

---

---

## Information technology — International symbology specification — Data matrix

*Technologies de l'information — Spécification internationale des  
symboles — Matrice de données*

---

---

Reference number  
ISO/IEC 16022:2000(E)



**PDF disclaimer**

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2000

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office  
Case postale 56 • CH-1211 Geneva 20  
Tel. + 41 22 749 01 11  
Fax + 41 22 734 10 79  
E-mail [copyright@iso.ch](mailto:copyright@iso.ch)  
Web [www.iso.ch](http://www.iso.ch)

Printed in Switzerland

## Contents

|                                                                              |          |
|------------------------------------------------------------------------------|----------|
| <b>Introduction .....</b>                                                    | <b>1</b> |
| <b>1 Scope .....</b>                                                         | <b>1</b> |
| <b>2 Normative References .....</b>                                          | <b>1</b> |
| <b>3 Definitions and Mathematical Symbols .....</b>                          | <b>1</b> |
| 3.1 Definitions .....                                                        | 1        |
| 3.1.1 Alignment Pattern .....                                                | 1        |
| 3.1.2 Codeword .....                                                         | 1        |
| 3.1.3 Module .....                                                           | 2        |
| 3.1.4 Convolutional Coding .....                                             | 2        |
| 3.1.5 Cyclic Redundancy Check (CRC) .....                                    | 2        |
| 3.1.6 Extended Channel Interpretation (ECI) .....                            | 2        |
| 3.1.7 Pattern Randomizing .....                                              | 2        |
| 3.2 Mathematical Symbols and Operation .....                                 | 2        |
| <b>4 Symbol Description .....</b>                                            | <b>2</b> |
| 4.1 Basic Characteristics .....                                              | 2        |
| 4.2 Summary of Additional Features .....                                     | 3        |
| 4.3 Symbol Structure .....                                                   | 3        |
| 4.3.1 Finder Pattern .....                                                   | 4        |
| 4.3.2 Symbol Sizes .....                                                     | 4        |
| <b>5 ECC 000 - 140 Requirements .....</b>                                    | <b>4</b> |
| 5.1 Encode Procedure Overview .....                                          | 4        |
| 5.2 Data Encodation .....                                                    | 5        |
| 5.2.1 Base 11 - Numeric Encodation .....                                     | 6        |
| 5.2.2 Base 27 - Upper-case Alphabetic Encodation .....                       | 6        |
| 5.2.3 Base 37 - Upper-case Alphanumeric Encodation .....                     | 6        |
| 5.2.4 Base 41 - Upper-case Alphanumeric plus Punctuation Encodation .....    | 6        |
| 5.2.5 ASCII Encodation .....                                                 | 7        |
| 5.2.6 8-bit Byte Encodation .....                                            | 7        |
| 5.3 User Selection of Error Correction Level .....                           | 7        |
| 5.3.1 Selection of Error Correction Level .....                              | 7        |
| 5.3.2 Additional Error Correction Levels Based on Convolutional Codes .....  | 7        |
| 5.4 Constructing the Unprotected Bit Stream .....                            | 7        |
| 5.4.1 Format ID Bit Field .....                                              | 7        |
| 5.4.2 CRC Bit Field .....                                                    | 7        |
| 5.4.3 Data Length Bit Field .....                                            | 7        |
| 5.4.4 Data Prefix Construction .....                                         | 7        |
| 5.4.5 Completing the Unprotected Bit Stream .....                            | 7        |
| 5.5 Constructing the Unrandomized Bit Stream .....                           | 7        |
| 5.5.1 Header Construction .....                                              | 8        |
| 5.5.2 Applying Convolutional Coding to Create the Protected Bit Stream ..... | 8        |
| 5.5.3 Trailer Construction .....                                             | 8        |
| 5.5.4 Completing the Unrandomized Bit Stream .....                           | 8        |
| 5.6 Pattern Randomizing .....                                                | 8        |
| 5.7 Module Placement in Matrix .....                                         | 8        |
| <b>6 ECC 200 Requirements .....</b>                                          | <b>8</b> |
| 6.1 Encode Procedure Overview .....                                          | 8        |
| 6.2 Data Encodation .....                                                    | 9        |
| 6.2.1 Overview .....                                                         | 9        |
| 6.2.2 Default Character Interpretation .....                                 | 9        |
| 6.2.3 ASCII Encodation .....                                                 | 9        |
| 6.2.4 Symbol Control Characters .....                                        | 10       |
| 6.2.5 C40 Encodation .....                                                   | 11       |
| 6.2.6 Text Encodation .....                                                  | 12       |
| 6.2.7 ANSI X12 Encodation .....                                              | 12       |
| 6.2.8 EDIFACT Encodation .....                                               | 13       |

|                                                                               |           |
|-------------------------------------------------------------------------------|-----------|
| 6.2.9 Base 256 Encodation .....                                               | 14        |
| 6.3 User Considerations .....                                                 | 14        |
| 6.3.1 User Selection of Extended Channel Interpretation .....                 | 14        |
| 6.3.2 User Selection of Symbol Size and Shape .....                           | 14        |
| 6.4 Extended Channel Interpretation .....                                     | 14        |
| 6.4.1 Encoding ECIs .....                                                     | 15        |
| 6.4.2 ECIs and Structured Append .....                                        | 15        |
| 6.4.3 Post-Decode Protocol .....                                              | 17        |
| 6.5 ECC 200 Symbol Attributes .....                                           | 17        |
| 6.5.1 Symbol Sizes and Capacity .....                                         | 17        |
| 6.5.2 Insertion of Alignment Patterns into Larger Symbols .....               | 17        |
| 6.6 Structured Append .....                                                   | 17        |
| 6.6.1 Basic Principles .....                                                  | 17        |
| 6.6.2 Symbol Sequence Indicator .....                                         | 17        |
| 6.6.3 File Identification .....                                               | 17        |
| 6.6.4 FNC1 and Structured Append .....                                        | 17        |
| 6.6.5 Buffered and Unbuffered Operation .....                                 | 18        |
| 6.7 Error Detection and Correction .....                                      | 18        |
| 6.7.1 Generating the Error Correction Codewords .....                         | 18        |
| 6.7.2 Error Correction Capacity .....                                         | 18        |
| 6.8 Symbol Construction .....                                                 | 19        |
| 6.8.1 Symbol Character Placement .....                                        | 19        |
| 6.8.2 Alignment Pattern Module Placement .....                                | 19        |
| 6.8.3 Finder Pattern Module Placement .....                                   | 19        |
| <b>7 Symbol Dimensions .....</b>                                              | <b>20</b> |
| 7.1 Dimensions .....                                                          | 20        |
| 7.2 Quiet Zone .....                                                          | 20        |
| <b>8 Symbol Quality .....</b>                                                 | <b>20</b> |
| 8.1 Obtaining the Test Image .....                                            | 20        |
| 8.2 Symbol Quality Parameters .....                                           | 20        |
| 8.2.1 Decode .....                                                            | 20        |
| 8.2.2 Symbol Contrast .....                                                   | 20        |
| 8.2.3 "Print" Growth .....                                                    | 20        |
| 8.2.4 Axial Nonuniformity .....                                               | 20        |
| 8.2.5 Unused Error Correction .....                                           | 20        |
| 8.3 Overall Symbol Grade .....                                                | 21        |
| 8.4 Process Control Measurements .....                                        | 21        |
| <b>9 Reference Decode Algorithm for Data Matrix .....</b>                     | <b>21</b> |
| <b>10 User Guidelines .....</b>                                               | <b>26</b> |
| 10.1 Human Readable Interpretation .....                                      | 26        |
| 10.2 Autodiscrimination Capability .....                                      | 26        |
| 10.3 System Consideration .....                                               | 27        |
| <b>11 Transmitted Data .....</b>                                              | <b>27</b> |
| 11.1 Protocol for FNC1 (ECC 200 Only) .....                                   | 27        |
| 11.2 Protocol for FNC1 in the Second Position .....                           | 27        |
| 11.3 Protocol for Macro Characters in the First Position (ECC 200 only) ..... | 27        |
| 11.4 Protocol for ECIs (ECC 200 Only) .....                                   | 27        |
| 11.5 Symbology Identifier .....                                               | 28        |
| 11.6 Transmitted Data Example .....                                           | 28        |
| <b>Annexe A (Normative) .....</b>                                             | <b>29</b> |
| ECC 000 - 140 Symbol Attributes .....                                         | 29        |
| A.1 ECC 000 .....                                                             | 29        |
| A.2 ECC 050 .....                                                             | 29        |
| A.3 ECC 080 .....                                                             | 30        |
| A.4 ECC 100 .....                                                             | 30        |
| A.5 ECC 140 .....                                                             | 31        |
| <b>Annexe B (Normative) .....</b>                                             | <b>32</b> |

|                                                                         |           |
|-------------------------------------------------------------------------|-----------|
| ECC 000 - 140 Data Module Placement Grids .....                         | 32        |
| <b>Annexe C (Normative) .....</b>                                       | <b>41</b> |
| ECC 000 - 140 Character Encodation Schemes .....                        | 41        |
| C.1 Base 11 Encodation Scheme .....                                     | 44        |
| C.1.1 First Stage Procedure .....                                       | 44        |
| C.1.2 Second Stage Procedure .....                                      | 44        |
| C.1.3 Example .....                                                     | 44        |
| C.2 Base 27 Encodation Scheme .....                                     | 45        |
| C.2.1 First Stage Procedure .....                                       | 45        |
| C.2.2 Second Stage Procedure .....                                      | 45        |
| C.2.3 Example .....                                                     | 45        |
| C.3 Base 37 Encodation Scheme .....                                     | 46        |
| C.3.1 First Stage Procedure .....                                       | 46        |
| C.3.2 Second Stage Procedure .....                                      | 46        |
| C.3.3 Example .....                                                     | 46        |
| C.4 Base 41 Encodation Scheme .....                                     | 47        |
| C.4.1 First Stage Procedure .....                                       | 47        |
| C.4.2 Second Stage Procedure .....                                      | 47        |
| C.4.3 Example .....                                                     | 47        |
| <b>Annexe D (Normative) .....</b>                                       | <b>48</b> |
| ECC 000 - 140 CRC Algorithm .....                                       | 48        |
| D.1 CRC State Machine .....                                             | 48        |
| D.2 CRC Polynomial .....                                                | 48        |
| D.3 CRC 2-Byte Header .....                                             | 48        |
| <b>Annexe E (Normative) .....</b>                                       | <b>49</b> |
| ECC 000 - 140 Error Checking and Correcting Algorithms .....            | 49        |
| E.1 ECC 000 .....                                                       | 49        |
| E.2 ECC 050 .....                                                       | 49        |
| E.3 ECC 080 .....                                                       | 49        |
| E.4 ECC 100 .....                                                       | 49        |
| E.5 ECC 140 .....                                                       | 49        |
| E.6 Processing the Convolutional Code .....                             | 49        |
| E.7 Convolutional Codes Reference Decode Algorithm .....                | 49        |
| <b>Annexe F (Normative) .....</b>                                       | <b>55</b> |
| ECC 000 - 140 Master Random Bit Stream .....                            | 55        |
| <b>Annexe G (Normative) .....</b>                                       | <b>56</b> |
| ECC 200 Interleaving Process .....                                      | 56        |
| G.1 Schematic Illustration .....                                        | 56        |
| G.2 Starting Sequence for Interleaving in Different Sized Symbols ..... | 57        |
| <b>Annexe H (Normative) .....</b>                                       | <b>59</b> |
| ECC 200 Pattern Randomizing .....                                       | 59        |
| H.1 253-State Algorithm .....                                           | 59        |
| H.1.1 253-State Randomizing Algorithm .....                             | 59        |
| H.1.2 253-State Un-Randomizing Algorithm .....                          | 59        |
| H.2 255-State Algorithm .....                                           | 59        |
| H.2.1 255-State Randomizing Algorithm .....                             | 59        |
| H.2.2 255-State Un-Randomizing Algorithm .....                          | 59        |
| <b>Annexe J (Normative) .....</b>                                       | <b>60</b> |
| ECC 200 Encodation Character Sets .....                                 | 60        |
| J.1 C40 Encodation Character Set .....                                  | 60        |
| J.2 Text Encodation Character Set .....                                 | 61        |
| J.3 EDIFACT Encodation Character Set .....                              | 62        |
| <b>Annexe K (Normative) .....</b>                                       | <b>63</b> |
| ECC 200 Alignment Patterns .....                                        | 63        |
| <b>Annexe L (Normative) .....</b>                                       | <b>65</b> |
| ECC 200 Reed-Solomon Error Detection and Correction .....               | 65        |
| L.1 Error Correction Codeword Generator Polynomials .....               | 65        |

|                                                                          |           |
|--------------------------------------------------------------------------|-----------|
| L.2 Error Correction Calculation .....                                   | 66        |
| <b>Annexe M (Normative) .....</b>                                        | <b>68</b> |
| ECC 200 Symbol Character Placement .....                                 | 68        |
| M.1 Symbol Character Placement Program .....                             | 68        |
| M.2 Symbol Character Placement Rules .....                               | 71        |
| M.2.1 Non-standard Symbol Character Shapes .....                         | 71        |
| M.2.2 Symbol Character Arrangement .....                                 | 71        |
| M.3 Symbol Character Placement Examples for ECC 200 .....                | 75        |
| <b>Annexe N (Normative) .....</b>                                        | <b>82</b> |
| 2D Matrix Bar Code Print Quality - Guideline .....                       | 82        |
| N.1 Obtaining the Test Image .....                                       | 82        |
| N.2 Assessing Symbol Parameters .....                                    | 82        |
| N.2.1 Decode .....                                                       | 82        |
| N.2.2 Symbol Contrast .....                                              | 82        |
| N.2.3 "Print" Growth .....                                               | 83        |
| N.2.4 Axial Nonuniformity .....                                          | 83        |
| N.2.5 Unused Error Correction .....                                      | 83        |
| N.3 Overall Symbol Grade .....                                           | 84        |
| <b>Annexe P (Normative) .....</b>                                        | <b>84</b> |
| Symbology Identifier .....                                               | 84        |
| <b>Annexe Q (Informative) .....</b>                                      | <b>84</b> |
| ECC 000-140 Encode Example Using ECC 050 .....                           | 84        |
| Q.1 CRC Calculation for Example .....                                    | 89        |
| <b>Annexe R (Informative) .....</b>                                      | <b>90</b> |
| ECC 200 Encode Example .....                                             | 90        |
| <b>Annexe S (Informative) .....</b>                                      | <b>91</b> |
| Encoding Data Using the Minimum Symbol Data Characters for ECC 200 ..... | 91        |
| <b>Annexe T (Informative) .....</b>                                      | <b>92</b> |
| Useful Process Control Techniques .....                                  | 92        |
| T.1 Symbol Contrast .....                                                | 93        |
| T.2 Special Reference Symbol .....                                       | 93        |
| T.3 Assessing Axial Nonuniformity .....                                  | 93        |
| T.4 Visual Inspection for Symbol Distortion and Defects .....            | 93        |
| <b>Annexe U (Informative) .....</b>                                      | <b>94</b> |
| Autodiscrimination Capabilities .....                                    | 94        |
| <b>Annexe V (Informative) .....</b>                                      | <b>94</b> |
| System Considerations .....                                              | 94        |

## Foreword

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

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

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

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

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

International Standard ISO/IEC 16022 was prepared by AIM International (as ANSI/AIM BC11) and was adopted, under a special “fast-track procedure”, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

Annexes A to P form a normative part of this International Standard. Annexes Q to V are for information only.



# Information technology — International symbology specification — Data matrix

## Introduction

Data Matrix is a two-dimensional matrix symbology which is made up of square modules arranged within a perimeter finder pattern. Though primarily shown and described in this document 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 Symbology Specifications is designed to achieve this.

## 1 Scope

This specification 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.

## 2 Normative References

This specification incorporates provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed below. The latest edition of the publication referred to applies.

|                |                                                                                                                                                                          |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EN796          | Bar Coding : Symbology Identifiers                                                                                                                                       |
| EN1556         | Bar Coding : Terminology                                                                                                                                                 |
| ANSI<br>X3.182 | Bar Code Print Quality - Guideline<br>(Same as EN1635 - Bar Coding :<br>Test Specifications for Bar Code<br>Symbols)                                                     |
| ANSI<br>X3.4   | Coded Character Sets - 7-bit<br>American National Standard Code<br>for Information Interchange (7-bit<br>ASCII)<br>(equivalent to the US national<br>version of ISO 646) |

ISO/IEC  
8859-1      Information Processing - 8-bit  
Single-byte Coded Graphic  
Character Sets - Part 1 (Latin  
Alphabet Number 1)

ECI Assignments Document- AIM International

## 3 Definitions and Mathematical Symbols

### 3.1 Definitions

For the purposes of this specification, the following definitions in EN 1556 shall apply:

algorithm, application standard, ASCII, autodiscrimination, binary, bit, CCD, code page, code set, data character, data codeword, data region, data separator character, decode algorithm, decoder, error correction, finder pattern, human readable character, latch character, leading zeros, matrix symbology, modulo, numeric, omnidirectional, orientation pattern, overhead, pad character, pixel, quiet zone, reference decode algorithm, Reed-Solomon error correction, scanner, shift characters, structured append, symbol character, symbology, symbology identifier, X-dimension

The following definitions also apply to this specification. Although some of the terms below are defined in EN1556, the definitions which follow below are more appropriate for this specification.

#### 3.1.1 Alignment Pattern

A unique pattern in larger ECC 200 Data Matrix symbols, made up of a solid line of contiguous dark cells abutting a line of alternating dark and light cells. The alignment patterns run horizontally and vertically within the symbols.

#### 3.1.2 Codeword

A symbol character value. An intermediate level of coding between source data and the graphical encodation in the symbol.

### 3.1.3 Module

A single cell in a matrix symbology used to encode one bit of data. In Data Matrix the module is a square shape.

### 3.1.4 Convolutional Coding

An Error Checking and Correcting (ECC) algorithm that processes a set of input bits into a set of output bits that can recover from damage. The encoding process consists of 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. These encoders can be constructed in hardware using input and output switches, shift registers, and exclusive-or (XOR) gates.

### 3.1.5 Cyclic Redundancy Check (CRC)

An error detection algorithm which performs binary modulo 2 long division of a binary input stream by a fixed binary bit stream and produces as output the remainder of this division. The fixed binary bit stream is usually represented by a polynomial. The remainder of the division is the CRC value. The division can be implemented in hardware by shift registers and XOR gates. The purpose of a CRC is to provide error detection allowing validation of the user data after correction by the ECC process.

### 3.1.6 Extended Channel Interpretation (ECI)

A protocol used by some symbologies that allows the output data stream to have interpretations other than that of the default character set.

### 3.1.7 Pattern Randomizing

A procedure which converts an original bit pattern to another bit pattern by inverting selected bits. The resulting bit stream is less likely to have repeating patterns.

## 3.2 Mathematical Symbols and Operation

For the purposes of this specification, the mathematical symbols which follow shall apply globally 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 a encodation scheme

p number of misdecode protection codewords

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

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

- div is the integer division operator
- mod is the integer remainder after division
- XOR is the exclusive-or logic function whose output is one only when its two inputs are not equivalent.
- 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 000 - 140 with several available levels of convolutional error correction and ECC 200 which uses Reed-Solomon error correction. For new applications ECC 200 is recommended. 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 ANSI X3.4, i.e. all 128 ASCII characters

(equivalent to the U.S. national version of ISO 646)

2. values 128 - 255 in accordance with ISO 8859-1; Latin Alphabet No. 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.
- c. Symbol size in modules (not including quiet zone):
 

|                      |                |
|----------------------|----------------|
| <u>ECC 000 - 140</u> | <u>ECC 200</u> |
| 9 by 9               | 10 by 10       |
| to 49 by 49          | to 144 by 144  |
| Odd Only             | Even Only      |
- d. Data characters per symbol (for maximum symbol size in ECC200):
 

|                       |                       |
|-----------------------|-----------------------|
| 1. Alphanumeric data: | up to 2335 characters |
| 2. 8-bit byte data:   | 1556 characters       |
| 3. Numeric data:      | 3116 digits           |
- e. Selectable error correction:
 

|                |                                                                                              |
|----------------|----------------------------------------------------------------------------------------------|
| ECC 000 - 140: | Four levels of convolutional error correction, plus the option to apply only error detection |
| ECC 200:       | Reed-Solomon error correction                                                                |
- f. Code type: Matrix
- g. Orientation independence: Yes

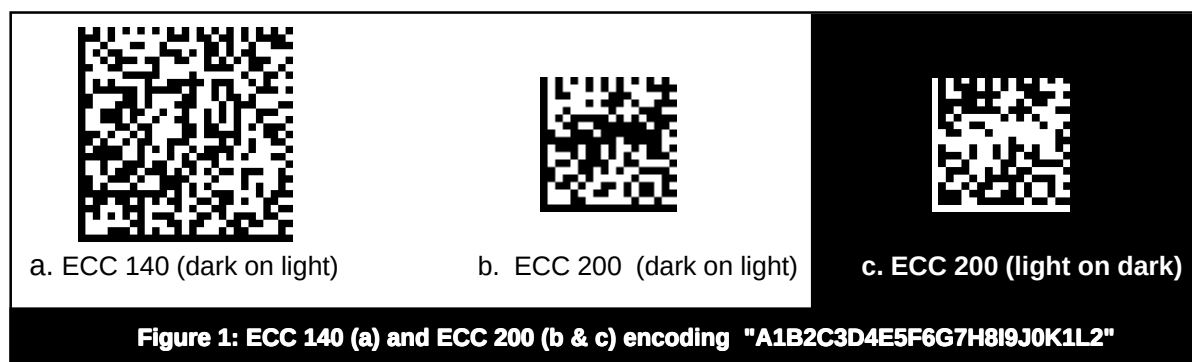
#### 4.2 Summary of Additional Features

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).
- 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 is surrounded by a finder pattern, and this shall 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.



#### 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.

#### 4.3.2 Symbol Sizes and Capacities

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 recognized by the upper right corner module being dark. The complete attributes of ECC 000 - 140 symbols are given in Annexe A.

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 zone. Some symbols are rectangular with sizes from 8 x 18 to 16 x 48 not including quiet zone. All ECC 200 symbols can be recognized by the upper right corner module being light. The complete attributes of ECC 200 symbols are given in Table 11 in Section 6.5.

## 5 ECC 000 - 140 Requirements

For new applications or open systems ECC 200 is recommended (See Section 6).

### 5.1 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

Annexe Q. The following steps convert user data to an ECC 000 - 140 symbol:

#### Step 1: Data Encodation

The user data is analyzed 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 Correcting

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 Unrandomized Bit Stream.

#### Step 5: Pattern Randomizing

The Unrandomized Bit Stream is processed by the pattern randomizing algorithm and produces the Randomized Bit Stream.

| 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                       |

**Table 1: Encodation Schemes**

Step 6: Module Placement in Matrix  
Modules are placed in a matrix to construct the finder pattern. The Randomized Bit Stream is placed into the matrix one module at a time according to the data module placement algorithm given in Annexe B.

Figure 2 shows the various bit streams during the encode process.

5.2 Data Encodation

The data shall be encoded using one of six encodation schemes (see Table 1).

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 Annexe C. 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.

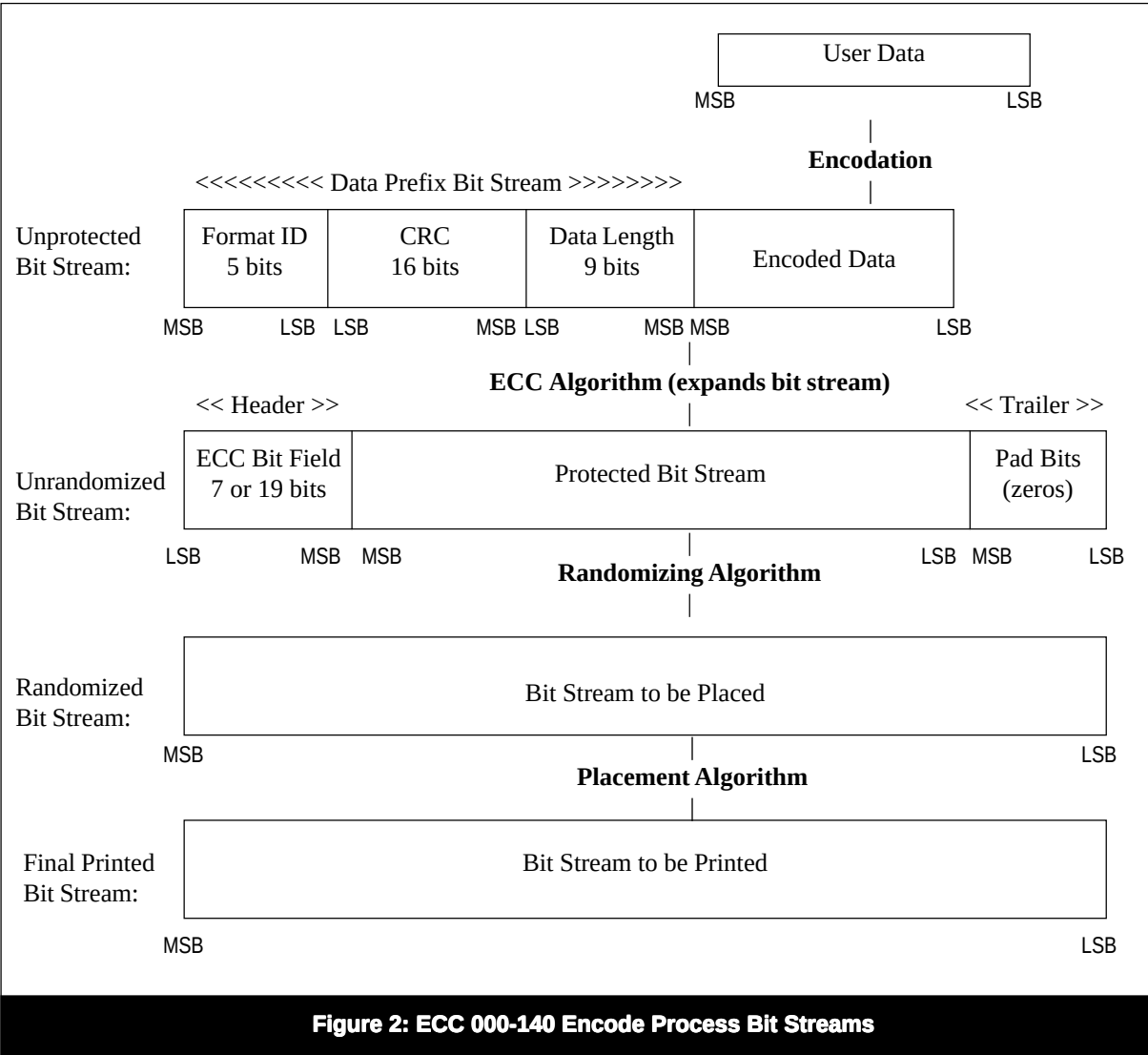


Figure 2: ECC 000-140 Encode Process Bit Streams

To determine the appropriate encodation scheme, the data to be encoded should be analyzed. 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 4-24 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.

#### 5.2.1 Base 11 - Numeric Encodation

The Base 11 (Numeric) Encodation scheme encodes 6 data characters as 21 bits, achieving an encodation 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 C.

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 Annex C.1.

#### 5.2.2 Base 27 - Upper-case Alphabetic Encodation

The Base 27 (Upper-case Alphabetic) Encodation scheme encodes 5 data characters as 24 bits, achieving an encodation 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 C.

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 Annex C.2.

#### 5.2.3 Base 37 - Upper-case Alphanumeric Encodation

The Base 37 (Upper-case Alphanumeric) Encodation scheme encodes 4 data characters as 21 bits, achieving an encodation 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 C.

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 Annex C.3.

#### 5.2.4 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 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 C.

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 Annex C.4.

#### 5.2.5 ASCII Encodation

The ASCII Encodation scheme enables all 128 characters from ANSI X3.4 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 C1 of Annex C.

### 5.2.6 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.

### 5.3 User Selection of Error Correction Level

### 5.3.1 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 2. 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.

| Error Correction Code Level | Maximum % Damage | Increase in User Bits from ECC 0 |
|-----------------------------|------------------|----------------------------------|
| 000                         | none             | none                             |
| 050                         | 28               | 33                               |
| 080                         | 5.5              | 50                               |
| 100                         | 12.6             | 100                              |
| 140                         | 25               | 300                              |

**Table 2: Error Correction Error Recovery and Overhead Percentages**

### 5.3.2 Additional Error Correction Levels Based on Convolutional Codes

Additional levels of error correction, based on convolutional code algorithms, are possible in Data Matrix. Symbols conforming to these constructions may be found in systems implemented prior to the publication of this specification. Information on these additional Error Correction levels is available from AIM USA.

#### 5.4 Constructing the Unprotected Bit Stream

Figure 2 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.

#### 5.4.1 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 3.

| 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                |     |

### Table 3: Encoding the Format ID

### 5.4.2 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. Annexe D describes the complete procedure for generating the CRC Value.

### 5.4.3 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.

#### 5.4.4 Data Prefix Construction

The Data Prefix Bit Stream is constructed as 30 bits as illustrated in Figure 3.

|                     |                      |                       |
|---------------------|----------------------|-----------------------|
| Format ID<br>5 bits | CRC Value<br>16 bits | Data Length<br>9 bits |
| MSB    LSB          | LSB                  | MSB    LSB    MSB     |

Note: Some bit fields start with the MSB others start with LSB

### Figure 3: Structure of Data Prefix Bit Stream

#### 5.4.5 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.

### 5.5 Constructing the Unrandomized Bit Stream

Figure 2 illustrates that the Unrandomized Bit

Stream has three constituent parts:

- a. Header
- b. Protected Bit Stream
- c. Trailer

The component parts shall be generated as defined below.

#### 5.5.1 Header Construction

The header of the Unrandomized 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 4.

| ECC Level | Binary Segment Identifier |         |
|-----------|---------------------------|---------|
|           | MSB                       | LSB     |
| 000       |                           | 1111110 |
| 050       | 0001110000000001          | 110     |
| 080       | 1110001110000001          | 110     |
| 100       | 1111111110000001          | 110     |
| 140       | 1111110001110001          | 110     |

**Table 4: ECC Bit Field**

#### 5.5.2 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 5.3. 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 Annexe E.

#### 5.5.3 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 Unrandomized Bit Stream shall be an odd integer between 7 and 47.

This ensures that the symbol is square.

#### 5.5.4 Completing the Unrandomized Bit Stream

The Protected Bit Stream, with the header and trailer added, is called the Unrandomized Bit Stream and is shown in Figure 2.

#### 5.6 Pattern Randomizing

The Unrandomized Bit Stream is processed by the pattern randomizing algorithm and produces the Randomized Bit Stream. The pattern randomizing algorithm consists of a bitwise XOR operation between the Unrandomized Bit Stream and the Master Random Bit Stream as given in Annexe F starting with the MSB position and continuing for the length of the Unrandomized Bit Stream.

#### 5.7 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 Section 5.5.3. The Randomized Bit Stream is placed into the matrix one module at a time according to the data module placement grids given in Annexe B.

The finder pattern (as defined in Section 4.3.1) shall be placed to produce an external border to the data module grid.

## 6 ECC 200 Requirements

### 6.1 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 Annexe R. The following steps convert user data to an ECC 200 symbol:

#### Step 1: Data Encodation

Analyze 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 out the required number of codewords. If the user does not specify the matrix size, then

| Encodation Scheme | Characters                        | Bits per Data Character |
|-------------------|-----------------------------------|-------------------------|
| ASCII             | double digit numerics             | 4                       |
|                   | ASCII values 0-127                | 8                       |
|                   | Extended ASCII values 128-255     | 16                      |
| C40               | Primarily upper-case alphanumeric | 5.33                    |
| Text              | Primarily lower-case alphanumeric | 5.33                    |
| X12               | ANSI X12 EDI data set             | 5.33                    |
| EDIFACT           | ASCII values 32-94                | 6                       |
| Base 256          | All byte values 0-255             | 8                       |

**Table 5: Encodation Schemes for ECC 200**

choose the smallest size that accommodates the data. A complete list of matrix sizes is shown in Section 6.5, Table 11.

**Step 2: Error Checking and Correcting 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 Annexe G. 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.

## 6.2 Data Encodation

### 6.2.1 Overview

The data may be encoded using any combination of six encodation schemes (see Table 5).

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 5 need to be interpreted carefully. The best scheme for a given set of data may not be the one with the fewest bits per data character. If the highest degree of compaction is required, account has to be taken of switching between code sets and of switching to and from encodation schemes (see Annexe S). It should also be noted that even if the number of codewords is minimized, the codeword stream may need to be expanded to fill a symbol. This fill process is done using pad characters.

### 6.2.2 Default Character Interpretation

The default character interpretation for character values 0 to 127 shall conform to ANSI X3.4. 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 Section 6.4. The default interpretation corresponds to ECI 000003.

### 6.2.3 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 Section 6.2.4.2). 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 6.

| 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                            |

**Table 6: ASCII Encodation Values**

#### 6.2.4 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 Section 6.2.4.1 to 6.2.4.9. These symbology control characters are found predominantly in the ASCII encodation set.

##### 6.2.4.1 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.

##### 6.2.4.2 Upper Shift Character

The Upper Shift character is used to encode a single extended ASCII character (values 128-255). 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 code set.

In ASCII encodation, the Upper Shift character is represented by codeword 235. The reduced data

value (i.e. ASCII value - 128) is transformed into its codeword value by adding 1.

For example, to encode ¥ (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.

##### 6.2.4.3 Pad Character

If the encoded data, irrespective of the encodation scheme in force, does not fill the data capacity of the symbol, pad characters shall be added (prefixed by codeword 254 or Unlatch if necessary) to fill out the remaining data capacity of the symbol. The pad characters shall only be used for this purpose.

The 253-State pattern randomizing algorithm is applied to the pad characters starting at the second pad character and continuing to the end of the symbol (see Annexe H.1).

##### 6.2.4.4 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 Section 6.4.

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.

##### 6.2.4.5 Shift Characters in C40 and Text Encodation

The three shift characters in C40 and Text encodation are used to switch for a single data character to one of the subsets of these encodation schemes.

##### 6.2.4.6 FNC1 Alternate Data Type Identifier

To encode data to conform to specific industry standards as authorized by AIM International, 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 GS control character (ASCII value 29).

#### 6.2.4.7 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.

The two Macro characters apply only when in the first symbol character position. They shall not be used in conjunction with Structured Append and their functions are defined in Table 7. The header will be transmitted as a prefix to the data stream and the trailer will be transmitted as a suffix to the data stream. The symbology identifier, if used, will precede the header.

| Macro Codeword | Name     | Interpretation          |                            |
|----------------|----------|-------------------------|----------------------------|
|                |          | Header                  | Trailer                    |
| 236            | 05 Macro | $\text{D} >^R_s 05^G_s$ | $^R_s \text{E} \text{O}_T$ |
| 237            | 06 Macro | $\text{D} >^R_s 06^G_s$ | $^R_s \text{E} \text{O}_T$ |

**Table 7: Macro Functions**

#### 6.2.4.8 Structured Append Character

A Structured Append character is used to indicate that the symbol is part of a Structured Append according to the rules defined in Section 6.6.

#### 6.2.4.9 Reader Programming Character

A Reader Programming character indicates that the symbol encodes a message used to program the reader system. The Reader Programming character should appear as the first codeword and Reader

Programming should not be used with Structured Append.

#### 6.2.5 C40 Encodation

C40 encodation enables three alphanumeric data characters to be packed into two codewords. The upper-case alphabetic characters, numerics and space can each be encoded as a single C40 value. All other ASCII characters and special characters require two C40 values starting with one of three shift characters. A C40 shift character only affects the single C40 value following the shift character. The full C40 character set assignments are shown in Annex J.1.

##### 6.2.5.1 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.

##### 6.2.5.2 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 ( $C_1$ ,  $C_2$ ,  $C_3$ ) shall be encoded as:

$$(1600 * C_1) + (40 * C_2) + C_3 + 1$$

which produces a value 1 to 64000. Figure 4 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

|                                      |                                            |
|--------------------------------------|--------------------------------------------|
| 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 | $23307 \text{ div } 256 = 91$              |
| 2nd Codeword: (16-bit value) mod 256 | $23307 \text{ mod } 256 = 11$              |
| Codewords                            | 91, 11                                     |

**Figure 4: Example of C40 Encoding**

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 whenever an odd number of symbol characters remain in the symbol in the C40 encodation scheme.

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.

#### 6.2.5.3 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 Annexe J.1; V has been used to indicate the appropriate C40 value.

#### 6.2.6 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 Annexe J.2.

##### 6.2.6.1 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.

##### 6.2.6.2 Text Encodation Rules

The rules for C40 encodation apply.

#### 6.2.7 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 8. There are no shift characters in the ANSI X12 encodation set.

| X12 Value | Encoded Characters          | ASCII Values |
|-----------|-----------------------------|--------------|
| 0         | X12 segment terminator <CR> | 13           |
| 1         | X12 segment separator *     | 42           |
| 2         | X12 subelement separator >  | 62           |
| 3         | space                       | 32           |
| 4-13      | 0-9                         | 48-57        |
| 14-39     | A-Z                         | 65-90        |

**Table 8: ANSI X12 Encodation Set****6.2.7.1 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.

The rules of C40 encodation apply. The exception is at the end of encoding ANSI X12 data. If the data characters do not fully utilize 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.

**6.2.8 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.

**6.2.8.1 Switching to and from EDIFACT Encodation**

It is possible to switch to EDIFACT encodation from

**6.2.7.2 ANSI X12 Encodation Rules**

| 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 5: The Relationship Between the EDIFACT Value and the 8-bit Byte Value**

| Data characters            | D           | A           | T           | A        |
|----------------------------|-------------|-------------|-------------|----------|
| Binary values (Annexe J.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 6: Example of EDIFACT Encodation**

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.

#### 6.2.8.2 EDIFACT Encodation Rules

The EDIFACT encodation character set is defined in Annexe J.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 5.

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 6.

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.

#### 6.2.9 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 6.2.2

The 255-State pattern randomizing algorithm is applied to each Base 256 sequence within the encoded data (see Annexe H.2). It starts after the latch to Base 256 encodation and ends at the last character specified by the Base 256 field length.

##### 6.2.9.1 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.

##### 6.2.9.2 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 9 specifies how the field length is defined.

Thereafter, all encodation shall be of the byte values.

| 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<br>d2 = length MOD 250 | d1 = 250 to 255<br>d2 = 0 to 249 |

**Table 9: Base 256 field Length**

### 6.3 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. It is also possible to append up to 16 ECC 200 symbols.

#### 6.3.1 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 Section 6.4) 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).

#### 6.3.2 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 Section 6.5.

### 6.4 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:

- international character sets (or code pages)
- general purpose interpretations such as encryption and compaction
- user defined interpretations for closed systems
- control information for structured append in unbuffered mode.

The Extended Channel Interpretation protocol is fully specified in the *Extended Channel Interpretation (ECI) Assignments* document. 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.

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.

#### 6.4.1 Encoding ECIs

The various encodation schemes of Data Matrix for ECC 200 (defined in Table 5) 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 are 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 10.

The following examples illustrate the encodation:

ECI = 015000

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

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

= [241][186][142]

ECI = 090000

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]

#### 6.4.2 ECIs and Structured Append

ECIs may occur anywhere in the message encoded in a single or Structured Append (see Section 6.6) set

| ECI Assignment Value | Codeword Sequence                                                    | Codeword Values                                                                                                   | Ranges                                                                                      |
|----------------------|----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| 000000 to 000126     | C <sub>0</sub><br>C <sub>1</sub>                                     | 241<br>ECI_no + 1                                                                                                 | C <sub>1</sub> = (1 to 127)                                                                 |
| 000127 to 016382     | C <sub>0</sub><br>C <sub>1</sub><br>C <sub>2</sub>                   | 241<br>(ECI_no - 127) div 254 + 128<br>(ECI_no - 127) mod 254 + 1                                                 | C <sub>1</sub> = (128 to 191)<br>C <sub>2</sub> = (1 to 254)                                |
| 0016383 to 999999    | C <sub>0</sub><br>C <sub>1</sub><br>C <sub>2</sub><br>C <sub>3</sub> | 241<br>(ECI_no - 16383) div 64516 + 192<br>[(ECI_no - 16383) div 254] mod 254 + 1<br>(ECI_no - 16383) mod 254 + 1 | C <sub>1</sub> = (192 to 207)<br>C <sub>2</sub> = (1 to 254)<br>C <sub>3</sub> = (1 to 254) |

**Table 10: Encoding ECI Assignment Numbers in ECC 200**

| Symbol Size <sup>1</sup>                                                                                                                                                                                                                                                                                                                  |     | Data Region |        | Mapping Matrix Size | Total Codewords |       | Reed-Solomon Block |       | Inter-leaved Blocks | Data Capacity |                |           | Error Correction Overhead % | Max. Correctable Codeword Error/Erasure |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|-------------|--------|---------------------|-----------------|-------|--------------------|-------|---------------------|---------------|----------------|-----------|-----------------------------|-----------------------------------------|
| Row                                                                                                                                                                                                                                                                                                                                       | Col | Size        | Number |                     | Data            | Error | Data               | Error |                     | Num. Cap.     | Alphanum. Cap. | Byte Cap. |                             |                                         |
| 10                                                                                                                                                                                                                                                                                                                                        | 10  | 8x8         | 1      | 8x8                 | 3               | 5     | 3                  | 5     | 1                   | 6             | 3              | 1         | 62.5                        | 2/+                                     |
| 12                                                                                                                                                                                                                                                                                                                                        | 12  | 10x10       | 1      | 10x10               | 5               | 7     | 5                  | 7     | 1                   | 10            | 6              | 3         | 58.3                        | 3/+                                     |
| 14                                                                                                                                                                                                                                                                                                                                        | 14  | 12x12       | 1      | 12x12               | 8               | 10    | 8                  | 10    | 1                   | 16            | 10             | 6         | 55.6                        | 5/7                                     |
| 16                                                                                                                                                                                                                                                                                                                                        | 16  | 14x14       | 1      | 14x14               | 12              | 12    | 12                 | 12    | 1                   | 24            | 16             | 10        | 50                          | 6/9                                     |
| 18                                                                                                                                                                                                                                                                                                                                        | 18  | 16x16       | 1      | 16x16               | 18              | 14    | 18                 | 14    | 1                   | 36            | 25             | 16        | 43.8                        | 7/11                                    |
| 20                                                                                                                                                                                                                                                                                                                                        | 20  | 18x18       | 1      | 18x18               | 22              | 18    | 22                 | 18    | 1                   | 44            | 31             | 20        | 45                          | 9/15                                    |
| 22                                                                                                                                                                                                                                                                                                                                        | 22  | 20x20       | 1      | 20x20               | 30              | 20    | 30                 | 20    | 1                   | 60            | 43             | 28        | 40                          | 10/17                                   |
| 24                                                                                                                                                                                                                                                                                                                                        | 24  | 22x22       | 1      | 22x22               | 36              | 24    | 36                 | 24    | 1                   | 72            | 52             | 34        | 40                          | 12/21                                   |
| 26                                                                                                                                                                                                                                                                                                                                        | 26  | 24x24       | 1      | 24x24               | 44              | 28    | 44                 | 28    | 1                   | 88            | 64             | 42        | 38.9                        | 14/25                                   |
| 32                                                                                                                                                                                                                                                                                                                                        | 32  | 14x14       | 4      | 28x28               | 62              | 36    | 62                 | 36    | 1                   | 124           | 91             | 60        | 36.7                        | 18/33                                   |
| 36                                                                                                                                                                                                                                                                                                                                        | 36  | 16x16       | 4      | 32x32               | 86              | 42    | 86                 | 42    | 1                   | 172           | 127            | 84        | 32.8                        | 21/39                                   |
| 40                                                                                                                                                                                                                                                                                                                                        | 40  | 18x18       | 4      | 36x36               | 114             | 48    | 114                | 48    | 1                   | 228           | 169            | 112       | 29.6                        | 24/45                                   |
| 44                                                                                                                                                                                                                                                                                                                                        | 44  | 20x20       | 4      | 40x40               | 144             | 56    | 144                | 56    | 1                   | 288           | 214            | 142       | 28                          | 28/53                                   |
| 48                                                                                                                                                                                                                                                                                                                                        | 48  | 22x22       | 4      | 44x44               | 174             | 68    | 174                | 68    | 1                   | 348           | 259            | 172       | 28.1                        | 34/65                                   |
| 52                                                                                                                                                                                                                                                                                                                                        | 52  | 24x24       | 4      | 48x48               | 204             | 84    | 102                | 42    | 2                   | 408           | 304            | 202       | 29.2                        | 42/78                                   |
| 64                                                                                                                                                                                                                                                                                                                                        | 64  | 14x14       | 16     | 56x56               | 280             | 112   | 140                | 56    | 2                   | 560           | 418            | 278       | 28.6                        | 56/106                                  |
| 72                                                                                                                                                                                                                                                                                                                                        | 72  | 16x16       | 16     | 64x64               | 368             | 144   | 92                 | 36    | 4                   | 736           | 550            | 366       | 28.1                        | 72/132                                  |
| 80                                                                                                                                                                                                                                                                                                                                        | 80  | 18x18       | 16     | 72x72               | 456             | 192   | 114                | 48    | 4                   | 912           | 682            | 454       | 29.6                        | 96/180                                  |
| 88                                                                                                                                                                                                                                                                                                                                        | 88  | 20x20       | 16     | 80x80               | 576             | 224   | 144                | 56    | 4                   | 1152          | 862            | 574       | 28                          | 112/212                                 |
| 96                                                                                                                                                                                                                                                                                                                                        | 96  | 22x22       | 16     | 88x88               | 696             | 272   | 174                | 68    | 4                   | 1392          | 1042           | 694       | 28.1                        | 136/260                                 |
| 104                                                                                                                                                                                                                                                                                                                                       | 104 | 24x24       | 16     | 96x96               | 816             | 336   | 136                | 56    | 6                   | 1632          | 1222           | 814       | 29.2                        | 168/318                                 |
| 120                                                                                                                                                                                                                                                                                                                                       | 120 | 18x18       | 36     | 108x108             | 1050            | 408   | 175                | 68    | 6                   | 2100          | 1573           | 1048      | 28                          | 204/390                                 |
| 132                                                                                                                                                                                                                                                                                                                                       | 132 | 20x20       | 36     | 120x120             | 1304            | 496   | 163                | 62    | 8                   | 2608          | 1954           | 1302      | 27.6                        | 248/472                                 |
| 144                                                                                                                                                                                                                                                                                                                                       | 144 | 22x22       | 36     | 132x132             | 1558            | 620   | 156                | 62    | 8*                  | 3116          | 2335           | 1556      | 28.5                        | 310/590                                 |
|                                                                                                                                                                                                                                                                                                                                           |     |             |        |                     |                 |       | 155                | 62    | 2*                  |               |                |           |                             |                                         |
| Rectangular Symbols                                                                                                                                                                                                                                                                                                                       |     |             |        |                     |                 |       |                    |       |                     |               |                |           |                             |                                         |
| 8                                                                                                                                                                                                                                                                                                                                         | 18  | 6x16        | 1      | 6x16                | 5               | 7     | 5                  | 7     | 1                   | 10            | 6              | 3         | 58.3                        | 3/+                                     |
| 8                                                                                                                                                                                                                                                                                                                                         | 32  | 6x14        | 2      | 6x28                | 10              | 11    | 10                 | 11    | 1                   | 20            | 13             | 8         | 52.4                        | 5/+                                     |
| 12                                                                                                                                                                                                                                                                                                                                        | 26  | 10x24       | 1      | 10x24               | 16              | 14    | 16                 | 14    | 1                   | 32            | 22             | 14        | 46.7                        | 7/11                                    |
| 12                                                                                                                                                                                                                                                                                                                                        | 36  | 10x16       | 2      | 10x32               | 22              | 18    | 22                 | 18    | 1                   | 44            | 31             | 20        | 45.0                        | 9/15                                    |
| 16                                                                                                                                                                                                                                                                                                                                        | 36  | 14x16       | 2      | 14x32               | 32              | 24    | 32                 | 24    | 1                   | 64            | 46             | 30        | 42.9                        | 12/21                                   |
| 16                                                                                                                                                                                                                                                                                                                                        | 48  | 14x22       | 2      | 14x44               | 49              | 28    | 49                 | 28    | 1                   | 98            | 72             | 47        | 36.4                        | 14/25                                   |
| Note 1: Symbol size does not include quiet zone.<br>*Note: In the largest symbol (144x144), the first eight Reed-Solomon blocks shall be 218 codewords long encoding 156 data codewords. The last two blocks shall encode 217 codewords (155 data codewords). All the blocks have 62 error correction codewords.<br>+Note: Does not apply |     |             |        |                     |                 |       |                    |       |                     |               |                |           |                             |                                         |
| Table 11: ECC 200 Symbol Attributes                                                                                                                                                                                                                                                                                                       |     |             |        |                     |                 |       |                    |       |                     |               |                |           |                             |                                         |

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.

#### 6.4.3 Post-Decode Protocol

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

### 6.5 ECC 200 Symbol Attributes

#### 6.5.1 Symbol Sizes and Capacity

There are 24 square symbols and 6 rectangular symbols available in ECC 200. These are as specified in Table 11.

#### 6.5.2 Insertion of Alignment Patterns into Larger Symbols

As shown in Table 11, square symbols 32x32 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. The square symbols are divided into 4, 16, or 36 data regions (as illustrated in Annexe K.1, K.2, and K.3). The rectangular symbols are divided into two data regions (as illustrated in Annexe K.4). These data regions are bounded by alignment patterns. 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.

### 6.6 Structured Append

#### 6.6.1 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.

#### 6.6.2 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 to those defined in Table 12.

#### 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       |

#### 6.6.3 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.

| 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      |

**Table 12: Structured Append Symbol Position Bits**

#### 6.6.4 FNC1 and Structured Append

If Structured Append is used in conjunction with FNC1 (see Section 6.2.4.6), 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.

#### 6.6.5 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 defines a control block that shall be prefixed to the beginning of each transmission.

#### 6.7 Error Detection and 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 Annexe G.

Each ECC 200 symbol employs a specific combination of interleaving and number of error correction codewords. These combinations are given in Table 11.

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 Annexe L.1.

##### 6.7.1 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 Annexe L.1).

The data codewords are the coefficients of the terms of a polynomial with the coefficient of the highest term being the first data codeword and 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 7. The registers  $b_0$  through  $b_{k-1}$  are initialized as zeros. There are two phases to generate the encoding. In the first phase, with the switch in the down position the data codewords are passed both to the output and the circuit. The first phase is complete after  $n$  clock pulses. In the second phase ( $n+1 \dots n+k$  clock pulses), with the switch in the up position, the error correction codewords  $\epsilon_{k-1} \dots \epsilon_0$  are generated by flushing the registers in order and complementing the output while keeping the data input at 0. The codewords output from the shift register are in the order that they are to be placed in the symbol. If interleaving is used, the codewords will not be placed in consecutive symbol characters. (See Annexe G).

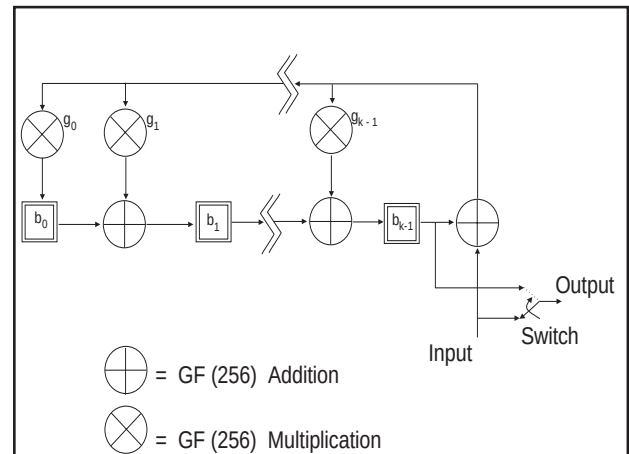


Figure 7: Error Correction Codeword Encoding Circuit

##### 6.7.2 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 correctable is given by the following formula:

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

where:

- e = number of erasures
- t = number of errors
- d = number of error correction codewords
- p = number of 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 (10x10, 12x12, 8x18, and 8x32), erasure correction should not be used ( $e = 0$  and  $p = 1$ ).

### 6.8 Symbol Construction

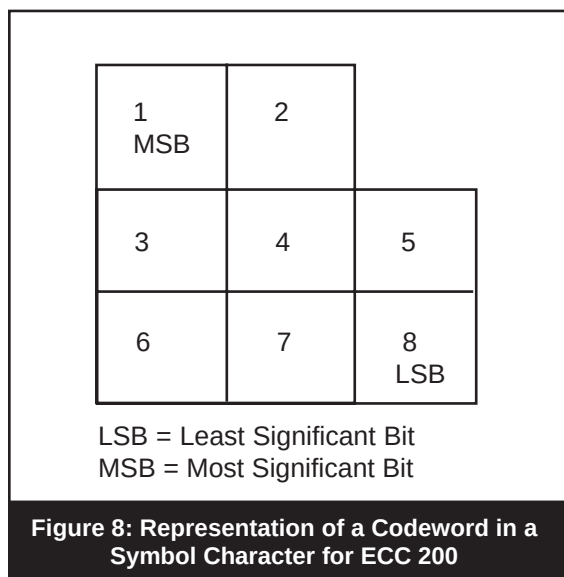
Given the codeword sequence obtained in the previous sections, an ECC 200 symbol is constructed using the following steps:

1. Place codeword modules in a mapping matrix
2. Insert alignment pattern modules, if any
3. Place finder modules along the perimeter

#### 6.8.1 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 8.

Because the symbol character shape defined in Figure 8 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 Annexe M.1, described in Annexe M.2 and illustrated in Annexe M.3.



#### 6.8.2 Alignment Pattern Module Placement

This step is only needed for larger matrices:

- square: 32x32 and larger
- rectangular: 8x32, 12x36 and larger

The mapping matrix is sub-divided into data regions, of the sizes defined in Table 11, 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 Annexe K.1 - K.3. For rectangular matrices, only a single vertical alignment pattern is placed between the data regions as shown in Annexe K.4.

#### 6.8.3 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.

## 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 2X minimum                                                                                                     |

### 7.2 Quiet zone

The minimum quiet zone is equal to one module width on all four sides.

## 8 Symbol Quality

Data Matrix symbols shall be assessed for quality using the 2D matrix bar code symbol print quality guidelines presented in Annexe N, as augmented and modified below.

### 8.1 Obtaining the Test Image

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

### 8.2 Symbol Quality Parameters

#### 8.2.1 Decode

The reference decode algorithm presented in Section 9 shall be applied to the test image. If it results in a successful decode of the entire data message, then Decode passes with a grade of "A" (4.0), otherwise it fails with a grade of "F" (0.0).

#### 8.2.2 Symbol Contrast

Symbol contrast shall be graded using all of the grey-scale pixel values in the test image which fall within the symbol's boundary, including the 1X wide quiet zones, as determined by the reference decode.

The procedure is defined in Annexe N.2.2.

#### 8.2.3 "Print" Growth

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

Taking the top and right side of the symbol independently (since horizontal and vertical growth can differ substantially), proceed from the L-boundary along each perimeter peak line and sum the number of light ( $N_L$ ) and dark ( $N_D$ ) pixels encountered, traversing the full width or height of the symbol but excluding the final dark corner module on ECC 000 - 140 symbols which have an odd number of modules per side. The resulting measure of print growth in each direction is  $D = N_D / (N_L + N_D)$ , which shall be graded against  $D_{NOM} = 0.50$  with  $D_{MIN} = 0.35$  and  $D_{MAX} = 0.65$  as prescribed in Annexe N.2.3. The Print Growth grade shall be the lower of those achieved along the top and right side of the symbol.

#### 8.2.4 Axial Nonuniformity

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

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

#### 8.2.5 Unused Error Correction

For ECC 000 - ECC 140 versions of Data Matrix, the convolutional error encoding can correct for the following percentages of bit errors:

| Grade  | Reference Decode | Symbol Contrast | "Print" Growth              | Axial Nonuniformity | Unused Error Correction |
|--------|------------------|-----------------|-----------------------------|---------------------|-------------------------|
| A(4.0) | Passes           | $SC \geq 70\%$  | $-0.50 \leq D' \leq 0.50$   | $AN \leq 0.06$      | $UEC \geq 0.62$         |
| B(3.0) |                  | $SC \geq 55\%$  | $-0.70 \leq D' \leq 0.70$   | $AN \leq 0.08$      | $UEC \geq 0.50$         |
| C(2.0) |                  | $SC \geq 40\%$  | $-0.85 \leq D' \leq 0.85$   | $AN \leq 0.10$      | $UEC \geq 0.37$         |
| D(1.0) |                  | $SC \geq 20\%$  | $-1.00 \leq D' \leq 1.00$   | $AN \leq 0.12$      | $UEC \geq 0.25$         |
| F(0.0) | Fails            | $SC < 20\%$     | $D' < -1.00$ or $D' > 1.00$ | $AN > 0.12$         | $UEC < 0.25$            |

**Table 13: Summary of Data Matrix Symbol Print Quality Parameters**

ECC 000:  $E_{MAX} = 0.0\%$   
ECC 050:  $E_{MAX} = 2.8\%$   
ECC 080:  $E_{MAX} = 5.5\%$   
ECC 100:  $E_{MAX} = 12.6\%$   
ECC 140:  $E_{MAX} = 25.0\%$

Following a successful decode, the actual percentage of bit errors  $E_{ACT}$  is simply the number of bits that were corrected divided by the total number of bits in the symbol's data fields. The Unused Error Correction for these symbols is  $UEC = 1.0 - (E_{ACT} / E_{MAX})$  which is then graded using the table in Annex N.2.5.

The ECC 200 version of Data Matrix employs Reed-Solomon error correction. The smaller symbols contain just a single error correcting block while the larger symbols are subdivided into two or more interleaved blocks. In all cases, each Reed-Solomon block shall be graded independently as specified in Annex N.2.5, then the Unused Error Correction grade shall be the lowest achieved by any of the blocks.

### 8.3 Overall Symbol Grade

The overall print quality grade for a Data Matrix symbol is the lowest of the five grades achieved above. Table 13 summarizes the test criteria.

### 8.4 Process Control Measurements

Several tools and methods are available which can perform useful measurements for monitoring and controlling the process of creating Data Matrix symbols. These include:

1. Symbol contrast readings from a linear bar code verifier.
2. Horizontal print growth from an ECC 200 reference symbol.
3. Determination of axial nonuniformity by

physical measurement

4. Visual inspection of the perimeter pattern for grid nonuniformities and defects.

These tools and methods are described in Annex T.

## 9 Reference Decode Algorithm for Data Matrix

This reference decode algorithm finds the symbols in an image and decodes them. This is the decode algorithm used in the determination of symbol quality. This decode algorithm refers to black and white states in the image. The algorithm is actually applied to the normal image and again with black and white states reversed.

- A. Find the candidate areas which may contain an "L" shaped finder pattern:
  1. Select a global threshold midway between the maximum reflectance and minimum reflectance in the image. Convert the image to black and white using the global threshold.
  2. Define a distance  $d_{min}$  which is the length of the shortest expected leg of an imaged symbol's "L" finder pattern ( $d_{min}$  will depend on the pixel density, camera focal length, maximum reading distance, and minimum X-dimension.)
  3. Find all the areas with closed continuous boundaries of white on the outside and black on the inside. Note that one candidate area may be totally contained within another candidate area.
  4. Disregard all areas which have a longest axis less than  $0.5 d_{min}$ .
  5. Form new merged candidate areas by merging any adjacent areas with a small

separation:

- a. If two areas are separated by less than  $0.15 d_{\min}$ , form a new area by joining the two areas across the minimum separation point.
  - b. Keep the two adjacent areas and the merged area as three candidate areas.
  - c. Repeat the merging process to form new unique merged areas from all original and merged area combinations.
6. Form new divided candidate areas from any candidate areas with a narrow neck separating large sub-areas, ignoring the merged areas from the previous step:
    - a. If an area contains a neck less than  $0.15 d_{\min}$  wide at its narrowest and separates two sub-areas each with a longest axis of at least  $d_{\min}$ , create two new candidate areas by breaking the original area into two new areas across the narrowest part of the neck.
    - b. Keep the original larger area and the two new divided areas as three candidate areas.
    - c. Repeat the dividing process to form new divided areas from all original and divided areas.
  7. Ignore any area with a longest axis less than  $d_{\min}$ . The remaining candidate areas will be processed in the following steps to try to decode valid symbols.
- B. Find the “L” shaped finder pattern in a candidate area:
1. Sort the points in the area’s boundary into sets that fall along straight lines, within 3% of the longest axis of the candidate area. For each set of points, plot a line which minimizes the sum of distances between each of the points in the set and the closest point on the line.
  2. Select the line with the most points. Select a second line with the most number of points which is also perpendicular to the first line to within 10 degrees. If the second line does not contain at least 20% of the number of points in the first line, disregard the area.
  3. The two lines may form the outline of two sides of the “L” in a symbol’s finder pattern.
- C. Find two new lines which are a weighted best fit to the outside of the “L” in the area around the corner of the “L”:
1. Plot the intersection of the two lines. For each of the two sets of points, calculate the weights for each of the distances from the points to the line as follows.
  2. Divide the distance from the farthest point in the set and the intersection of the two “L” lines by  $d_{\min}$ . Discard the remainder. Call the quotient N.
  3. Divide the set of points into N subsets in sequence starting from the point closest to the intersection of the two lines and ending with the farthest point.
  4. Weight the point-to-line distances for the subset of points closest to the intersection by weight N, the next subset’s distances by N-1, ..., and the farthest subset’s distances by one.
  5. For each side, find a new line which minimizes the sum of the weighted distances between the new line and the side’s set of points.
- 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 normalized 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 in step C.5 as shown in Figure 9. Define the two equal areas formed by the bisecting line as the right side and 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”

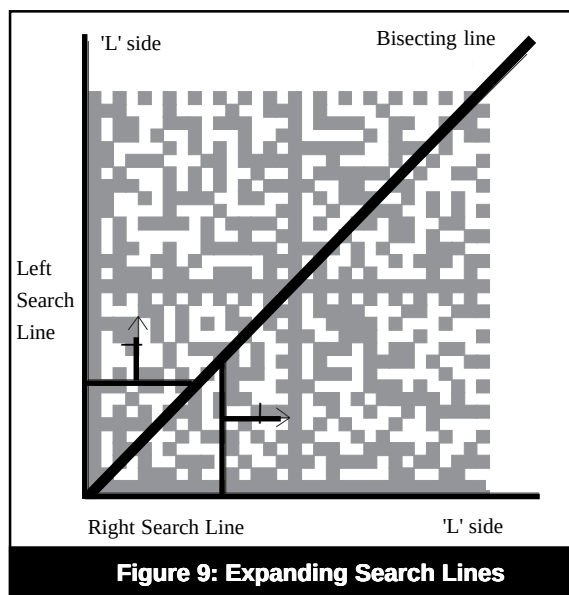


**Figure 9: Expanding Search Lines**

side divided by the current length of the search line measured between the two bounding lines:

$$T = (\text{number of transitions}) \times (\text{"L" max line length}) / (\text{search line length}).$$

This formula normalizes 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%.

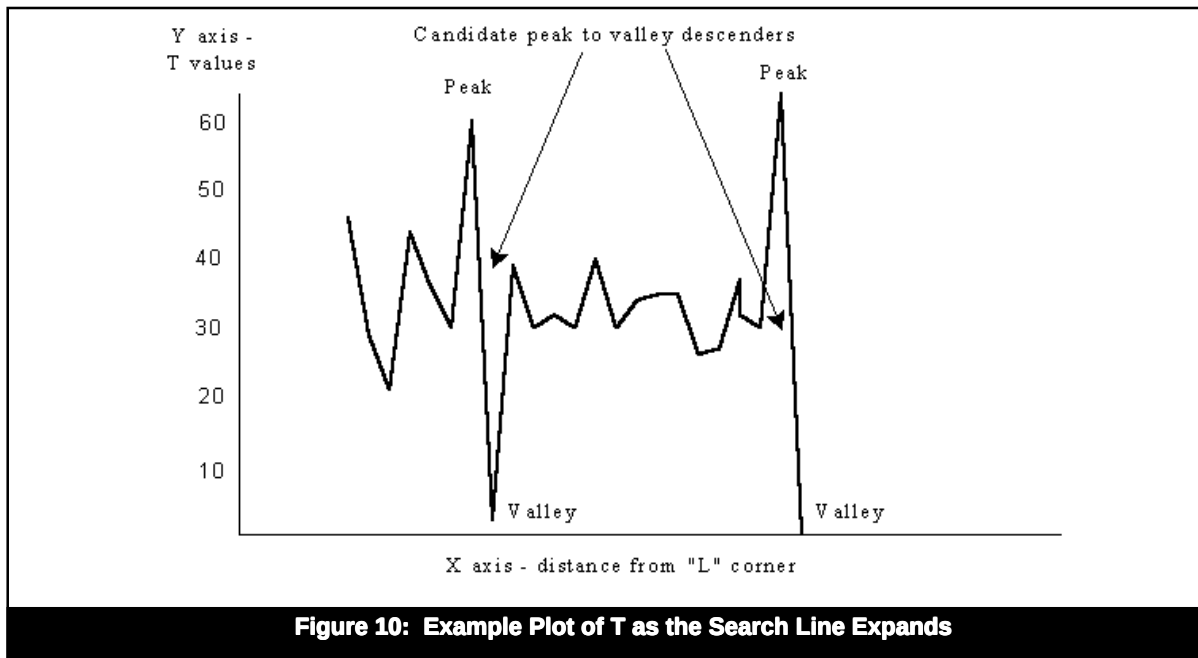
4. Form a plot of the T values for each side, where the Y axis is the T value and the X axis is the search line's distance from the corner of the "L". A sample plot is shown in Figure 10.
5. Starting from the T value with the smallest X in the right side's plot and then increasing X, find the first instance of a descending line where the T 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 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 center 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.
  2. In each rectangular region, sort the black/white boundary points into sets which fit a straight line as in step B.1. Select the set with the most points and find the minimum distance line as in step B.1. This line should pass along the outside of the alternating pattern, shown as the "best fit line" in Figure 12.
  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:

$$\text{Offset} = \text{length of peak line} / (\text{number of transitions} \times 2)$$

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



- 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 each bounded mid-line, measure the edge-to-edge distances between all the similar edges (note that this will include measurements across all adjacent sets of 2, 4, 6, ... modules).
  3. Sort the edge-to-edge measurements into sets where the edge-to-edge distances within each set differ by less than plus or minus 25% from the geometric mean of the set's two extremes.
  4. Calculate the average of the largest set of distances for the side. Call the average  $E_{avg}$ .
  5. 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.
  6. If the number of edge-to-edge measurements in the set is greater than "dm", delete the excess number from the edge-to-edge measurements found in step F.2 which differ most from  $E_{avg}$ . Recalculate  $E_{avg}$  and a new "dm" according to steps F.4 and F.5 using the remaining edge-to-edge measurements in the set. This should remove any spurious short and long edge-to-edge measurements caused by spots and voids.
  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 center points of the alternating pattern modules:
1. Divide the  $M_d$  by the sum of "dm" and 1.5 to form the module pitch:  

$$\text{Module pitch} = M_d / (dm + 1.5).$$
  2. Calculate "dm" module centers along the step E.3 line with the first being 1.5 times the module pitch from the "L" side and each successive center spaced one module pitch further along the step E.3 line.
- H. Plot the data module sampling grid in the data region by extending the alternating pattern