1	1011011	1	0010	0	0
34	00000	1	1101101	0	1001	0	1
35	10000	0	1110110	0	0100	0	0
36	01000	1	0111011	0	0010	1	1
37	10100	0	1011101	1	0001	1	0
38	01010	0	0101110	0	1000	0	0
39	00101	1	0010111	1	0100	0	0
40	00010	0	1001011	1	1010	0	0
41	00001	1	0100101	1	1101	1	0
42	00000	0	1010010	0	1110	0	0
43	00000	1	0101001	0	0111	0	1
44	10000	0	1010100	0	0011	1	0
45	01000	0	0101010	0	0001	1	0
46	00100	0	0010101	1	0000	0	0
47	00010	0	0001010	0	1000	0	0
48	00001	0	0000101	0	0100	1	1
49	10000	0	0000010	0	0010	0	0
50	01000	0	0000001	1	0001	1	0
51	00100	1	0000000	1	1000	1	1
52	10010	0	1000000	0	1100	0	0
53	01001	0	0100000	1	0110	1	1

Machine cycle	XOR register bits 1-5	gate out3	XOR register bits 6-12	gate out2	register bits 13-16	input bit	XOR gate out1
54	10100	1	0010000	1	1011	0	1
55	11010	1	1001000	1	1101	0	1
56	11101	1	1100100	0	1110	0	0
57	01110	1	1110010	1	0111	0	1
58	10111	0	1111001	0	1011	0	1
59	11011	1	0111100	0	0101	1	0
60	01101	0	1011110	1	0010	1	1
61	10110	1	0101111	0	1001	0	1
62	11011	0	1010111	0	0100	1	1
63	11101	1	0101011	1	0010	0	0
64	01110		1010101		1001		

Annex R
(informative)

Useful process control techniques

This Annex describes tools and procedures useful for monitoring and controlling the process of creating scannable Data Matrix symbols. These techniques do not constitute a print quality check of the produced symbols (the method of Clause 8 and Annex M is the required method for assessing symbol print quality) but they individually and collectively yield good indications of whether the symbol print process is creating workable symbols.

R.1 Symbol contrast

Most linear bar code verifiers have either a reflectometer mode or a mode for plotting scan reflectance profiles and/or reporting Symbol Contrast (as defined in ISO/IEC 15415 and ISO/IEC 19762) from undecodable scans. Except with symbols requiring special illumination configurations, the symbol contrast readings that can be obtained using a 6 or 10 mil aperture at 660 nm wavelength (either the reported SC value, the peak-to-peak scan profile excursions, or the difference between peak reflectometer readings) are found to correlate well with an image-derived symbol contrast. In particular these readings can be used to check that symbol contrast stays well above the minimum allowed for the intended symbol quality grade.

R.2 Special reference symbol

For process control purposes, a 16 x 16 ECC 200 reference symbol can be printed which encodes the data “30Q324343430794<OQQ”. As shown in Figure R.1, this reference symbol has a region of parallel bars and spaces which can be linearly scanned and then evaluated for print growth using the edge-measurement methodologies of ISO/IEC 15416.

Figure R.1 — ECC 200 reference symbol encoding "30Q324343430794<OQQ"

Many linear bar code verifiers can be programmed to list element widths derived by the ISO/IEC 15416 methodology even for undecoded scans. The left-hand portion of any linear test scan across the upper half of the ECC 200 reference symbol will contain four bar-space pairs whose widths may be designated b_1 to b_4 and s_1 to s_4.

A normalised indication of horizontal print growth can be calculated as:

$(b_1 + b_2 + b_3 + \mathrm{b}_4) / (b_1 + s_1 + b_2 + s_2 + b_3 + s_3 + b_4 + s_4)$

This value in Data Matrix symbols should nominally be 50% and stay within 35% to 65% limits.

Note that this measurement will not be sensitive to printing variations parallel to the long dimension of the elements in the reference symbol. If a more complete assessment of the print process is desired, the Data Matrix reference symbol should be printed in both orientations and tested.

R.3 Assessing Axial Nonuniformity

For any symbol, measure the length of both legs of the "L" shaped finder pattern. Divide each length by the number of modules in that dimension, e.g. a 12 x 36 symbol would have 12 and 36 as divisors. These two normalised dimensions are X_{AVG} and Y_{AVG} which can be used in the formula below to grade Axial Nonuniformity.

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

If the value of AN is greater than 0,12 the symbol would fail according to ISO/IEC 15415. Values up to 0,06 correspond to a grade 4 for this parameter.

R.4 Visual inspection for symbol distortion and defects

Ongoing visual inspection of the perimeter patterns in sample symbols can monitor two important aspects of the print process. First, 2D matrix symbols are susceptible to errors caused by local distortions of the matrix grid. Any such distortions will show up visually in a Data Matrix symbol as either crooked edges on the "L" shaped finder pattern or uneven spacings within the alternating patterns found along the other two margins of the symbol. Larger ECC 200 symbols also include alignment patterns whose straightness and evenness can be visually checked. Symbols likely to fail the reference decode can be quickly identified in this way. Second, the two arms of the finder pattern and the adjacent quiet zones should always be solidly in opposite reflectance states. Failures in the print mechanism which may produce defects in the form of light or dark streaks through the symbol should be visibly evident where they infringe the finder or quiet zone. Such systematic failures in the print process should be corrected.

Annex S
(informative)

Autodiscrimination capability

Data Matrix may be read by suitably programmed bar code decoders which have been designed to autodiscriminate it from other symbologies. The decoder's valid set of symbologies should be limited to those needed by a given application to maximise reading security.

Annex T
(informative)

System considerations

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

- While compliance with the specifications is one key to assuring overall system success, other considerations come into play which may influence performance as well. The following guidelines suggest some factors to keep in mind when specifying or implementing bar code systems:
- Select a print density which will yield tolerance values that can be achieved by the marking or printing technology being used.
- Choose a reader with a resolution suitable for the symbol density and quality produced by the printing technology.
- Ensure that the printed symbol's optical properties are compatible with the wavelength of the scanner's light source or sensor.
- Verify symbol compliance in the final label or package configuration. Overlays, showthrough, and curved or irregular surfaces can all affect symbol readability.

Marking technologies that are not consistently capable of producing a solid line of continuous modules, for example, dot peen and ink jet, require particular care to ensure that gaps between nominally touching modules do not interfere with the decoding of the symbol using the application specified aperture size. In addition, the relative positioning of modules and the horizontal and vertical axes needs to comply with the requirements for axial non-uniformity specified in ISO/IEC 15415. Application specifications should also consult ISO/IEC 15415 for guidance regarding the specification of aperture size, lighting, and other parameters.

Scanning systems must take into consideration the variations in diffuse reflection between dark and light features. At some scanning angles, the specular component of the reflected light can greatly exceed the desired diffuse component, making successful reading more difficult. In cases where the surface of the part or material can be altered, matte, non-glossy finishes may help minimise specular effects. Where this option is not available, particular care must be taken to ensure the illumination for the mark being read optimises the desired contrast components.

Bibliography

[1] LIN and COSTELLO, “Error Control Coding: Fundamentals and Applications,” Prentice Hall, 1983.

[2] C. BRITTON RORBAUGH, “Error Coding Cookbook,” McGraw Hill, 1996.

[3] AIM Inc. “Data Matrix Developer's Diskette” (AIM Inc., 125 Warrendale-Bayne Road, Suite 100, Warrendale, PA 15086, USA).

ICS 01.080.50; 35.040

Price based on 132 pages