INTERNATIONAL STANDARD

ISO/IEC 24728

First edition
2006-06-01

Information technology — Automatic identification and data capture techniques — MicroPDF417 bar code symbology specification

Technologies de l'information — Techniques d'identification automatique et de capture des données — Spécifications pour la symbologie de code à barres MicroPDF417

Reference number
ISO/IEC 24728:2006(E)

Contents

Page

Foreword

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

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

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

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

ISO/IEC 24728 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic Identification and data capture techniques.*

This International Standard contains many provisions which are identical with those of ISO/IEC 15438.

Introduction

MicroPDF417 is a multi-row symbology, derived from and closely based on PDF417. MicroPDF417 is designed for applications with a need for improved area efficiency but without the requirement for PDF417's maximum data capacity. A limited set of symbol sizes is available, together with a fixed level of error correction for each symbol size. Module dimensions are user-specified to enable symbol production and reading by a wide variety of techniques.

Since MicroPDF417's data character encodation, its error correction method, and many of its other symbol characteristics are, and are intended to remain, identical to those of PDF417, descriptions of these characteristics are quoted verbatim from the PDF417 symbology specification (ISO/IEC 15438) wherever appropriate, or with the appropriate modifications. For ease of cross-reference, this International Standard follows a similar document structure, with minor differences in clause/subclause numbering, to ISO/IEC 15438.

Information technology — Automatic identification and data capture techniques — MicroPDF417 bar code symbology specification

1 Scope

This International Standard specifies the requirements for the bar code symbology known as MicroPDF417. It specifies the MicroPDF417 symbology characteristics, data character encodation, symbol formats, dimensions, error correction rules, decoding algorithm, and a number of application parameters.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

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

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

ISO/IEC 15417, *Information technology — Automatic identification and data capture techniques — Bar code symbology specification — Code 128*

ISO/IEC 15418, *Information technology — EAN/UCC Application Identifiers and Fact Data Identifiers and Maintenance*

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

ISO/IEC 24723, *Information technology — Automatic identification and data capture techniques — EAN.UCC Composite bar code symbology specification*

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

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

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

GS1 General Specification [2]

---

[1] Published by AIM Global, 125 Warrendale-Bayne Road, Suite 100, Warrendale, PA 15086, USA.

[2] Published by GS1, Blue Tower, Avenue Louise 326, bte 10, B-1050 Brussels, Belgium.

3 Terms and definitions

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

3.1
application identifier
sequence of 2 to 4 digits, used to define the nature or use of the subsequent data characters, in accordance with ISO/IEC 15418 and the GS1 General Specification

3.2
bar-space sequence
sequence which represents the module widths of the elements of a symbol character

3.3
basic channel mode
standard system for encoding and transmitting bar code data where data message bytes are output from the decoder but no control information about the message is transmitted

3.4
cluster
any of three mutually exclusive subsets of PDF417 symbol characters, also used in MicroPDF417; the symbol characters in a given cluster conform with particular structural rules which are used in decoding the symbology

3.5
compaction mode
any of three data compaction algorithms in PDF417 (Text, Numeric and Byte Compaction modes), also used in MicroPDF417, which are used to map 8-bit data bytes efficiently to PDF417 codewords

3.6
e-distance
measurement from the leading edge of one element to the leading edge of the next element, or from the trailing edge of an element to the trailing edge of the next element, in a symbol character or Row Address Pattern

3.7
error correction codeword
codeword which encodes a value derived from the error correction codeword algorithm to enable decode errors to be detected and, depending on the number of error correction codewords, to be corrected

3.8
Extended Channel Interpretation
ECI
procedure within some symbologies, including MicroPDF417, to replace the default interpretation with another interpretation in a reliable manner

NOTE The interpretation intended prior to producing the symbol can be retrieved after decoding the scanned symbol to recreate the data message in its original format.

3.9
extended channel model
system for encoding and transmitting both data message bytes and control information about the message, the control information being communicated using Extended Channel Interpretation (ECI) escape sequences

3.10
family
set of MicroPDF417 symbol versions sharing the same number of columns and the same RAP rotation

3.11
function codeword
codeword which initiates a particular operation within a symbology, for example to switch between data encoding sets, to invoke a compaction scheme, to program the reader, or to invoke Extended Channel Interpretations

3.12
Global Label Identifier
GLI
procedure in the PDF417 symbology which behaves in a similar manner to Extended Channel Interpretation

NOTE The GLI system was the symbology-dependent precursor to the symbology-independent ECI system.

3.13
macro character
character representing a predefined sequence of data characters (e.g. an industry-specific message header) used to reduce the number of symbol characters needed to encode data in a symbol using certain structured formats

3.14
Mode Latch codeword
codeword which is used to switch from one mode to another mode, which stays in effect until another latch or shift codeword is implicitly or explicitly brought into use, or until the end of the symbol is reached

3.15
Mode Shift codeword
codeword which is used to switch from one mode to another for one codeword, after which encoding returns to the original mode

3.16
RAP rotation
difference between the number designating a Center or Right Row Address Pattern and the number designating the nearest Row Address Pattern to the left, in the same row of a symbol

3.17
Row Address Patterns
special patterns made up of three bars and three spaces occupying ten modules that serve both as start or (with the stop bar) stop patterns and as row indicators in MicroPDF417 symbols

3.18
stop bar
single-module bar, adjoining the rightmost Row Address Pattern, which forms the right boundary of the symbol

3.19
Structured Append
procedure within the MicroPDF417 symbology to distribute data logically from a computer file across a number of related symbols

NOTE 1 This procedure is identical to the Macro PDF417 feature of PDF417.

NOTE 2 The procedure considerably extends the data capacity beyond that of a single symbol.

3.20
UCC/EAN-128
subset of Code 128 symbols as defined in ISO/IEC 15417, reserved for use in accordance with GS1 General Specification

4 Symbols, operations and abbreviated terms

4.1 Symbols

For the purposes of this document, the following mathematical symbols apply. There are some cases where the symbols below have been used in a different manner in an equation. This has been done for consistency with a more general use of the notation and is always clearly defined in the text.

b	the element width in a symbol character
c	number of columns in the symbol in the data region (excluding Row Address Patterns)
d	data codeword including ECI Descriptor and all function codewords
E	error correction codeword
e	an edge to similar edge dimension in a symbol character
f	number of substitution errors
F	row number
H	height of symbol including quiet zone
K	cluster number
k	number of error correction codewords
L	number of erasures
m	number of source data codewords (including the codeword in the ECI Descriptor position but prior to the addition of any pad codewords)
n	total number of data codewords (including ECI Descriptor and pad codewords)
p	pitch or width of a symbol character
Q_H	horizontal quiet zone
Q_V	vertical quiet zone
r	number of rows in the symbol
W	width of symbol including quiet zone
X	X-dimension or module width
Y	module height (also called row height)

4.2 Mathematical operations

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

div	is the integer division operator, rounding down
INT	is the integer value i.e. where a number is rounded down to its whole number component, ignoring its decimal fractions
mod	is the positive integer remainder after division. If the remainder is negative, then add the value of the divisor in order to make the result positive. For example, the remainder of –29 160 divided by 929 is –361, which when added to 929 yields 568

4.3 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

AI	Application Identifier
EC	Error Correction
ECI	Extended Channel Interpretation
GLI	Global Label Identifier
RAP	Row Address Pattern

5 Requirements

5.1 Symbology characteristics

5.1.1 Basic characteristics

MicroPDF417 is a multi-row symbology which may be utilized by applications needing to encode a moderate amount of data in a two-dimensional symbol (up to 150 bytes, 250 alphanumeric characters, or 366 numeric digits), and when minimizing symbol size is a primary concern. MicroPDF417 is identical to PDF417 in terms of its encodation modes, error correction method, and symbol character sets. However, MicroPDF417 replaces PDF417's 17-module-wide start/stop patterns and left/right row indicators with a unique set of 10-module-wide Row Address Patterns, which were designed both to reduce overall symbol width and to facilitate linear scanning at row heights as low as 2*X*. MicroPDF417, unlike PDF417, may only be printed in certain defined combinations of *r* (number of rows), *c* (number of columns), and *k* (number of error correction codewords), up to a maximum of four data columns by 44 rows (see 5.2.2).

MicroPDF417 has the following basic characteristics:

a) Encodable character set:

   1) Text Compaction mode (see 5.4.2) permits all printable ASCII characters to be encoded, i.e. values 32 to 126 inclusive in accordance with ISO/IEC 646, as well as selected control characters.

   2) Byte Compaction mode (see 5.4.3) permits all 256 possible 8-bit byte values to be encoded. This includes all ASCII characters value 0 to 127 inclusive and provides for international character set support.

   3) Numeric Compaction mode (see 5.4.4) permits efficient encoding of numeric data strings.

   4) Up to 811 800 different character sets or data interpretations.

   5) Various function codewords for control purposes.

b) Symbol character structure: (*n, k, m*) characters of 17 modules (*n*), 4 bar and 4 space elements (*k*), with the largest element 6 modules wide (*m*).

c) Maximum possible number of data characters per symbol (for a maximum size MicroPDF417 symbol): 125 data codewords which can encode

   1) Text Compaction Mode: 250 characters (2 data characters per codeword).

   2) Byte Compaction mode: 150 characters (1,2 data characters per codeword).

   3) Numeric Compaction mode: 366 characters (2,93 data characters per codeword).

d) Symbol Size:

   1) Number of rows: 4 to 44 (available in defined combinations with number of columns).

   2) Number of data columns: either one, two, three, or four

   3) Width in modules: 40*X*, 57*X*, 84*X*, or 101*X* including quiet zones

   4) Maximum codeword capacity: 176 codewords.

   5) Maximum data codeword capacity: 125 codewords.

e) Number of error correction codewords: fixed for each available row/column combination, ranging from 7 to 50 codewords per symbol and reserving from 28% to 67% of codewords for error detection and correction, depending on symbol size.

f) Non-data overhead: per row: 23 modules for the one- and two-column versions; 33 modules for the three- or four-column version, including quiet zones.

g) Code type: continuous, multi-row bar code symbology.

h) Character self-checking: Yes.

i) Bi-directionally decodable: Yes.

5.1.2 Summary of additional features

The following are additional features in MicroPDF417:

a) **Data compaction**: Three schemes are defined to compact a number of data characters into codewords. Generally data is not directly represented on a one character for one codeword basis (see 5.4.2 to 5.4.4).

b) **Extended Channel Interpretations**: These mechanisms allow up to 811 800 different data character sets or interpretations to be encoded (see 5.5).

c) **Structured Append**: MicroPDF417 uses the Macro PDF417 mechanism for Structured Append. This mechanism allows files of data to be represented logically and consecutively in a number of MicroPDF417 symbols. Up to 99 999 different MicroPDF417 symbols can be so linked or concatenated and be scanned in any sequence to enable the original data file to be correctly reconstructed (see 5.13).

d) **Edge to edge decodable**: MicroPDF417 can be decoded by measuring elements from edge to similar edge (see 5.3.1).

e) **Cross-row scanning**: The combination of three characteristics in MicroPDF417 facilitates cross-row scanning:

   - being synchronised horizontally, or self clocking
   - row identification
   - being vertically synchronised, by using the cluster values to achieve local row discrimination.

This combination allows a single linear scan to cross a number of rows and achieve a partial decode of the data so long as at least one complete symbol character per row is decoded into its codeword. The decoding algorithm can then place the individual codewords into a meaningful matrix.

f) **Code 128 emulation codewords**: special codewords may be used at the start of a MicroPDF417 symbol as a signal to the decoder to transmit the symbol's data as if it were encoded in a Code 128 symbol. The Symbology Identifier of the resulting transmission can thus signal the presence of an implied FNC1 flag, for special applications (see 5.4.1.5).

5.2 Symbol structure

5.2.1 MicroPDF417 symbol parameters

Each MicroPDF417 symbol consists of a stack of vertically-aligned rows (with a minimum of 4 and maximum of 44 rows); the allowable numbers of rows are specified separately for each of the one-, two-, three- or four-column versions. Each row shall include a minimum of 1 symbol character and a maximum of 4 symbol characters, excluding the Row Address Pattern columns. The symbol shall include a quiet zone on all four sides.

Figure 1 shows sample one-, two-, three-, and four-column MicroPDF417 symbols, each with twenty rows.

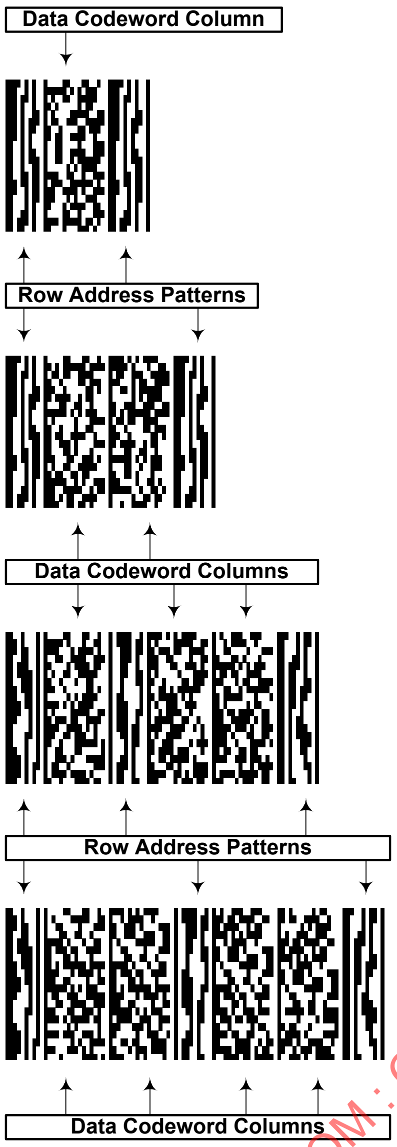


Figure 1 — Four MicroPDF417 symbols

5.2.2 Row and column combinations

MicroPDF417 symbols shall conform with certain predefined combinations of numbers of rows, columns, and number of error-correction codewords. These versions are defined in Table 1.

Table 1 — MicroPDF417 version characteristics

Number of Data Columns (*c*)	Number of Rows (*r*)	Total CWs in Data Region	Number of EC CWs (*k*)	% of CWs for EC	Number of non-EC CWs	Number of CWs for Data (Note 1)	Max Data Bytes (Note 2)	Max Alpha chars (Note 3)	Max Digits (Note 3)	Symbol Width, in *X* (Note 4)	Symbol Height, in *X* (Note 5)
1	11	11	7	64	4	3	3	6	8	40	24
1	14	14	7	50	7	6	7	12	17	40	30
1	17	17	7	41	10	9	10	18	26	40	36
1	20	20	8	40	12	11	13	22	32	40	42
1	24	24	8	33	16	15	18	30	44	40	50
1	28	28	8	29	20	19	22	38	55	40	58
2	8	16	8	50	8	7	8	14	20	57	18
2	11	22	9	41	13	12	14	24	35	57	24
2	14	28	9	32	19	18	21	36	52	57	30
2	17	34	10	29	24	23	27	46	67	57	36
2	20	40	11	28	29	28	33	56	82	57	42
2	23	46	13	28	33	32	38	64	93	57	48
2	26	52	15	29	37	36	43	72	105	57	54
3	6	18	12	67	6	5	6	10	14	84	14
3	8	24	14	58	10	9	10	18	26	84	18
3	10	30	16	53	14	13	15	26	38	84	22
3	12	36	18	50	18	17	20	34	49	84	26
3	15	45	21	47	24	23	27	46	67	84	32
3	20	60	26	43	34	33	39	66	96	84	42
3	26	78	32	41	46	45	54	90	132	84	54
3	32	96	38	40	58	57	68	114	167	84	66
3	38	114	44	39	70	69	82	138	202	84	78
3	44	132	50	38	82	81	97	162	237	84	90
4	4	16	8	50	8	7	8	14	20	101	10
4	6	24	12	50	12	11	13	22	32	101	14
4	8	32	14	44	18	17	20	34	49	101	18
4	10	40	16	40	24	23	27	46	67	101	22
4	12	48	18	38	30	29	34	58	85	101	26
4	15	60	21	35	39	38	45	76	111	101	32
4	20	80	26	33	54	53	63	106	155	101	42
4	26	104	32	31	72	71	85	142	208	101	54
4	32	128	38	30	90	89	106	178	261	101	66
4	38	152	44	29	108	107	128	214	313	101	78
4	44	176	50	28	126	125	150	250	366	101	90

CW = Codeword; EC = Error correction

NOTE 1 Excludes EC codewords and codeword in the ECI Descriptor position
NOTE 2 Assumes remaining in default Byte Compaction Mode (see 5.4)
NOTE 3 Assumes that a Latch to Text or Numeric Compaction mode replaces the ECI Descriptor
NOTE 4 Including 1*X* quiet zones on either side
NOTE 5 Assumes *Y* = 2*X*; includes 1*X* quiet zones at top and bottom

5.2.3 Row parameters

Each row within a MicroPDF417 symbol has the following structure (examples are shown in Figure 1):

1. a leading Quiet Zone (see 5.8.3);
2. a Left MicroPDF417 Row Address Pattern (see 5.2.5);
3. one of the following:
   a) for the one-column version, one PDF417 codeword, or
   b) for the two-column version, two PDF417 codewords, or
   c) for the three-column version, one PDF417 codeword followed by a Center Row Address Pattern (see 5.2.5) and two more PDF417 codewords, or
   d) for the four-column version, two PDF417 codewords followed by a Center Row Address Pattern and two more PDF417 codewords;
4. a Right MicroPDF417 Row Address Pattern;
5. a one-module stop bar; (see 5.2.5);
6. a trailing Quiet Zone (see 5.8.3).

All MicroPDF417 symbols contain at least two Row Address Pattern columns (separated from each other by one or two data columns), and every codeword column is adjacent to at least one Row Address Pattern column.

5.2.4 Codeword sequence

5.2.4.1 Overall sequence

A MicroPDF417 symbol may contain up to 176 symbol characters or codewords. Symbol character is the more appropriate term to refer to the printed bar/space pattern; codeword is more appropriate for the numeric value of the symbol character. The codewords shall follow this sequence:

a. The first codeword in a MicroPDF417 symbol may have a special meaning as an ECI Descriptor (see 5.2.4.2).

b. The data codewords shall follow, from the most significant encodable character. Codewords with values above 899 may be inserted to change the compaction mode, to perform other special functions, or to indicate ECIs.

c. Pad codewords as needed to enable the codeword sequence to be represented in the number of rows and columns specified for the selected MicroPDF417 version (usually the smallest of sufficient capacity).

d. An optional Structured Append Control Block.

e. Error correction codewords for error detection and correction.

The codewords are arranged with the most significant codeword at the upper left corner of the matrix, and are encoded from left to right, top row to bottom. In the three- and four-column versions, the center Row Address Pattern does not change the codeword sequence (that is, the first three or four codewords of the sequence are placed left to right in the first row, the Center Row Address Pattern being inserted before the last two codewords in the row; the next three or four codewords are placed in the same manner in the second row, and so on). Figure 2 illustrates in layout format the sequence for a 14-row, two-column MicroPDF417 symbol.

Left RAP_1	d_{18}	d_{17}	Right RAP_1
Left RAP_2	d_{16}	d_{15}	Right RAP_2
Left RAP_3	d_{14}	d_{13}	Right RAP_3
Left RAP_4	d_{12}	d_{11}	Right RAP_4
Left RAP_5	d_{10}	d_9	Right RAP_5
Left RAP_6	d_8	d_7	Right RAP_6
Left RAP_7	d_6	d_5	Right RAP_7
Left RAP_8	d_4	d_3	Right RAP_8
Left RAP_9	d_2	d_1	Right RAP_9
Left RAP_{10}	d_0	e_8	Right RAP_{10}
Left RAP_{11}	e_7	e_6	Right RAP_{11}
Left RAP_{12}	e_5	e_4	Right RAP_{12}
Left RAP_{13}	e_3	e_2	Right RAP_{13}
Left RAP_{14}	e_1	e_0	Right RAP_{14}

where:

RAP_j is the appropriate Left or Right Row Address Pattern for row j of this symbol version

d_{18}	=	first codeword (ECI Descriptor or function codeword)
d_{17} - d_0	=	encoded data, together with pad codewords if required
e_8 - e_0	=	Error Correction codewords

Figure 2 — MicroPDF417 example of symbol layout schematic

The rules and advice for structuring the matrix are included in 5.9.

5.2.4.2 ECI Descriptor codeword

Because MicroPDF417 is intended for applications where symbol size is critical, it is expected that MicroPDF417 will frequently be used in conjunction with a Transformation ECI appropriate for the data requirements of the application. MicroPDF417 supports efficient encoding of a single Transformation ECI sequence at the start of each data stream by defining an optional ECI Descriptor codeword that can be encoded at the start of the first symbol of each MicroPDF417-encoded data stream. This Transformation ECI co-exists with the Interpretative ECI (whether default or explicitly encoded) currently in effect, since Interpretative ECIs and Transformation ECIs have independent scopes.

If the input to the MicroPDF417 encoder calls for a Transformation ECI in the range 000900 to 001799 to be encoded at the start of the symbol (or at the start of the first symbol of a Structured Append series), then the MicroPDF417 encoder may encode that ECI in shortened form as a single data codeword (range 0 to 899), as the first codeword in the data region. This codeword is calculated as (ECI_number MOD 900). Otherwise the

MicroPDF417 encoder shall encode a codeword of 900 or greater as the first codeword, operating as a standard MicroPDF417 function codeword (see 5.4.1).

EXAMPLE A Transformation ECI 001002 needs to precede the data to be encoded. It can be efficiently encoded at the start of a MicroPDF417 symbol using the single codeword 102 (because 102 is the third codeword of the standard PDF417 representation [926] [0] [102] of that ECI. (See 5.5.1)).

Where no ECI needs to be encoded at the start of the symbol and the data is most efficiently represented in Text Compaction mode, the encoder should place a codeword 900 (Latch to Text Compaction) at the start of the symbol. Since the codeword is above 899, it will not be interpreted as an ECI Descriptor.

If the first codeword has a value less than 900, then once the MicroPDF417 decoder has successfully performed error correction, it shall, for the purposes of high-level decoding and transmission, interpret this codeword as if it were the third codeword of a Transformation ECI sequence (whose first two codewords are 926, 0). However, if the symbol is the second or later symbol within a Structured Append series, then this special interpretation shall not apply (see 5.13).

5.2.5 MicroPDF417 Row Address Patterns

MicroPDF417 Row Address Patterns replace the PDF417 Start and Stop patterns and Left and Right Row Indicators. These patterns are placed at the left and right ends of each row. A Center Row Address Pattern appears after the first codeword (reading left to right) of each row of a three-column MicroPDF417 symbol or the second codeword of a four-column MicroPDF417 symbol. The Row Address Patterns indicate each row number. Unlike the PDF417 Row Indicators, the Row Address Patterns do not explicitly encode the number of rows (*r*), the number of columns (*c*), nor the number of error correction codewords (*k*). For this reason, MicroPDF417 only supports fixed combinations of *r*, *c*, and *k* as specified in 5.2.2.

The Row Address Patterns in a MicroPDF417 symbol are (10,3) symbol characters, where 10 shows each character consists of ten modules, and 3 shows that it comprises three bars and three spaces. These (10,3) characters create two distinct pattern sets (one set is used for the left and right patterns, and the other set is used for the center patterns); each pattern encodes a pattern number from 1 to 52. The Row Address Patterns always begin with a bar and end with a space. There is a one-module stop bar after the rightmost space of the Right Row Address Pattern.

The sets of MicroPDF417 Row Address Patterns are arranged in an order such that vertically adjacent patterns only differ by a single one-module edge shift. Each of the 52 Pattern Numbers, as shown in Table 2, has two distinct patterns associated with it: one pattern when used as a Left or Right Row Address Pattern, and another pattern when used as a Center Row Address Pattern.

Table 2 — MicroPDF417 Row Address Patterns

Pattern	L or R							C							Pattern	L or R							C					
Number	B	S	B	S	B	S		B	S	B	S	B	S		Number	B	S	B	S	B	S		B	S	B	S	B	S
1	2	2	1	3	1	1		1	1	2	2	3	1		27	2	2	1	1	1	3		1	1	3	2	2	1
2	3	1	1	3	1	1		1	2	1	2	3	1		28	2	2	1	1	2	2		1	1	3	2	1	2
3	3	1	2	2	1	1		1	2	2	1	3	1		29	2	2	1	1	3	1		1	1	3	1	2	2
4	2	2	2	2	1	1		1	3	1	1	3	1		30	2	2	1	2	2	1		1	2	2	1	2	2
5	2	1	3	2	1	1		1	3	1	2	2	1		31	2	2	2	1	2	1		1	3	1	1	2	2
6	2	1	4	1	1	1		1	3	2	1	2	1		32	3	1	2	1	2	1		1	3	1	1	1	3
7	2	2	3	1	1	1		1	4	1	1	2	1		33	3	2	1	1	2	1		1	2	2	1	1	3
8	3	1	3	1	1	1		1	4	1	2	1	1		34	2	3	1	1	2	1		1	1	3	1	1	3
9	3	2	2	1	1	1		1	4	2	1	1	1		35	2	3	1	1	1	2		1	1	2	2	1	3
10	4	1	2	1	1	1		1	3	3	1	1	1		36	2	2	2	1	1	2		1	1	2	2	2	2
11	4	2	1	1	1	1		1	3	2	2	1	1		37	2	1	3	1	1	2		1	1	2	3	1	2
12	3	3	1	1	1	1		1	3	1	3	1	1		38	2	1	2	2	1	2		1	1	2	3	2	1
13	2	4	1	1	1	1		1	2	2	3	1	1		39	2	1	2	2	2	1		1	1	1	4	2	1
14	2	3	2	1	1	1		1	2	3	2	1	1		40	2	1	2	1	3	1		1	1	1	3	3	1
15	2	3	1	2	1	1		1	2	4	1	1	1		41	2	1	2	1	2	2		1	1	1	3	2	2
16	3	2	1	2	1	1		1	1	5	1	1	1		42	2	1	2	1	1	3		1	1	1	2	3	2
17	4	1	1	2	1	1		1	1	4	2	1	1		43	2	1	1	2	1	3		1	1	1	2	2	3
18	4	1	1	1	2	1		1	1	4	1	2	1		44	2	1	1	1	2	3		1	1	1	1	3	3
19	4	1	1	1	1	2		1	2	3	1	2	1		45	2	1	1	1	3	2		1	1	1	1	2	4
20	3	2	1	1	1	2		1	2	3	1	1	2		46	2	1	1	1	4	1		1	1	1	2	1	4
21	3	1	2	1	1	2		1	2	2	2	1	2		47	2	1	1	2	3	1		1	1	2	1	1	4
22	3	1	1	2	1	2		1	2	2	2	2	1		48	2	1	1	2	2	2		1	2	1	1	1	4
23	3	1	1	2	2	1		1	2	1	3	2	1		49	2	1	1	3	1	2		1	2	1	1	2	3
24	3	1	1	1	3	1		1	2	1	4	1	1		50	2	1	1	3	2	1		1	2	1	1	3	2
25	3	1	1	1	2	2		1	1	2	4	1	1		51	2	1	1	4	1	1		1	1	2	1	3	2
26	3	1	1	1	1	3		1	1	3	3	1	1		52	2	1	2	3	1	1		1	1	2	1	4	1

The specific arrangement of the Row Address Patterns minimizes the signal interference from adjacent rows, and thus maximizes the readability of the Row Address Patterns. The Reference Decode algorithm (see Annex J of this document) describes how any residual signal interference is detected and corrected, using the cluster number of the PDF417 codeword adjacent to the decoded Row Address Pattern.

Table 2 shows that the sequence of the 52 vertically adjacent row patterns is cyclical in nature and the minimum-difference property therefore also applies when comparing pattern 52 with pattern 1. Thus, the minimum-difference property also holds true if the patterns are "rotated" any arbitrary amount. After RAP rotation, it remains true that any two vertically adjacent patterns differ by only one module.

In order to facilitate discrimination between the versions of MicroPDF417, groups of versions utilize different predefined "RAP rotations" of the set of 52 patterns for the center (if present) and right Row Address Pattern columns relative to the nearest neighboring RAP column to the left. These "RAP rotations," and the mapping of these pattern combinations to row numbers, are described in 5.11, Low Level Encodation. In addition to providing row location information, MicroPDF417 Row Address Patterns also provide for the detection of scan direction across the symbol (i.e., whether a scan is a forward scan, in which case the Left Row Address Pattern will have been scanned first, or a reverse scan, in which case the stop bar and Right Row Address Pattern will have been scanned first). Table 2 shows that all of the Left and Right Row Address Patterns begin with a two-, three- or four-module bar. Since there is always a one-module stop bar following the MicroPDF417 Right Row Address Pattern, it is possible to detect a reverse scan of these patterns.

5.3 Basic encodation

5.3.1 Symbol character structure

MicroPDF417 utilizes the PDF417 symbol characters, with the exception of the PDF417 Start and Stop Characters.

Each symbol character shall consist of four bar elements and four space elements, each of which can be one to six modules wide. The four bar and four space elements shall measure 17 modules in total. MicroPDF417 symbol characters can be decoded by measuring the e-distances within the character.

Each symbol character is defined by an 8-digit sequence which represents the module widths of the eight elements of that symbol character. Figure 3 illustrates a symbol character with the bar-space sequence 51111125.

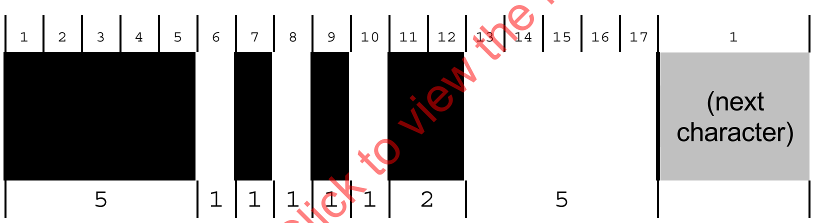


Figure 3 — A PDF417 symbol character

There are 929 defined symbol character values (codewords), numbered from 0 to 928.

The set of codewords is represented in three mutually exclusive symbol character sets, or clusters. Each cluster encodes the 929 available codewords into different bar-space patterns so that one cluster is distinct from another. The cluster numbers are 0, 3, 6. The cluster definition applies to all symbol characters, but not to the MicroPDF417 Row Address Patterns.

The cluster number K is defined by the following formula:

$$K = (b_1 - b_2 + b_3 - b_4 + 9) \bmod 9$$

where b_1, b_2, b_3 and b_4 represent the width in modules of the four bar elements respectively

The cluster number K for the symbol character in Figure 3 is:

$$K = (5 - 1 + 1 - 2 + 9) \bmod 9 = 3$$

The codewords and the bar-space sequences for each cluster of symbol characters are given in Annex A.

5.3.2 Start and Stop Patterns

The start character function is performed by the Left Row Address Pattern, and the stop character function by the Right Row Address Pattern together with the stop bar.

5.4 High level (data) encodation

High level encoding converts the data characters into their corresponding codewords.

Data compaction schemes shall be used to achieve high level encoding. Three modes are defined below, each of which defines a particular efficient mapping between user defined data and codeword sequences. MicroPDF417 utilises the three data compaction modes of PDF417:

- Text Compaction mode (see 5.4.2).
- Byte Compaction mode (see 5.4.3).
- Numeric Compaction mode (see 5.4.4).

A given string of data bytes may be represented by different codeword sequences, depending on how the encoder switches between compaction modes and sub-modes. There is no single specified way to encode data in a MicroPDF417 symbol.

900 codewords are available in each mode for data encodation and other functions within the mode. The remaining 29 codewords are assigned to specific functions (see 5.4.1) independent of the current compaction mode. If data codewords appear before any ECI is invoked in the symbol then the first codeword must be a function codeword. This may require the use of a codeword 901 (latch to Byte Compaction mode) even though this is the default compaction mode (see 5.4.3).

MicroPDF417 also supports the Extended Channel Interpretation system, which allows different interpretations of data to be accurately encoded in the symbol (see 5.5).

5.4.1 Function codewords

Codewords 900 to 928 are assigned as function codewords as follows:

- for switching between modes (see 5.4.1.1)
- for enhanced applications using Extended Channel Interpretations (ECIs) (see 5.4.1.2)
- for other enhanced applications (see 5.4.1.3 through 5.4.1.7)

At present codewords 919 is reserved. Table 3 defines the complete list of assigned and reserved function codewords. Their functions are defined in 5.4.1.1 to 5.4.1.7. See 5.4.6 for the treatment of reserved codewords.

Table 3 — Assignments of function codewords

Codeword	Function	See clause
900	Mode Latch to Text Compaction mode	5.4.1.1
901	Mode Latch to Byte Compaction mode	5.4.1.1
902	Mode Latch to Numeric Compaction mode	5.4.1.1
903	UCC/EAN-128 emulation: transmit]C1 or]L3; implied Mode Latch 900	5.4.1.5
904	UCC/EAN-128 emulation: transmit]C1 or]L3; implied Mode Latch 902	5.4.1.5
905	UCC/EAN-128 emulation: transmit]C1 or]L3; implied Mode Latch 902, followed by implied “01” AI and 14-digit expansion of first 13 digits	5.4.1.5
906	“Linked” UCC/EAN-128: transmit]C1 or]L3; implied Mode Latch 900	5.4.1.5
907	“Linked” UCC/EAN-128: transmit]C1 or]L3; implied Mode Latch 902	5.4.1.5
908	Code 128 emulation: transmit]C2 or]L4; implied Mode Latch 900	5.4.1.5
909	Code 128 emulation: transmit]C2 or]L4; implied Mode Latch 902	5.4.1.5
910	Code 128 emulation: transmit]C0 or]L5; implied Mode Latch 900	5.4.1.5
911	Code 128 emulation: transmit]C0 or]L5; implied Mode Latch 902	5.4.1.5
912	"Linked" UCC/EAN-128, with leading date field: transmit]C1 or]L3; implied Mode Latch 902; specially encoded date field.	5.4.1.5
913	mode shift to Byte Compaction mode	5.4.1.1
914	"Linked" UCC/EAN-128, with lot number: transmit]C1 or]L3; implied Mode Latch 902, followed by implied "10" AI.	5.4.1.5
915	"Linked" UCC/EAN-128, with serial number: transmit]C1 or]L3; implied Mode Latch 902; followed by implied "21" AI.	5.4.1.5
916	05 Macro strings added to transmitted data; implied mode latch 902	5.4.1.7
917	06 Macro strings added to transmitted data; implied mode latch 900	5.4.1.7
918	linkage flag to associated linear component, in a composite symbol (other than an EAN.UCC Composite symbol)	5.4.1.6
919	reserved	
920	linkage flag to associated linear component, in an EAN.UCC Composite symbol	5.4.1.6
921	reader initialisation	5.4.1.4
922	terminator codeword for Structured Append control block	5.13
923	sequence tag to identify the beginning of optional fields in the Structured Append control block	5.13
924	mode latch to Byte Compaction mode (used differently from 901)	5.4.1.1
925 to 927	identifier for an Extended Channel Interpretation (ECI)	5.5
928	Structured Append marker codeword to indicate the beginning of a Structured Append Control Block	5.13

5.4.1.1 Function codewords for mode switching

In one MicroPDF417 symbol it is possible to switch back and forth between modes as often as required. Advice about selecting the appropriate modes is given in 5.4.5.

A mode latch codeword may be used to switch from the current mode to the indicated destination mode which stays in effect until another mode switch is explicitly brought into use. Codewords 900 to 902 and 924 are assigned to this function. Table 3 defines their function.

The mode shift codeword 913 shall cause a temporary switch from Text Compaction mode to Byte Compaction mode. This switch shall be in effect for only the next codeword, after which the mode shall revert to the prevailing sub-mode of the Text Compaction mode. Codeword 913 is only available in Text Compaction mode; its use is described in 5.4.2.4.

Table 4 — Mode definition and mode switching codewords

Destination mode	Mode latch	Mode shift
Text Compaction	900	▨
Byte Compaction	901/924	913
Numeric Compaction	902	▨

NOTE The table identifies the codeword which shall be used to switch to the defined mode.

The switching rules between the three modes are defined in Table 5 and shown schematically in Figure 4.

Table 5 — Mode transition table, showing codewords and their function

Original mode	Destination mode		
	Text	Byte	Numeric
Text	900 mode latch	913 mode shift 901 mode latch 924 mode latch	902 mode latch
Byte	900 mode latch	901 mode latch 924 mode latch	902 mode latch
Numeric	900 mode latch	901 mode latch 924 mode latch	902 mode latch

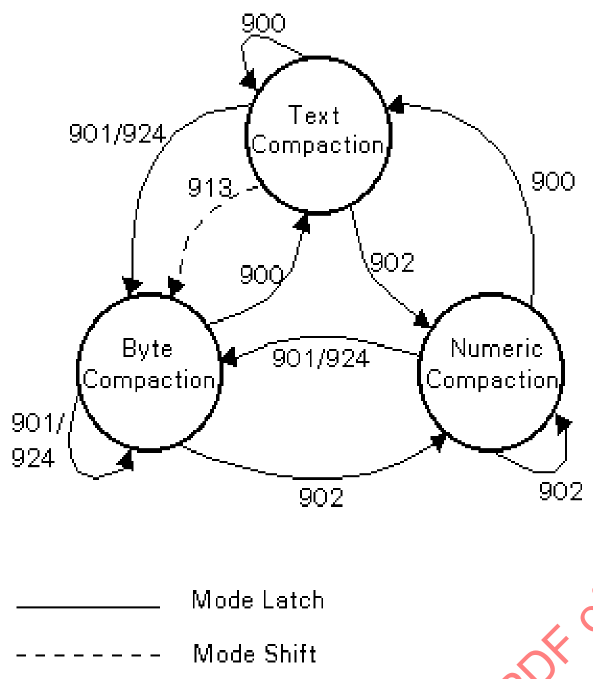


Figure 4 — Available mode switching

The switching rules into Byte Compaction mode are more fully defined in 5.4.3.1.

5.4.1.2 Function codewords for switching to Extended Channel Interpretations

An ECI codeword can be used to switch to a particular interpretation, which stays in effect until another ECI codeword is explicitly brought into use or until the end of the data. Codewords 925 to 927 are assigned to this function (see 5.5).

5.4.1.3 Function codewords for Structured Append

Structured Append symbols (see 5.13) shall use codeword 928 at the start of the Structured Append Control Block. Codewords 922 and 923 are used for special functions in Structured Append.

5.4.1.4 Function codeword for reader initialization

Codeword 921 shall be used to instruct the reader to interpret the data contained within the symbol as programming for reader initialisation. If used, codeword 921 shall appear as the first codeword. In the case of a Structured Append initialisation sequence, Codeword 921 shall appear in every symbol.

The data contained in an initialisation symbol, or sequence of symbols, shall not be transmitted by the reader.

5.4.1.5 Function codewords for Code 128 emulation

Codewords 903 through 915 (excluding 913), if they appear in the first codeword position (that is, in place of the ECI Descriptor codeword), indicate that the MicroPDF417 symbol's data output shall conform with the Code 128 specification as defined in ISO/IEC 15417. Each of these codewords implies a mode latch to either Text Compaction Mode (with an implied **ml** latch to Mixed sub-mode character) (see 5.4.2) or Numeric

Compaction Mode (see 5.4.4); see Table 3 for the specific codeword assignments. If Symbology Identifiers are enabled, then each such symbol's transmission shall be prefixed by either "]Cm" or "]Lm", where the modifier "m" is as shown in Table 3. MicroPDF417 readers shall be configurable to transmit either symbology identifier prefix, as required by the application.

Whether or not a codeword in the range 903 to 905 appears in the first codeword position, codewords in that range may appear in subsequent codeword positions as a field delimiter, similar to the use of FNC1 in Code 128 symbols. The reader shall transmit any such delimiter codeword as ASCII character 29 ($^{G}{}_{S}$). When appearing in other than the first codeword position, codeword 903 implies a latch to the Alpha sub-mode of Text Compaction mode, codeword 904 implies a latch to the Mixed sub-mode of Text Compaction mode, and codeword 905 implies a latch to Numeric Compaction mode. A 905 delimiter codeword, following a series of codewords in Numeric Compaction mode, serves to terminate the current grouping and then restarts Numeric Compaction mode. The appearance of an adjacent pair of repeated 903, 904 or 905 codewords will cause the decoder to divide the transmission at that point (but without transmitting the two implied ASCII 29 ($^{G}{}_{S}$) characters themselves); both substrings shall begin with the appropriate Symbology Identifier prefix.

In addition to the features just described, these Code 128 emulation codewords provide additional functionality as follows:

- Codewords 903 through 907, 912, 914 and 915 emulate UCC/EAN-128 symbols. In order to conform fully with that application standard:

    - It is the encoder's responsibility to ensure that strings longer than 48 characters are divided into two or more shorter strings by placing a single or double delimiter codeword (903, 904 or 905) before an AI; and

    - It is the reader's responsibility to break up long transmissions at doubled delimiter (two 903, 904 or 905 codewords) boundaries, or at single delimiter boundaries if necessary, to create multiple transmissions no more than 48 characters in length.

- Codeword 905 provides additional compaction when the message to be encoded begins with an Application Identifier "01" representing an SSC-14 number. In this case, then the leading "01" is not encoded (but will be transmitted by the reader), nor is the fourteenth digit of the SSC-14 number encoded (a mod 10 check digit is calculated and transmitted by the reader).

- Codewords 906 and 907 indicate UCC/EAN-128 emulation, but also indicate that a linear symbol printed below the MicroPDF symbol is "linked" to the data of the MicroPDF symbol. When the reader decodes a MicroPDF417 symbol beginning with 906 or 907, the reader shall not transmit the MicroPDF417 data unless a linear symbol is also found and decoded within the scanner's field of view. If successful, the decoder shall transmit the data from the linear symbol first (using the appropriate symbology identifier for the linear symbol) followed by the MicroPDF417 data in UCC/EAN-128 format as a second transmission. If the linear symbol is a UCC/EAN-128 symbol, then the MicroPDF417 data may be concatenated to the linear symbol's data, and sent as a single transmission (if the 48-character limit is not exceeded).

- Codewords 908 and 909 cause the decoder's symbology identifier prefix to be set to]C2 or]L4, indicating that the first two digits or single alpha character of the data represents an Application Indicator, assigned by AIM International, followed by an implied FNC1 character. No special action is required on the decoder's part to conform with this rule (beyond setting the prefix to]C2 or]L4), as the Application Indicator is transmitted exactly as encoded.

- Codewords 910 and 911 cause the decoder's symbology identifier to be set to]C0 or]L5. No other special action is required on the decoder's part to conform with this rule as the data is transmitted exactly as encoded.

- Codeword 912 is similar to Codeword 907, indicating a "linked" UCC/EAN 128 emulation with an implied start in Numeric Compaction Mode. However, Codeword 912 also indicates that the data begins with a 6-digit date field (AI's 11, 13, 15, or 17), and this date field may be followed by an implied AI 10 or 21 as well. The first 6 digits of encoded data are defined as follows:

- The first 2 digits YY represent a year (range 00..99)
- The next two digits xx represent the month, and the date type, as follows:
  - (xx mod 13) represents the month MM (range 01..12).

NOTE xx values of 00, 13, 26, and 39 are unused.

  - (xx div 13) represents D, the type of date, as follows:
    - if D is 0: Implied AI is "17" (Expiration date)
    - if D is 1: Implied AI is "15" (Sell-by date)
    - if D is 2: Implied AI is "13" (Packaging date)
    - if D is 3: Implied AI is "11" (Production date)
- The last two digits zz represent the day, and the following AI, as:
  - (zz mod 32) represents the day DD (range 00..31, where 00 means "day not specified")
  - (zz div 32) represents *A*, the presence or absence of a next implied AI:
    - if *A* is 0: no Implied AI follows the date (either the data string ends after the date, or an explicit AI follows the date)
    - if *A* is 1: an Implied AI of "10" (Lot number) follows the date
    - if *A* is 2: an Implied AI of "21" (Serial number) follows the date

- Codewords 914 and 915 are similar to Codeword 907, indicating a "linked" UCC/EAN 128 emulation with an implied start in Numeric Compaction Mode. However, Codeword 914 also indicates that the data begins with an implied AI 10 (Lot number); similarly, Codeword 915 indicates that the data begins with an implied AI 21 (Serial number).

Refer to the GS1 specifications for further information about the data content of UCC/EAN 128 symbols.

5.4.1.6 Function codewords for linkage flags in composite symbols

Codeword 920 shall be used as a linkage flag to signal the presence of an associated linear component in accordance with ISO/IEC 24723.

Codeword 918 shall be used as a linkage flag to signal the presence of an associated linear component in any other composite symbology.

When used, the 918 or 920 codeword may appear in any position in the symbol. The applicable composite symbology specification may define a specific position of the linkage flag.

Readers supporting the indicated composite application should decode and transmit the data from all components as specified in the relevant composite symbology specification. Readers not supporting the indicated composite application may treat the 918 or 920 codeword as a reserved codeword (see 5.4.6). In addition, readers not supporting the indicated 918 composite application may have an option to ignore the 2D component and transmit only the data from the associated linear component.

5.4.1.7 Macro Characters

MicroPDF417 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. Their functions are defined as follows:

- if the first symbol character is 916 (i.e. encoding Macro character 05), then the preamble $[)>{}^{R}_{S}05{}^{G}_{S}$ shall precede the encoded data that follows it. The data begins with an implied latch to Numeric Compaction mode.

- if the first symbol character is 917 (i.e. encoding Macro character 06), then the preamble $[)>{}^{R}_{S}06{}^{G}_{S}$ shall precede the encoded data that follows it. The data begins with an implied latch to the mixed sub-mode of Text Compaction mode.

- the postamble ${}^{R}_{S}{}^{E}o_{T}$ shall be transmitted after the data in both cases.

The header shall be transmitted as a prefix to the data stream and the trailer shall be transmitted as a suffix to the data stream. Within the encoded data, codewords 903 through 905 may be used as field separators transmitted as ASCII ${}^{G}_{S}$. The symbology identifier, if used, shall precede the header.

The Macro characters shall not be used in conjunction with a series of Structured Append symbols.

5.4.2 Text Compaction mode

The Text Compaction mode includes all the printable ASCII characters (i.e. values from 32 to 126) and three ASCII control characters: HT or tab (ASCII value 9), LF or line feed (ASCII value 10), and CR or carriage return (ASCII value 13). The Text Compaction mode also includes various latch and shift characters which are used exclusively within the mode.

The Text Compaction mode encodes up to 2 characters per codeword. The compaction rules for converting data into PDF417 codewords are defined in 5.4.2.2. The sub-mode switches are defined in 5.4.2.3.

5.4.2.1 Text Compaction sub-modes

The Text Compaction mode has four sub-modes:

- Alpha (uppercase alphabetic)
- Lower (lowercase alphabetic)
- Mixed (numeric and some punctuation)
- Punctuation

Each sub-mode contains 30 characters, including sub-mode latch and shift characters.

A latch codeword from another mode to the Text Compaction mode shall always switch to the Text Compaction Alpha sub-mode.

All the characters and their values are defined in Table 6.

Table 6 — Text Compaction sub-mode definition

Base 30 value	Text Compaction sub-modes							
	Alpha		Lower		Mixed		Punctuation	
	Char	ASCII	Char	ASCII	Char	ASCII	Char	ASCII
0	A	65	a	97	0	48	;	59
1	B	66	b	98	1	49	<	60
2	C	67	c	99	2	50	>	62
3	D	68	d	100	3	51	@	64
4	E	69	e	101	4	52	[	91
5	F	70	f	102	5	53	\	92
6	G	71	g	103	6	54	]	93
7	H	72	h	104	7	55	_	95
8	I	73	i	105	8	56	`	96
9	J	74	j	106	9	57	~	126
10	K	75	k	107	&	38	!	33
11	L	76	l	108	CR	13	CR	13
12	M	77	m	109	HT	9	HT	9
13	N	78	n	110	,	44	,	44
14	O	79	o	111	:	58	:	58
15	P	80	p	112	#	35	LF	10
16	Q	81	q	113	-	45	-	45
17	R	82	r	114	.	46	.	46
18	S	83	s	115	$	36	$	36
19	T	84	t	116	/	47	/	47
20	U	85	u	117	+	43	"	34
21	V	86	v	118	%	37	|	124
22	W	87	w	119	*	42	*	42
23	X	88	x	120	=	61	(	40
24	Y	89	y	121	^	94	)	41
25	Z	90	z	122	pl		?	63
26	space	32	space	32	space	32	{	123
27	ll		as		ll		}	125
28	ml		ml		al		'	39
29	ps		ps		ps		al	

al	=	latch to alpha	ml	=	latch to mixed
as	=	shift to alpha	pl	=	latch to punctuation
ll	=	latch to lower	ps	=	shift to punctuation

NOTE The 'Char' columns above show the default interpretation in ECI 000003 of the byte values shown in the adjacent 'ASCII' columns. Each table entry represents half a codeword, i.e. the value range from 0 to 29 (see 5.4.2.2)

5.4.2.2 Compaction rules for encoding in Text Compaction mode

In Text Compaction mode, pairs of data characters are represented in a single codeword. The values assigned to the data characters are in the range 0 to 29 (i.e. base 30) and are defined in Table 6. Annex G defines the algorithm and illustrates a worked example.

5.4.2.3 Text Compaction sub-mode switching : latch and shift function

Switching from one sub-mode to another within Text Compaction mode shall be through the latch and shift values defined for the sub-mode in effect prior to the switch.

A **sub-mode shift** shall be used to switch from one Text Compaction sub-mode to another for only one data character. Subsequent codewords revert to the sub-mode being used immediately prior to the shift (except when **ps** is used as a pad, see 5.4.2.4). The shift functions are as follows:

ps	=	shift to punctuation sub-mode
as	=	shift to uppercase alphabetic sub-mode

A **sub-mode latch** shall be used to switch from one Text Compaction sub-mode to another, which stays in effect until another latch or shift is explicitly brought into use. The latch functions are as follows:

al	=	latch to uppercase alphabetic sub-mode
ll	=	latch to lowercase alphabetic sub-mode
ml	=	latch to mixed (numeric and other punctuation) sub-mode
pl	=	latch to punctuation sub-mode

A limited set of latch and shift functions is available within each Text Compaction sub-mode. Those which are available are listed in Table 6. Table 7 shows the transition table between Text Compaction sub-modes; Figure 5 shows this schematically.

NOTE: A sub-mode latch may be followed by another sub-mode latch or sub-mode shift; but a sub-mode shift may not be followed by either a sub-mode shift or sub-mode latch.

Table 7 — Text Compaction sub-mode transition table

Original sub-mode	**Destination sub-mode**			
	Alpha	**Lower**	**Mixed**	**Punctuation**
Alpha		ll	ml	ps
Lower	as		ml	ps
Mixed	al	ll		ps pl
Punctuation	al			

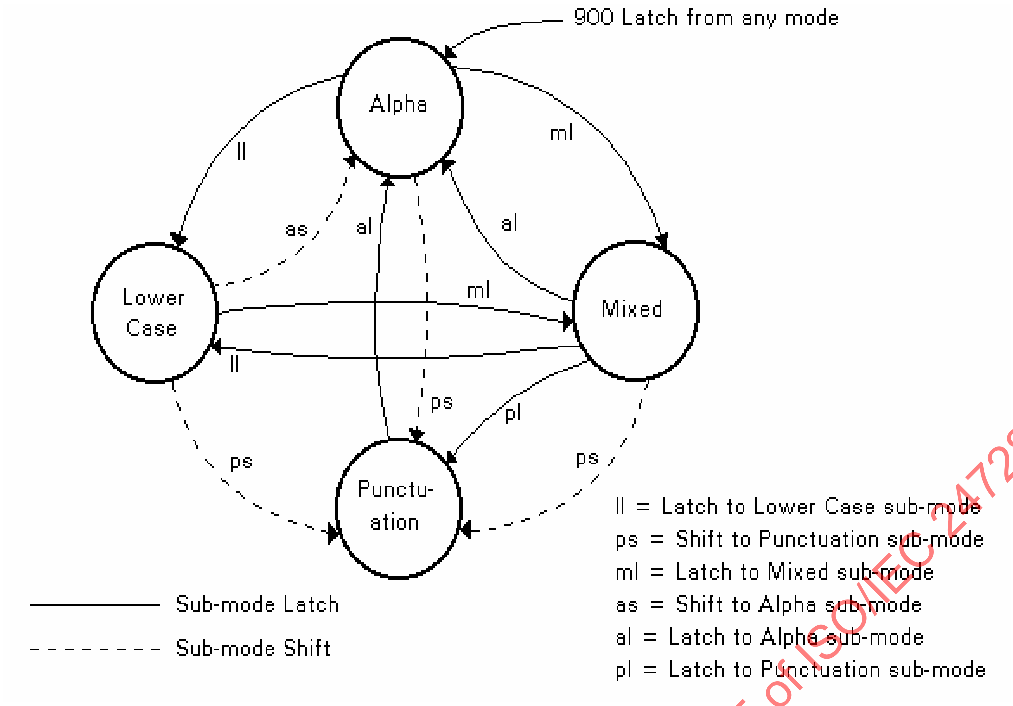

900 Latch from any mode
Alpha
Lower Case
Mixed
Punctu-ation
ll
as
al
ml
al
ml
ll
ps
pl
ps
ps
Sub-mode Latch
Sub-mode Shift
ll = Latch to Lower Case sub-mode
ps = Shift to Punctuation sub-mode
ml = Latch to Mixed sub-mode
as = Shift to Alpha sub-mode
al = Latch to Alpha sub-mode
pl = Latch to Punctuation sub-mode

Figure 5 — Text Compaction sub-mode switching

5.4.2.4 Mechanisms for using a pad in Text Compaction mode

If the Text Compaction character sequence does not result in an even number of base 30 values, a pad shall be added to the end of the character sequence. An example is illustrated in Table G.1. As there are no specific null functions in Text Compaction mode, the sub-mode shift and latch shall be used in accordance with the mechanisms defined for the following cases.

The cases are as follows:

a) If the character sequence continues to the end of the data, or the Text Compaction mode character sequence is followed by latching to another compaction mode, then the pad can be any of the sub-mode shifts or sub-mode latches.

b) If the Text Compaction mode character sequence is followed by a byte shift (codeword 913) to encode a single Byte Compaction mode character, two mechanisms can be used depending on the Text Compaction sub-mode being used prior to the Byte Compaction shift:

   1) If the Text Compaction sub-mode is other than punctuation, then base 30 value 29 (**ps**) should be used if encodation is intended to revert to the same Text Compaction sub-mode. The decoder shall ignore a **ps** immediately preceding codeword 913. If encodation is intended to change to a different sub-mode after the Byte Compaction mode character, the appropriate sub-mode latch may be used as the last base 30 value before the codeword 913.

   2) If the Text Compaction sub-mode is punctuation, then base 30 value 29 (**al**) shall be used. The decoder shall not ignore the **al** and shall therefore return to the alpha sub-mode.

5.4.2.5 Switching from Text Compaction mode

Text Compaction mode may be terminated by the end of the symbol, or by any of the following codewords:

- 900 (Text Compaction mode latch)
- 901 (Byte Compaction mode latch)

- 902 (Numeric Compaction mode latch)
- 924 (Byte Compaction mode latch)
- 928 (Beginning of Structured Append Control Block)
- 923 (Beginning of Structured Append Optional Field)
- 922 (Structured Append Terminator)

The last three codewords only occur within the Structured Append Control Block of a Structured Append symbol (see 5.13.1). Text Compaction mode is also affected by the presence of a reserved codeword (see 5.4.6).

If the decoder is in the Text Compaction mode and encounters codeword 913 (Byte Compaction mode shift), it decodes the codeword following codeword 913 as a single binary byte and then returns to the Text Compaction mode. The sub-mode to which the decoder returns is the most-recently-latched sub-mode that was in effect prior to codeword 913; a **ps** sub-mode shift immediately prior to codeword 913 is ignored.

If the decoder is in the Text Compaction mode and encounters codeword 900 (Text Compaction mode latch), the decoder reinitialises to the Alpha sub-mode.

5.4.3 Byte Compaction mode

The Byte Compaction mode enables a sequence of 8-bit bytes to be encoded into a sequence of codewords. It is accomplished by a Base 256 to Base 900 conversion, which achieves a compaction ratio of six bytes to five codewords (1,2 : 1).

All the characters and their values (0 to 255) are defined in Annex B. This shall be treated as the default graphical and control character interpretation. When ECIs are invoked (see 5.5) this interpretation is defined as ECI 000003 (see 5.5.2).

NOTE In AIM ITS/98-001, superseded by this International Standard, the default character set corresponded to ECI 000002 (a code page of the MS-DOS operating system). The interpretation of byte character values below 128 is unchanged, and the operation of MicroPDF417 printing and scanning equipment is unaffected. New applications that use byte character values above 127 should assume the ECI 000003 default interpretation for broadest compatibility with current systems. Existing applications utilizing values above 127 may continue to encode and process data as before. Applications that rely upon the prior default interpretation of values above 127, may encode ECI 000002 explicitly if they wish to signal this interpretation.

The default compaction mode for MicroPDF417 in effect at the start of each symbol shall always be Byte Compaction mode, introduced by an implied mode latch codeword 901 (see 5.4.3.3) or an explicit codeword 901 or 924 if data codewords appear before any ECI is invoked in the symbol.

If Text or Numeric Compaction mode would result in a smaller symbol, then explicit latches to these modes may be used. In such situations the appropriate Mode Latch may be encoded in the first or subsequent codeword position (if the ECI Descriptor codeword is not used), or in the second or subsequent codeword position (if the ECI Descriptor codeword is used).

5.4.3.1 Switching to Byte Compaction mode

When in either Text or Numeric Compaction mode, to switch to Byte Compaction mode, it is necessary to use one of the following codewords:

- **mode latch 924** shall be used when the total number of bytes to be encoded is an integer multiple of 6
- **mode latch 901** shall be used when the total number of bytes to be encoded is not a multiple of 6
- **mode shift 913** can be used instead of codeword 901 when in Text Compaction mode and a single Byte Compaction character has to be encoded

5.4.3.2 Compaction rules for encoding a single Byte Compaction character (using mode shift 913)

To encode a single Byte Compaction character, the codeword shall be the decimal value (0 to 255) of the character as defined in Annex B.

5.4.3.3 Compaction rules for encoding longer Byte Compaction character strings (using mode latch 924 or 901)

The following procedure shall be used to encode Byte Compaction character data:

a) Establish the total number of bytes.

b) If not a perfect multiple of 6, mode latch 901 shall be used (either implicitly at the beginning of the symbol if in the default mode and an ECI is invoked before any data codewords, or explicitly); if a perfect multiple of 6, mode latch 924 shall be used.

c) Sub-divide the number of bytes into a sequence of 6 characters, from left to right (the most to least significant characters). If less than 6 characters go to Step 7.

d) Assign the decimal values of the 6 data bytes to be encoded in Byte Compaction mode as b_5 to b_0 (where b_5 is the first data byte).

e) Carry out a base 256 to base 900 conversion to produce a sequence of 5 codewords. Annex C defines an algorithm and illustrates a worked example.

f) Repeat from Step 3 as necessary.

g) For the remaining bytes when mode latch 901 is used, (i.e. when the last group is less than 6 bytes) the codeword(s) shall be the decimal value(s) (0 to 255) of the character(s) as defined in Annex B, the most to the least significant.

NOTE: Byte Compaction mode following mode latch 901 assumes that the total number of bytes to be encoded is not a multiple of six. If the number of bytes to be encoded in Byte Compaction mode happens to be an integer multiple of six, then either a 901 or a 924 Byte Compaction Latch shall be encoded, placed at any point in the symbol that would create a correct encodation according to these encodation rules. For example, a 924 codeword as either the first or second codeword would identify the following stream of Byte Compaction mode codewords as encoding a multiple-of-six number of bytes. Alternatively, a 901 could be placed at any position within the Byte Compaction mode codeword stream that would split that stream into two segments, neither of which encodes a multiple-of-six number of bytes.

If additional encodation is required in Text Compaction or Numeric Compaction modes, the appropriate latch characters shall be used (see 5.4.1.1).

5.4.3.4 Switching from Byte Compaction mode

Byte Compaction mode may be terminated by the end of the symbol, or by any of the following codewords:

- 900 (Text Compaction mode latch)
- 901 (Byte Compaction mode latch)
- 902 (Numeric Compaction mode latch)
- 924 (Byte Compaction mode latch)
- 928 (Beginning of Structured Append Control Block)
- 923 (Beginning of Structured Append Optional Field)
- 922 (Structured Append Terminator)

The last three codewords only occur within the Structured Append Control Block of a Structured Append symbol (see 5.13.1). Byte Compaction mode is also affected by the presence of a reserved codeword (see 5.4.6).

Re-invoking Byte Compaction mode (by using codeword 901 or 924 while in Byte Compaction mode) serves to terminate the previous Byte Compaction mode grouping of 6 Byte Compaction characters as described in 5.4.3.3, and then to start a new grouping. This procedure may be necessary when an ECI assignment number needs to be encoded (see 5.5.3.2).

During the decode process for Byte Compaction mode, the treatment of the final group of codewords differs depending on whether Byte Compaction mode is invoked with codeword 901 or 924.

If Byte Compaction mode is invoked with codeword 924, the total number of codewords within the compaction mode shall be a multiple of five. If this is not the case, the symbol is invalid. All the 5-codeword groups are decoded into 6-byte groups.

If Byte Compaction mode is invoked with codeword 901, the final group of codewords is interpreted directly as one byte per codeword, without compaction. Therefore, if the last group consists of five codewords, the group is interpreted as 5 bytes, rather than 6.

5.4.4 Numeric Compaction mode

The Numeric Compaction mode is a method for base 10 to base 900 data compaction and should be used to encode long strings of consecutive numeric digits. The Numeric Compaction mode encodes up to 2,93 numeric digits per codeword.

5.4.4.1 Latch to Numeric Compaction mode

Numeric Compaction mode may be invoked when in Text Compaction or Byte Compaction modes using mode latch 902.

5.4.4.2 Compaction rules for encoding long strings of consecutive numeric digits

The following procedure shall be used to compact numeric data:

1. Divide the string of digits into groups of 44 digits, except for the last group, which may contain fewer.

2. For each group add the digit 1 to the most significant position to prevent the loss of leading zeros.

EXAMPLE

original data	00246812345678	
after step 2	1	00246812345678

NOTE The leading digit 1 is removed in the decode algorithm.

3. Perform a base 10 to base 900 conversion. Annex D defines an algorithm for this and illustrates a worked example.

4. Repeat from Step 2 as necessary.

The following rules can be used to determine the precise number of codewords in Numeric Compaction mode:

- Groups of 44 numeric digits compact to 15 codewords.

- For groups of shorter sequences of digits, the number of codewords can be calculated as follows:

  Codewords = INT (number of digits / 3) + 1

EXAMPLE:

For a 28 digit sequence

INT (28 / 3) + 1

= 9 + 1

= 10 codewords

5.4.4.3 Switching from Numeric Compaction mode

Numeric Compaction mode may be terminated by the end of the symbol, or by any of the following codewords:

- 900 (Text Compaction mode latch)
- 901 (Byte Compaction mode latch)
- 902 (Numeric Compaction mode latch)
- 924 (Byte Compaction mode latch)
- 928 (Beginning of Structured Append Control Block)
- 923 (Beginning of Structured Append Optional Field)
- 922 (Structured Append Terminator)

The last three codewords only occur within the Structured Append Control Block of a Structured Append symbol (see 5.13.1). Numeric Compaction mode is also affected by the presence of a reserved codeword (see 5.4.6).

Re-invoking Numeric Compaction mode (by using codeword 902 while in Numeric Compaction mode) serves to terminate the current Numeric Compaction mode grouping as described in 5.4.4.2, and then to start a new grouping. This procedure may be necessary when an ECI assignment number needs to be encoded (see 5.5.3.4).

During the decode process for Numeric Compaction mode, the result of the base 900 to base 10 conversion shall result in a number whose most significant digit is a ‘1'. If the base 900 to base 10 conversion does not result in a number beginning with ‘1', the symbol shall be treated as invalid. The leading ‘1' is removed to produce the original number.

5.4.5 Advice to select the appropriate compaction mode

All basic implementations for printing and scanning MicroPDF417 symbols shall support the three modes: Text Compaction, Byte Compaction and Numeric Compaction. The default character set for Text Compaction shall be as defined in Table 6; and that for Byte Compaction shall be as defined in Annex B. Text Compaction mode is usually more efficient than Byte Compaction mode for encoding standard ASCII text files because of its better compaction of ASCII character values 9, 10, 13 and 32 to 126.

The Numeric Compaction mode should be used for long numeric strings.

Advice about switching between modes to minimise the number of codewords is provided as an algorithm in Annex N.

5.4.6 Treatment of MicroPDF417 reserved codewords

5.4.6.1 Overview

MicroPDF417 symbols intended for use in open systems should not employ any of the codewords that are listed as reserved (see 5.4.1) in the current edition of this standard. However, decoding equipment should support the transmission of reserved codewords using escape sequences as defined in 5.17.4. Decoding equipment may also support an option of treating such symbols as invalid, as would be the case when operating in Basic Channel Mode.

Receiving systems should discard data containing any escape sequences using reserved codewords, unless the system is aware of a new definition for a previously reserved codeword.

5.4.6.2 Making future use of reserved codewords

Currently, function codeword 919 is reserved. Any new function codewords, to be defined in future revisions of this standard, shall have their encoding rules specified to provide backwards compatibility with pre-existing equipment. If a new function codeword introduces a new compaction mode, then the interpretation of the subsequent codewords has a new meaning unknown to pre-existing equipment, and thus the subsequent codewords all must be transmitted as escape sequences, until a known compaction mode codeword is reached. Alternatively, if a new function codeword defines a "signalling" function (such as to alter the transmitted Symbology Identifier option value) with no effect on the interpretation of subsequent codewords, then only the new codeword itself would need be transmitted as an escape sequence. Specifically:

- When a new signalling codeword is encoded, it shall immediately be followed by an appropriate compaction mode latch so that the subsequent data codewords are interpreted and transmitted as a byte stream, rather than as a series of escaped uninterpreted codewords. This approach will achieve the desired results with decoding equipment conforming with any revision of this MicroPDF417 standard, regardless of whether that equipment employs the original or the new transmission protocol.

- At the receiving system, the ECI decoder will process the control ECIs (i.e. Structured Append Control Blocks and escaped uninterpreted codewords) before the encodable ECIs (such as encryption schemes and character sets). Thus, the encoder should take into account the order of operations as follows:

    1. The Structured Append Control Block ECIs, if present, will be used to assemble the complete byte stream in the proper order.

    2. The escaped data codewords will be translated by the ECI decoder according to the rules of the new compaction mode or signal codeword, and the resulting data bytes will be inserted into their proper place within the byte stream.

    3. Finally, the character set and other encodable ECIs will be applied to the resulting byte stream.

5.5 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, including MicroPDF417. ECIs are assigned by AIM Global, Inc.

NOTE Originally a symbology specific scheme called Global Label Identifiers (GLIs) was defined for PDF417. Encoding and decoding of ECIs is identical to early specifications for PDF417 GLIs. However, the transmission protocol for decoded messages according to early PDF417 specifications for GLIs is different from the transmission protocol for ECIs. There are also differences with respect to the use of interpretative ECIs with Structured Append. This standard permits the use of the earlier and current protocols in such a way that old and new equipment can continue to co-exist.

Five broad types of interpretations are supported in MicroPDF417:

a) character sets (or code pages)

b) general-purpose interpretations such as data encryption and data compression (as distinct from the compaction modes of the symbology)

c) user defined interpretations for closed systems

d) transmission of control information for Structured Append

e) transmission of uninterpreted PDF417 codewords

Transmission of the Extended Channel Interpretation protocol is fully specified in AIM ITS/04-001, Part 1. The protocol provides a consistent method to specify particular interpretations of byte values before printing and after decoding.

The Extended Channel Interpretation (ECI) is identified by a 6-digit number which is encoded in the MicroPDF417 symbol by one of three specific codewords followed by one or two codewords (see 5.5.1). A specific ECI may be invoked anywhere in the encoded message subject to the rules of the compaction modes (see 5.5.3).

The ECI protocol can only be used with decoders enabled to transmit the symbology identifier (see 5.17.5). Decoders that are not enabled to transmit the symbology identifier cannot reliably convey the escape sequences from any symbol containing an ECI.

5.5.1 Encoding the ECI assignment number

An ECI can be invoked anywhere in the data stream, subject to the conditions defined in 5.5.3. Once an ECI has been invoked, switching may take place between any of the compaction modes. The compaction mode used is determined strictly by the 8-bit data values being encoded and does not depend on the ECI in force. For example, a sequence of values in the range 48 to 57 (decimal) would be most efficiently encoded in Numeric Compaction mode even if the sequence is not to be interpreted as numbers.

The ECI assignment number is encoded in one of three ECI codeword sequences, which begin with the codewords 927, 926 or 925. One or two additional codewords are used to encode the ECI assignment number. The encodation rules are defined in Table 8. In addition, MicroPDF417 supports the application of the rule in 5.2.4.2 (ECI Descriptor codeword).

Table 8 — Encoding ECI assignment numbers

ECI assignment number	Codeword sequence	Codewords	Ranges
000000 to 000899	C_0	927	
	C_1	ECI_no	C_1=(0 to 899)
000900 to 810899	C_0	926	
	C_1	ECI_no div 900 - 1	C_1=(0 to 899)
	C_2	ECI_no mod 900	C_2=(0 to 899)
810900 to 811799	C_0	925	
	C_1	ECI_no - 810900	C_1=(0 to 899)

There are 811 800 possible ECI assignment numbers available in MicroPDF417.

NOTE The encodation method is identical to the GLI scheme supported in the PDF417 sponsor's original specification and incorporated in the AIM USA (1994) and AIM Europe (1994) specifications.

EXAMPLE The following example illustrates the encodation:

ECI = 013579

Codewords: [926] [(13 579 div 900) - 1] [13 579 mod 900]

= [926] [15 - 1] [79]

= [926] [14] [79]

5.5.2 Pre-assigned and default Extended Channel Interpretations

The following ECIs, ECI 000000 to ECI 000003, have been pre-assigned to be backwards compatible with existing symbology specifications, including PDF417.

— ECI 000000 (equates to original GLI 0) and represents the default encodation scheme of encoders compliant with the original AIM USA (1994) and AIM Europe (1994) PDF417 specifications.

— ECI 000001 (equates to original GLI 1) represents the GLI encodation scheme of a number of symbologies with characters 0 to 127 being identical to those of ISO/IEC 646:1991, International Reference Version (equivalent to ANSI X3.4) and characters 128 to 255 being identical to those values of ISO 8859-1.

NOTE ECI 000000 (equivalent to GLI 0) and ECI 000001 (equivalent to GLI 1) require a reset-to-GLI 0 logic at the beginning of each encoded symbol of a Structured Append set of symbols (see Annex H.5.1). This protocol is not adopted for other Extended Channel Interpretations.

— ECI 000002 has an equivalent code table to ECI 000000, without the reset-to-GLI 0 logic.

— ECI 000003 has an equivalent code table to ECI 000001, without the return-to-GLI 0 logic. ECI 000003 is the default encodation scheme for encoders fully compliant with this edition of this standard.

ECI 000000 and ECI 000001 shall not be encoded in the same MicroPDF417 symbol or Structured Append symbol set as other ECIs, except for user defined ECIs. ECI 000002 and ECI 000003 provide the compatible alternatives to ECI 000000 and ECI 000001 respectively. ECI 000000 and ECI 000001 should not be used in new applications.

5.5.3 Encoding ECI sequences within compaction modes

The general encodation principle is that ECIs are applied to the source data byte stream (to signal various interpretations) producing a modified byte stream that is encoded into MicroPDF417 symbols using the symbology's compaction modes for efficiency. The ECI encoding, and symbology specific compaction, form two independent logical layers of the process.

Although ECI assignments and compaction modes may generally be intermixed, some combinations can produce illogical or ambiguous behaviour. The following sections define how ECIs may be incorporated without ambiguity by specifying the valid placements of ECI escape sequences.

5.5.3.1 ECIs and Text Compaction mode

An ECI escape sequence may be placed anywhere within Text Compaction mode. The sub-mode invoked immediately prior to the ECI escape sequence is preserved for the encodation immediately after it. Thus, sub-mode latches and shifts are preserved across an ECI escape sequence; and thus a sub-mode shift immediately before an ECI escape sequence is not ignored.

5.5.3.2 ECIs and Byte Compaction mode using mode latch 924 and 901

If encoding in Byte Compaction mode using mode latch 924, an ECI escape sequence may be positioned by an encoder immediately following codeword 924, or at any 5-codeword boundary thereafter. This is necessary to provide an unambiguous position in the decoded byte stream for the decoder to place the escape sequence.

If the decoder is in the 924 version of Byte Compaction mode and finds an ECI escape sequence following a 5-codeword group, it shall output the six data bytes associated with the codewords before the escape sequence, output the escape sequence, and then continue collecting codewords for decoding in Byte Compaction mode. If the decoder encounters an ECI escape sequence at other than these prescribed locations, it shall treat the symbol as invalid.

If encoding in Byte Compaction mode using mode latch 901, an ECI escape sequence may be positioned:

- Immediately following codeword 901 (but see also 5.2.4.1)
- Immediately after any set of five codewords encoding six bytes
- Immediately after any of the trailing single-byte codewords at the end of the sequence

NOTE The decoder cannot assume that, just because the ECI escape sequence follows a set of five codewords, the five codewords encode six bytes, since an input stream of length 6N + 5 (where N is an integer) will have a final set of five codewords that encode only five bytes, one byte per codeword. The decoder must, therefore, look ahead in the symbol past the ECI escape sequence(s) to determine whether the 901 mode terminates, as defined in 5.4.3.4. Based on this information, it can then determine how the group of five codewords has been encoded.

Figure 6 illustrates valid locations for ECI escape sequences when encoding in Byte Compaction mode. If the decoder encounters an ECI escape sequence within the 5-codeword group, it shall treat the symbol as invalid.

[901]♦□ □ □ □ □♦□ □ □ □ □♦□♦□♦□♦□♦□♦

[924]♦□ □ □ □ □♦□ □ □ □ □♦

5 codeword group 5 codeword group

where: □	=	Byte Compaction mode codeword
♦	=	Valid location for ECI escape sequence

Figure 6 — Valid locations for ECI escape sequences in Byte Compaction mode

5.5.3.3 ECIs and Byte Compaction using mode shift 913

If encoding in Byte Compaction mode using mode shift 913, an ECI escape sequence may be placed:

- Immediately preceding codeword 913
- Immediately following codeword 913
- Immediately following the codeword after codeword 913

In the first two cases, the ECI escape sequence is output before the encoded byte, while in the last case, the escape sequence is output following the encoded byte.

5.5.3.4 ECIs and Numeric Compaction mode

An ECI escape sequence shall not be placed within a group of codewords being processed through the base 10 to base 900 conversion as defined in 5.4.4.2. It may only be placed within a Numeric Compaction mode region at a boundary between (the typically) 15-codeword groups. This is necessary to provide an unambiguous position in the decoded digit stream for the decoder to place the escape sequence.

Thus, an ECI escape sequence may only be placed:

- Immediately after codeword 902
- After the 15th codeword
- After the 30th codeword
- etc.

If the encoder needs to place an ECI escape sequence at a location that does not result in a multiple of 15 codewords, it shall treat the numeric block before the ECI as a complete entity, as defined in 5.4.4.2 step 2. It shall re-invoke the Numeric Compaction mode by placing another codeword 902 in the stream followed by the ECI escape sequence.

If the decoder finds an ECI escape sequence on one of the boundary points defined above, it shall emit the data bytes associated with the codewords before the escape sequence (if any), then emit the escape sequence, and then continue collecting codewords for decoding in Numeric Compaction mode. If the decoder encounters an ECI escape sequence at other than the prescribed locations, it shall treat the symbol as invalid.

5.5.3.5 Combining ECIs

Two or more ECI escape sequences (e.g. assignment numbers) may be placed at any point where one ECI can be validly located; provided that no codewords, other than those used to encode the ECI escape sequence, are placed between them.

5.5.4 Post-decode protocol

The protocol for transmitting ECI data shall be as defined in 5.17.2. When transmitting ECIs, symbology identifiers (see 5.17.5) shall be fully implemented and the appropriate symbology identifier shall be transmitted as a preamble.

5.6 Determining the codeword sequence

The encoding process generates a sequence of codewords defined as:

$$d_{n-1}, d_{n-2}, \ldots\ d_0$$

where: d = data codeword, including ECI Descriptor (if used) and all function codewords

n = total number of data codewords, excluding the error correction codewords

During the encoding process, sequences of codewords are established. Like the original data itself, the most significant codewords shall appear first, for example textual and numeric data reads from the left to the right. The sequence of codewords shall be such that the most significant data codeword as defined above is the one designated d_{n-1} (the ECI Descriptor codeword position). The final data codeword is the one designated d_0.

The process used to determine the symbol matrix of rows and columns can require the addition of trailing pad codewords to the end of the data codeword sequence.

5.7 Error detection and correction

Each MicroPDF417 symbol contains at least seven error correction codewords. The Error Correction codewords provide capability for both error detection and correction.

5.7.1 Number of error correction codewords

The number of error correction codewords for a MicroPDF417 symbol is fixed for each symbol version. Table 1 shows the number of error correction codewords for each symbol version.

5.7.2 Error correction capacity

Error correction can be used to compensate for defects in the label and misreads during the decode procedure. A particular number of error correction codewords is incorporated into any given MicroPDF417 symbol. The error correction codeword algorithm used allows two types of error to be recovered:

- an **erasure**, which is a missing or undecodable codeword at a known position;
- a **substitution error**, which is an erroneously decoded codeword at an unknown position.

The error correction scheme requires one error correction codeword to rectify an erasure and two to recover a substitution error. Thus a given number of error correction codewords can rectify any combination of substitution errors and erasures which satisfy the following equations:

$$L + 2f \le k - 2$$

where: L, f and k are as defined in 4.1However, if most of the error correction capacity is used to correct erasures, the possibility of undetected errors is increased. For this reason, whenever there are fewer than 4 errors corrected, the error correction capacity should be reduced as follows:

$$L + 2f \le k - 3$$

where: L, f and k are as defined in 4.1

EXAMPLE A 3×10 MicroPDF417 symbol has 16 error detection and correction codewords, up to 14 of which can be used to correct errors and erasures. They can correct up to 13 erasures or 7 substitution errors, or any combination of L erasures and f substitution errors subject to the practical equations above. Table 9 specifies the possible combinations.

Table 9 — Possible error correction combinations for 16 error correction codewords

Recovered Substitution Errors	Recovered Erasures	Determining Equation
0	13 or less	$L + 2f \le k - 3$ (number of errors f is <4)
1	11 or less	
2	9 or less	
3	7 or less	
4	6 or less	$L + 2f \le k - 2$ (number of errors $f \ge 4$)
5	4 or less	
6	2 or less	
7	0	

5.7.3 Defining the error correction codewords

A two-stage process must be performed to define the error correction codewords:

1. Determining the number of error correction codewords. This depends on the symbol version and is indicated in Table 1.

2. Generating the error correction codewords. This is to a prescribed set of rules defined in 5.10. The procedures cannot be used until all the data codewords, including pad codewords *(*see 5.9.2*)* have been defined.

NOTE The procedures defined in 5.3 to 5.9, 5.13 and 5.14 are of prime interest to users. The more technical procedures defined in 5.10, 5.11 and 5.15 are likely to be achieved electronically and require no user decisions.

5.8 Dimensions

MicroPDF417 symbols should conform with the following dimensions:

5.8.1 Minimum width of a module (*X*)

This should be defined by the application specification, having due regard to the availability of equipment for the production and reading of symbols and complying with the general requirements of the application.

The *X* dimension shall be constant throughout a given symbol.

NOTE Current bar code symbol quality measurement standards (e.g. ISO/IEC 15415) do not require absolute dimensional measurements to be taken into account for assessing symbol quality. Non-compliance with any minimum dimension should not therefore, by itself, be a reason for rejection of a symbol under these standards.

5.8.2 Row height (*Y*)

The row height (Y) shall conform to:

$$Y \geq 2X$$

5.8.3 Quiet zones

Minimum width of horizontal quiet zone (to the left and right of the symbol): 1*X*

Minimum width of vertical quiet zone (above and below the symbol): 1*X*

5.9 Defining the symbol format

The MicroPDF417 symbol matrix and the overall size and shape of the symbol are determined by:

1. The module width and aspect ratio.

2. The number of rows and columns in the symbol matrix (in accordance with the set of available MicroPDF417 versions).

To create a MicroPDF417 symbol, these parameters are selected through a combination of user inputs, application constraints and default settings. The selection process can be iterative until the user is satisfied with the resultant format.

5.9.1 Defining the aspect ratio of the module

The aspect ratio of the printed module shall be defined by two dimensions:

X	the nominal dimension of the narrowest bar and narrowest space
Y	the nominal dimension of the height of each row

These parameters are defined by the user or application. The major factors that determine the values of these parameters are the resolutions of the printing and scanning systems used in the application.

These points are discussed in 5.14.

5.9.2 Defining the symbol matrix of rows and columns

There are several factors which need to be considered in order to determine the symbol matrix, i.e. the number of rows ***r*** and columns ***c***:

- the amount and type of data to be encoded
- the basic rules of the symbology which, for example, determine the available combinations of the numbers of rows and columns (see 5.2.1 and 5.2.2).
- the physical space available to print the symbol
- the fact that longer rows result in the use of less symbol overhead (Row Address Patterns and space for quiet zones)
- the fact that the length of the row (including the quiet zones) must be less than the length of the scan line prescribed or implied by the application
- the type of scanner, which may determine the overall aspect ratio of the symbol
- the fact that a larger symbol than the minimum for the amount of data may allow a higher number of error detection and correction codewords for increased security.

Annex O provides more precise guidelines which should be used to define the symbol matrix.

After the source data has been encoded using the appropriate compaction mode(s), the number of source data codewords *m* (including the codeword in the ECI Descriptor position but prior to the addition of any pad codewords) is known. Once the numbers of rows and columns have been selected, the total number of data codewords *n* is calculated as:

$$n = c \times r - k$$

where: *c*, *k*, *n* and *r* are as defined in 4.

The matrix can result in a situation where the number of rows and columns requires the use of pad codewords (by using any valid series of mode latch codewords and/or Text compaction mode sub-latches). This occurs when:

$$n > m$$

where: *m* and *n* are as defined in 4

The number of pad codewords is:

$$n - m$$

The pad codewords shall be placed in the least significant positions of the data codeword sequence, i.e. to the right of the least significant source data codeword (but before the Structured Append Control Block, if present). An example of this process is given below. Apart from the insertion of any pad codewords, the codeword sequence shall remain identical to the one originally generated when encoding the source data.

EXAMPLE (assuming a data codeword sequence of 117 codewords in the default compaction mode, with no ECI invoked, and not containing a Structured Append Control Block, commencing with codeword 901 in the ECI Descriptor position, followed by 116 codewords representing source data 247 ... 423):

let c = 4

let r = 44

let m = 117

Then (from Table 1):

n = 126

k = 50

NOTE The notation is as defined above

The number of pad codewords = n - m = 126 - 117 = 9. In this example, these all have the value 900 and are positioned d_8 ... d_0. The addition of the ECI Descriptor and the pad codewords is shown below:

Original data codeword sequence	d_{m-1}	d_{m-2}	...	d_0			
Codewords	901	247	...	423			
Transformed data codeword sequence	d_{n-1}	d_{n-2}	...	d_9	d_8	...	d_0
Codewords	901	247	...	423	900	...	900

5.10 Generating the error correction codewords

The error correction codewords shall be generated using the procedure defined below. They are calculated on the basis of the values of all the data codewords including the ECI Descriptor and any pad codewords. The codeword sequence is defined as:

$\boldsymbol{d_{n-1}, d_{n-2} \ldots d_0}$

where: d_{n-1} is the codeword in the ECI Descriptor position

The symbol data polynomial is:

$$\boldsymbol{d(x) = d_{n-1}x^{n-1} + d_{n-2}x^{n-2} + \ldots + d_1x + d_0}$$

The following describes mathematically how the error correction codewords shall be computed for a given stream of data and a selected symbol version. All the arithmetic shall be done in modulo 929.

The error correction codewords are the complement of coefficients of the remainder resulting from dividing the symbol data polynomial $d(x)$ multiplied by x^k by the generator polynomial $g(x)$. Negative values are mapped in to the Galois Field GF (929) by adding 929 until the value ≥ 0.

The following generator polynomial shall be used to calculate coefficients for k error correction codewords required for the symbol version:

$$\begin{aligned} g_k(x) &= (x - 3)(x - 3^2)(x - 3^3) \ldots (x - 3^k) \\ &= \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \ldots \alpha_{k-1} x^{k-1} + x^k \end{aligned}$$

where: $g_k(x)$ = the generator polynomial and x is the unknown variable

k = total number of error correction codewords

α_j = coefficient of powers of x produced by the generator polynomial $g_k(x)$

An example for calculating the coefficients is given in Annex P.

All the coefficients necessary to encode a MicroPDF417 symbol of any version are shown in Annex F. As a convenience, the diskette accompanying this specification contains all the coefficient values.

The error correction codewords shall be calculated according to the algorithm defined below using the following notation:

d_i	=	data codeword $d_{n-1} \ldots d_0$
E_j	=	error correction codeword $E_{k-1} \ldots E_0$
α_j	=	coefficient of powers of x taken from the generator polynomial (see above for an explanation)
t_1, t_2, t_3	=	temporary variables

The algorithm is:

1. Identify the data codeword sequence.

   $d_{n-1}, d_{n-2} \ldots d_0$

2. Initialise error correction codewords $E_0, \ldots, E_{k-1}$ to value = 0

3. For each codeword $d_i = d_{n-1} \ldots d_0$

```
BEGIN
    $t_1 = (d_i + E_{k-1}) \bmod 929$
    For each error correction codeword $E_j = E_{k-1} \ldots E_1$,
    BEGIN
        $t_2 = (t_1 \times \alpha_j) \bmod 929$
        $t_3 = 929 - t_2$
        $E_j = (E_{j-1} + t_3) \bmod 929$
    END
    $t_2 = (t_1 \times \alpha_0) \bmod 929$
    $t_3 = 929 - t_2$
    $E_0 = t_3 \bmod 929$
END
```

4. For each error correction codeword $E_j = E_0 \dots E_{k-1}$, calculate the complement

```
BEGIN
    if $E_j$ not equal to 0
        $E_j = 929 - E_j$
END
```

An example of calculating the error correction codewords is given in Annex Q.

An alternative procedure for generating the error correction codewords, using a division circuit, is given in Annex R.

5.11 Low level encodation

Low level encoding converts the codewords into their corresponding symbol characters (bar-space sequences) given that the symbol version has been selected.

Figure 7 illustrates schematically for a four-column MicroPDF417 symbol the position of each data codeword and error correction codeword, along with the Row Address Patterns.

LRA_1	d_{n-1}	d_{n-2}	CRA_1	d_{n-3}	d_{n-4}	RRA_1
LRA_2	d_{n-5}		CRA_2			RRA_2
	d_3	d_2		d_1	d_0	
	e_{k-1}	e_{k-2}		e_{k-3}		
LRA_{r-1}			CRA_{r-1}			RRA_{r-1}
LRA_r		e_2	CRA_r	e_1	e_0	RRA_r

where:

LRA_j is the appropriate Left Row Address Pattern for row *j* of this symbol version

CRA_j is the appropriate Center Row Address Pattern for row *j* of this symbol version

RRA_j is the appropriate Right Row Address Pattern for row *j* of this symbol version

Shaded area = data codeword area

Unshaded area under the codeword area is for error correction codewords

Figure 7 — A MicroPDF417 symbol schematic showing the positioning of codewords

5.11.1 Clusters

MicroPDF417 uses the cluster-based system of local row discrimination used in PDF417 in order to detect row-to-row transitions. In MicroPDF417, row numbers are mapped to Row Address Pattern numbers (depending on symbol format), and then Row Address Pattern numbers are mapped to cluster numbers.

NOTE: In PDF417, there is a direct relationship between cluster numbers and row numbers.

The set of codewords is represented in each of three clusters. Cluster numbers 0, 3 and 6 are used. The associated bar-space sequences of the symbol characters representing each codeword in each cluster are given in Annex A.

To encode the codewords used for data and error correction characters, each row shall contain symbol characters (bar-space patterns) from only one cluster. In a MicroPDF417 symbol, a row that begins with Row Address Pattern 1 shall use symbol characters from cluster 0, a row that begins with Row Address Pattern 2 shall use symbol characters from cluster 3, a row that begins with Row Address Pattern 3 shall use symbol characters from cluster 6, a row that begins with Row Address Pattern 4 shall use symbol characters from cluster 0, and so on. The cluster sequence 0, 3, 6 repeats continually within any MicroPDF417 symbol, but unlike PDF417, this sequence may begin with either cluster 0, 3, or 6 in the first row of a MicroPDF417 symbol, depending upon which version is selected. The cluster number K for any row, given that row's Left Row Address Pattern number, can be calculated as:

$K = ((\textit{left Row Address Pattern number} - 1) \bmod 3) \times 3$

where the Left Row Address Patterns are numbered from 1 to 52 (as defined in 5.2.5).

Tables 10a to 10c show the correspondence between Row Address Pattern numbers, row numbers, and cluster numbers for all versions of MicroPDF417, and may be used to determine the cluster number of a symbol character adjacent to any Row Address Pattern.

As is the case with PDF417, because any two adjacent rows use different clusters, the decoder can utilize scans that cross rows while decoding a MicroPDF417 symbol.

5.11.2 Determining the symbol matrix

The symbol matrix of rows and columns shall be finally determined by the procedures set out in 5.9, taking into account the limited selection of MicroPDF417 versions. These procedures provide the values of r and c, which completely specify which version of MicroPDF417 will be printed.

5.11.3 Determining the values of the Row Address Patterns

The specific Row Address Patterns for each row of a given MicroPDF417 version are determined from Table 10 for the one-column versions, Table 11 for the two-column versions, or Table 12 for the three- and four-column versions. These tables group the MicroPDF417 versions into families with the same RAP rotation of their Row Address Patterns. The RAP rotation is always calculated between a pair of nearest-neighbour Row Address Patterns, and is defined as the right-hand pattern number of the pair minus the left-hand pattern number, or if the result is negative, then it is defined as the result plus 52. These RAP rotations are used to identify the different MicroPDF417 families shown in Tables 10, 11 and 12.

In order to determine the Row Address Pattern numbers to be used for a given version of MicroPDF417, find the version format ($c \times r$) column in the appropriate table 10, 11 or 12. The populated entries in that column represent the row numbers for that version. For each row, the leftmost column of the table defines the Left Row Address Pattern to be used, and the second column indicates the cluster number applicable to that row. The Center (if applicable) and Right Row Address Patterns are shown in the shaded leftmost column(s) within that version's family.

Two examples are given to demonstrate the use of the tables.

EXAMPLE 1 For the second smallest MicroPDF417 version (1 column by 14 rows), 10 shows that:

row 1 utilizes pattern number 8 for its Left Row Address Pattern, and utilizes pattern number 8 for its Right Row Address Pattern

row 14 of this version uses patterns 21 and 21 on the left and right, respectively;

Cluster 3 is used on the first row of this symbol, Cluster 6 on the second row, and so on.

Table 13 (an extract from Table 10) illustrates this with relevant values shown in bold type.

Table 14 shows the cluster number and the LRA and RRA for each row.

EXAMPLE 2 Based on Table 12, Table 15 shows the row structure of the MicroPDF417 version with 4 columns by 38 rows.

Table 10 — One Data Column MicroPDF417 Row Number Assignments

ALL Versions		Rotation-0 Family				Rotation-8 Family			
Left RAP	Cluster	Right RAP	Row Numbers for:			Right RAP	Row Numbers for:		
			VER. 1x14	VER. 1x17	VER. 1x20		VER. 1x11	VER. 1x24	VER. 1x28
1	0	1				9	1		
2	3	2				10	2		
3	6	3				11	3		
4	0	4				12	4		
5	3	5				13	5		
6	6	6				14	6		
7	0	7				15	7		
8	3	8	1			16	8		
9	6	9	2			17	9	1	
10	0	10	3			18	10	2	
11	3	11	4			19	11	3	
12	6	12	5			20		4	
13	0	13	6			21		5	
14	3	14	7			22		6	
15	6	15	8			23		7	
16	0	16	9			24		8	
17	3	17	10			25		9	
18	6	18	11			26		10	
19	0	19	12		1	27		11	
20	3	20	13		2	28		12	
21	6	21	14		3	29		13	
22	0	22			4	30		14	
23	3	23			5	31		15	
24	6	24			6	32		16	
25	0	25			7	33		17	1
26	3	26			8	34		18	2
27	6	27			9	35		19	3
28	0	28			10	36		20	4
29	3	29			11	37		21	5
30	6	30			12	38		22	6
31	0	31			13	39		23	7
32	3	32			14	40		24	8
33	6	33			15	41			9
34	0	34			16	42			10
35	3	35			17	43			11
36	6	36		1	18	44			12
37	0	37		2	19	45			13
38	3	38		3	20	46			14
39	6	39		4		47			15
40	0	40		5		48			16
41	3	41		6		49			17
42	6	42		7		50			18
43	0	43		8		51			19
44	3	44		9		52			20
45	6	45		10		1			21
46	0	46		11		2			22
47	3	47		12		3			23
48	6	48		13		4			24
49	0	49		14		5			25
50	3	50		15		6			26
51	6	51		16		7			27
52	0	52		17		8			28

Table 11 — Two Data Column MicroPDF417 Row Number Assignments

ALL Versions		Rotation-0 Family					Rotation-8 Family			
Left RAP	Cluster	Right RAP	Row numbers for				Right RAP	Row numbers for		
			VER. 2x8	VER. 2x14	VER. 2x17	VER. 2x20		VER. 2x11	VER. 2x23	VER. 2x26
1	0	1	1				9	1		
2	3	2	2				10	2		
3	6	3	3				11	3		
4	0	4	4				12	4		
5	3	5	5				13	5		
6	6	6	6				14	6		
7	0	7	7				15	7		
8	3	8	8	1			16	8		
9	6	9		2			17	9	1	
10	0	10		3			18	10	2	
11	3	11		4			19	11	3	
12	6	12		5			20		4	
13	0	13		6			21		5	
14	3	14		7			22		6	
15	6	15		8			23		7	
16	0	16		9			24		8	
17	3	17		10			25		9	
18	6	18		11			26		10	
19	0	19		12		1	27		11	
20	3	20		13		2	28		12	
21	6	21		14		3	29		13	
22	0	22				4	30		14	
23	3	23				5	31		15	
24	6	24				6	32		16	
25	0	25				7	33		17	
26	3	26				8	34		18	
27	6	27				9	35		19	1
28	0	28				10	36		20	2
29	3	29				11	37		21	3
30	6	30				12	38		22	4
31	0	31				13	39		23	5
32	3	32				14	40			6
33	6	33				15	41			7
34	0	34				16	42			8
35	3	35				17	43			9
36	6	36			1	18	44			10
37	0	37			2	19	45			11
38	3	38			3	20	46			12
39	6	39			4		47			13
40	0	40			5		48			14
41	3	41			6		49			15
42	6	42			7		50			16
43	0	43			8		51			17
44	3	44			9		52			18
45	6	45			10		1			19
46	0	46			11		2			20
47	3	47			12		3			21
48	6	48			13		4			22
49	0	49			14		5			23
50	3	50			15		6			24
51	6	51			16		7			25
52	0	52			17		8			26

Table 12 — Three / Four Data Column MicroPDF417 Row Number Assignments

ALL Versions		Rotation-0 Family							Rotation-8 Family				Rotation-16 Family				Rotation-24 Family			
Left RAP	Cluster	Center RAP	Right RAP	Row Numbers for:					Center RAP	Right RAP	Row Numbers for:		Center RAP	Right RAP	Row Numbers for:		Center RAP	Right RAP	Row Numbers for:	
				Ver cx6	Ver cx8	Ver cx10	Ver cx12	Ver cx15			Ver cx26	Ver cx32			Ver cx20	Ver cx38			Ver cx44	Ver 4x4
1	0	1	1	1					9	17	1		17	33	1		25	49	1	(note)
2	3	2	2	2					10	18	2		18	34	2		26	50	2	
3	6	3	3	3					11	19	3		19	35	3		27	51	3	
4	0	4	4	4					12	20	4		20	36	4		28	52	4	
5	3	5	5	5					13	21	5		21	37	5		29	1	5	
6	6	6	6	6					14	22	6		22	38	6		30	2	6	
7	0	7	7		1				15	23	7		23	39	7		31	3	7	
8	3	8	8		2				16	24	8		24	40	8		32	4	8	
9	6	9	9		3				17	25	9		25	41	9		33	5	9	
10	0	10	10		4				18	26	10		26	42	10		34	6	10	
11	3	11	11		5				19	27	11		27	43	11		35	7	11	
12	6	12	12		6				20	28	12		28	44	12		36	8	12	
13	0	13	13		7				21	29	13		29	45	13		37	9	13	
14	3	14	14		8				22	30	14		30	46	14		38	10	14	
15	6	15	15			1			23	31	15		31	47	15	1	39	11	15	
16	0	16	16			2			24	32	16		32	48	16	2	40	12	16	
17	3	17	17			3			25	33	17		33	49	17	3	41	13	17	
18	6	18	18			4			26	34	18		34	50	18	4	42	14	18	
19	0	19	19			5			27	35	19		35	51	19	5	43	15	19	
20	3	20	20			6			28	36	20		36	52	20	6	44	16	20	
21	6	21	21			7			29	37	21	1	37	1		7	45	17	21	
22	0	22	22			8			30	38	22	2	38	2		8	46	18	22	
23	3	23	23			9			31	39	23	3	39	3		9	47	19	23	
24	6	24	24			10			32	40	24	4	40	4		10	48	20	24	
25	0	25	25				1		33	41	25	5	41	5		11	49	21	25	
26	3	26	26				2		34	42	26	6	42	6		12	50	22	26	
27	6	27	27				3		35	43		7	43	7		13	51	23	27	
28	0	28	28				4		36	44		8	44	8		14	52	24	28	
29	3	29	29				5		37	45		9	45	9		15	1	25	29	
30	6	30	30				6		38	46		10	46	10		16	2	26	30	
31	0	31	31				7		39	47		11	47	11		17	3	27	31	
32	3	32	32				8		40	48		12	48	12		18	4	28	32	
33	6	33	33				9		41	49		13	49	13		19	5	29	33	
34	0	34	34				10		42	50		14	50	14		20	6	30	34	
35	3	35	35				11		43	51		15	51	15		21	7	31	35	
36	6	36	36				12		44	52		16	52	16		22	8	32	36	
37	0	37	37					1	45	1		17	1	17		23	9	33	37	
38	3	38	38					2	46	2		18	2	18		24	10	34	38	
39	6	39	39					3	47	3		19	3	19		25	11	35	39	
40	0	40	40					4	48	4		20	4	20		26	12	36	40	
41	3	41	41					5	49	5		21	5	21		27	13	37	41	
42	6	42	42					6	50	6		22	6	22		28	14	38	42	
43	0	43	43					7	51	7		23	7	23		29	15	39	43	
44	3	44	44					8	52	8		24	8	24		30	16	40	44	
45	6	45	45					9	1	9		25	9	25		31	17	41		
46	0	46	46					10	2	10		26	10	26		32	18	42		
47	3	47	47					11	3	11		27	11	27		33	19	43		1
48	6	48	48					12	4	12		28	12	28		34	20	44		2
49	0	49	49					13	5	13		29	13	29		35	21	45		3
50	3	50	50					14	6	14		30	14	30		36	22	46		4
51	6	51	51					15	7	15		31	15	31		37	23	47		
52	0	52	52						8	16		32	16	32		38	24	48		

NOTE A 4 column × 4 row version is defined, but not a 3 column × 4 row version.

Table 13 — Example of using Table 10 to define a 1×14 symbol

ALL Versions		Rotation-0 Family				Rotation-8 Family			
Left RAP	Cluster	Right RAP	Row Numbers for:			Right RAP	Row Numbers for:		
			VER. 1x14	VER. 1x17	VER. 1x20		VER. 1x11	VER. 1x24	VER. 1x28
1	0	1				9	1		
2	3	2				10	2		
3	6	3				11	3		
4	0	4				12	4		
5	3	5				13	5		
6	6	6				14	6		
7	0	7				15	7		
8	3	8	1			16	8		
9	6	9	2			17	9	1	
10	0	10	3			18	10	2	
11	3	11	4			19	11	3	
12	6	12	5			20		4	
13	0	13	6			21		5	
14	3	14	7					6	
15	6	15	8					7	
16	0	16	9					8	
17	3	17	10					9	
18	6	18	11					10	
19	0	19	12		1			11	
20	3	20	13		2			12	
21	6	21	14		3			13	
22	0	22			4			14	
23	3	23			5			15	
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.

Table 14 — Row structure of 1×14 MicroPDF417 symbol

Row Number	Cluster Number	Left RAP Number	Right RAP Number
1	3	8	8
2	6	9	9
3	0	10	10
4	3	11	11
5	6	12	12
6	0	13	13
7	3	14	14
8	6	15	15
9	0	16	16
10	3	17	17
11	6	18	18
12	0	19	19
13	3	20	20
14	6	21	21

Table 15 — Row structure of 4×38 MicroPDF417 symbol

Row number	Cluster number	Left RAP number	Center RAP number	Right RAP number
1	6	15	31	47
2	0	16	32	48
3	3	17	33	49
4	6	18	34	50
5	0	19	35	51
6	3	20	36	52
7	6	21	37	1
8	0	22	38	2
:	:	:	:	:
37	6	51	15	31
38	0	52	16	32

5.11.4 Row encoding

In each row, the Row Address Patterns are determined from Tables 10 to 12. The PDF417 symbol characters representing data and/or error correction codewords within each row shall conform with the cluster number that is also determined from Tables 10 to 12.

The symbol shall be encoded row by row, taking *c* (the number of data columns) codewords into each row, commencing with the most significant data codeword and ending with the final error correction codewords.

5.12 Printing Row Address Patterns

MicroPDF417 does not have a truncated form. Unlike PDF417, which has an option to omit row indicators, MicroPDF417 must be printed with all specified Row Address Patterns.

5.13 Structured Append

Structured Append provides a mechanism for the data in a file to be split into blocks and be represented in more than one MicroPDF417 symbol. This mechanism is identical to the Macro PDF417 method defined in the PDF417 specification (ISO/IEC 15438) and is similar to the Structured Append feature of other symbologies.

Each Structured Append symbol shall contain additional control information to enable the original data file to be properly reconstructed, irrespective of the sequence in which the individual MicroPDF417 symbols are scanned and decoded.

Up to 99 999 individual MicroPDF417 symbols may be used to encode data in Structured Append.

The first codeword of the first symbol in a Structured Append series follows the rules for the ECI Descriptor codeword defined in 5.2.4.1; the first codeword of the second and subsequent symbols does not follow this special interpretation but rather is the first codeword representing source data in these symbols. Full details of the procedures for Structured Append are given in Annex H*.*

5.13.1 Compaction modes and Structured Append

The Structured Append Control Block has a predefined encoding method, so codeword 928 causes the termination of any compaction mode sequence in the body of the symbol. The Segment Index field shall be encoded in Numeric Compaction mode. Each defined Structured Append optional field has a specific, implied initial compaction mode, and the beginning of a new optional field serves to terminate the compaction mode from the previous field *(*see Annex H.2.3*)* and initiate its default mode. Specifically, even if two consecutive optional fields both use the Text Compaction mode, the Alpha sub-mode is reset when codeword 923 is encountered.

5.13.2 ECIs and Structured Append

Subject to the constraints defined in 5.5.3, ECIs may occur in the message encoded in a single symbol or Structured Append set of 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.

The ECI interpretation(s) in the body of the data codeword stream do not extend into the Structured Append Control Block although they resume automatically at the beginning of the following symbol. The Control Block's data is interpreted using the default ECI (000003), unless ECI escape sequences are explicitly encoded in an optional field in the Control Block; the effect of any such ECI is automatically terminated at the end of the field in which it appears.

NOTE When implemented as GLIs according to early PDF417 specifications (e.g. those published by AIM USA and AIM Europe), encodation implies a reset to GLI 0 (equivalent to ECI 000000) at the start of each symbol. If it is intended for GLI 1 to persist into the next symbol, then GLI 1 shall be explicitly encoded at the start of this next symbol. As encoders compliant with these earlier standards will be in use for some time, advice is given in Annex M on how to achieve compatibility with this specification.

5.14 User guidelines

5.14.1 Human readable interpretation

MicroPDF417 symbols are capable of encoding large amounts of data, which means that a human readable interpretation of the data characters may not be practical. As an alternative, descriptive text, rather than literal text, may accompany the symbol. The message may be printed anywhere in the area surrounding the symbol, but should not interfere with the symbol itself nor the quiet zones. Font and character size are not specified by this standard, but may be by application standards.

5.14.2 Autodiscrimination capability

MicroPDF417 can be used in an autodiscrimination environment with a number of other symbologies (see Annex S).

5.14.3 User-defined application parameters

Application standards shall define parameters of MicroPDF417 symbols specified in this standard as variable, as follows:

5.14.3.1 Symbology and dimensional characteristics

Application standards shall specify the following data, symbology and dimensional parameters:

— The selection and use of Extended Channel Interpretations, if required, to extend data encodation beyond the default interpretations of the basic modes.

— The volume of data in the symbol, which may be fixed, variable or variable up to a defined maximum.

— Range of *X*-dimension.

— Range of *Y*-dimension.

— Symbol parameters: the range of permissible aspect ratios and/or whether symbol width or height has a maximum size.

— Optionally, the selection of the symbol version.

NOTE Additional factors which should be taken into consideration when specifying MicroPDF417 applications are given in Annexes O and S.

5.14.3.2 Test specification

The parameters for the evaluation of symbols (see 5.14.4) shall be defined by specifying a quality grade in accordance with ISO/IEC 15415 in the application standard.

This grade is expressed in the form:

grade / aperture / peak response wavelength

The following example illustrates the types of value which need to be expressed:

1,5 / 10 / 660

where:	1,5	is the overall symbol quality grade
	10	is the measuring aperture reference number (approximately in thousandths of an inch; in this example 0,25mm diameter)
	660	is the peak response wavelength in nanometers

NOTE: ISO/IEC 15415 gives guidance on selection of grading parameters in application specifications. The values appropriate for the application shall be defined in the application standard.

5.14.4 MicroPDF417 symbol quality

The symbol quality of MicroPDF417 symbols shall be assessed in accordance with ISO/IEC 15415 for multi-row symbols with cross-row scanning capability. Annex I gives specific guidance for the application of ISO/IEC 15415 to MicroPDF417.

5.14.5 Separation of multiple symbols

Multiple MicroPDF417 symbols should have sufficient separation to ensure that they do not appear simultaneously in the scanner's field of view.

5.15 Reference decode algorithm

The reference decode algorithm for MicroPDF417 is defined in Annex J of this document. This reference decode algorithm is the basis for print quality assessment according to ISO/IEC 15415.

5.16 Error detection and error correction procedure

As part of the decode procedure, it is possible to reconstruct the symbol for erasures and substitution errors within the error correction capacity of the symbol. This can be done by using the procedure set out in Annex K.

5.17 Transmitted data

5.17.1 Transmitted data in the basic (default) interpretation

All data codewords shall be translated into user data and transmitted as 8-bit bytes, whether this data be encoded in Text Compaction, Byte Compaction or Numeric Compaction mode. Row Address Patterns, mode switching codewords, pad codewords and error correction codewords shall not be transmitted.

5.17.2 Transmission protocol for Extended Channel Interpretation (ECI)

In systems where ECIs are supported, a symbology identifier prefix shall be used with every transmission (see 5.17.5 and Annex L). Structured Append Control Blocks (if transmitted) shall be treated as part of a control set of escape sequences which operate in conjunction with the ECI transmission protocol (see 5.17.3 and Annex H).

Three codewords (925, 926 and 927) signal the encodation of an ECI value and are decoded as byte values as follows:

1. If the ECI sequence begins with codeword 927:

    a. Codeword 927 is transmitted as the escape character 92, which represents reverse solidus (\), or backslash, in the default encodation.

    b. The next codeword is converted into a 6-digit value, by placing leading zeros before the codeword. The 6-digit value is transmitted as the six corresponding byte values in the range 48 to 57.

    EXAMPLE:

    Symbol encodes: [927] [123]

    Data transmission (byte): 92, 48, 48, 48, 49, 50, 51

    ASCII interpretation: \000123

2. If the ECI sequence begins with codeword 926:

    a. Codeword 926 is transmitted as escape character 92.

    b. The next two codewords are converted into a 6-digit value, with leading zeros if required, using the following formula:

$$([\text{1st codeword}] + 1) \times 900 + [\text{2nd codeword}]$$

The 6-digit value is transmitted as the six corresponding byte values in the range 48 to 57.

EXAMPLE:

Symbol encodes: [926] [136] [156]

Data transmission (byte): 92, 49, 50, 51, 52, 53, 54

ASCII interpretation: \123456

3. If the ECI sequence begins with codeword 925:

   a. Codeword 925 is transmitted as escape character 92.

   b. The next codeword is converted into a 6-digit value by adding the value 810 900 to it. The 6-digit value is transmitted as the six corresponding byte values in the range 48 to 57.

EXAMPLE:

Symbol encodes: [925] [456]

Data transmission (byte): 92, 56, 49, 49, 51, 53, 54

ASCII interpretation: \811356

The procedure is repeated for each occurrence of an Extended Channel Interpretation (ECI) sequence.

If the first codeword d_{n-1} of a MicroPDF417 codeword stream has a value between 0 and 899 inclusive, then it shall be interpreted by the MicroPDF417 decoder as a Transformation ECI value between 000900 and 001799 inclusive by adding 900 to the value of the codeword and prefixing it with three or two leading zeros. In this case, the decoder shall transmit either an ECI escape sequence or a GLI escape sequence, depending on which protocol the decoder is programmed to use. If using the ECI protocol, the decoder shall transmit seven bytes consisting of a "\" (ASCII 92), followed by the 6-digit ECI value. If using the GLI protocol it shall transmit twelve bytes in the format of

\926\000\nnn

where nnn is the 3-digit representation of the first codeword.

NOTE: this special interpretation applies only at the start of a MicroPDF417 codeword stream; thus, no special interpretation is applied to d_{n-1} of the second or subsequent symbols of a Structured Append sequence of MicroPDF417 symbols.

Application software recognising the 7-byte escape sequence of 92 followed by six bytes (each in the range 48 to 57) should interpret all subsequent characters until the end of the encoded data, or until another single byte 92 is encountered, as being from the ECI defined by the 6-digit sequence.

If the reverse solidus, or other character represented by byte 92, needs to be used as encoded data, transmission shall be as follows. Whenever byte 92 occurs as data, two bytes of that value shall be transmitted; thus a single occurrence is always an escape character and a double occurrence indicates true data.

EXAMPLE:

Encoded data:	A\\B\C
Transmission:	A\\\\B\\C

5.17.3 Transmitted data for Structured Append

The protocol for transmitted data for Structured Append is included in Annex H.6.

5.17.4 Transmission of reserved codewords using the ECI protocol

When operating under the ECI transmission protocol, MicroPDF417 decoders should transmit a reserved codeword escape sequence of six bytes (interpreted as '\CnnnC'), representing an escape character (92) followed by 'C' (67), three digits which represent the decimal value of the reserved codeword, followed by another 'C', which terminates the escape sequence in a symbology-independent manner. The data codewords which follow the reserved codeword are not interpreted by the decoder according to any compaction mode, but instead are transmitted as a series of escape sequences representing the codewords using the same 6-byte escape sequence defined earlier in this paragraph. All remaining data codewords are transmitted in this manner, until one of the following is reached:

- the end of the encoded data in the symbol
- a latch to a recognised compaction mode
- a Structured Append Control Block function codeword (928, 923, or 922)

Codeword 913 (byte shift) is only permitted from Text Compaction mode, and thus shall not be part of the codeword stream while in this process of sending escaped uninterpreted codewords.

NOTE: This protocol can properly transmit the message syntax of any reserved codeword whose future definition is to provide either a signalling function or to represent a new compaction mode.

5.17.5 Symbology identifier

Once the structure of the data (in terms of Structured Append, ECI, etc.) has been identified, the appropriate symbology identifier should be added as a preamble to the transmitted data by the decoder. See Annex L for the symbology identifiers which apply to MicroPDF417. See also 5.4.1.5 for options in the case of Code 128 emulation.

5.17.6 Transmission using older protocols

The introduction of the Extended Channel Interpretation system, common to a number of symbologies, has had an impact on pre-existing symbologies including PDF417. The basic encoding and decoding rules remain identical in this standard with those in the 1994 versions of PDF417 published by AIM Europe and AIM USA. Transmission for both ECIs and Structured Append is different in format, but conveys equivalent information.

All MicroPDF417 decoding equipment and software should conform with this standard. However, PDF417 equipment conforming with the earlier standard will still be in existence for a number of years. Therefore, so that MicroPDF417 can be used transparently in existing PDF417 applications, where appropriate, MicroPDF417-capable readers should be configurable to support either the GLI or the ECI protocol. Annex M defines the rules which shall be followed when using decoding equipment and software not capable of being compliant with the current ECI and Structured Append protocols. In this way, old and new decoding equipment can continue to co-exist.

Annex A
(normative)

Encoding/decoding table of PDF417 symbol character bar-space sequences

The table on the subsequent pages of this Annex gives the value of each codeword and the bar-space sequence for symbol characters in clusters 0, 3 and 6. The table is also included on the companion diskette which forms part of this International Standard, as an aid to implementation.

The values ***e*** used in the decoding phase can be derived from the bar-space sequence by the following equation:

$$e_i = x_i + x_{i+1}$$

where x_i is the width in modules of the i-th element in the sequence

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
0	31111136	51111125	21111155
1	41111144	61111133	31111163
2	51111152	41111216	11111246
3	31111235	51111224	21111254
4	41111243	61111232	31111262
5	51111251	41111315	11111345
6	21111326	51111323	21111353
7	31111334	61111331	31111361
8	21111425	41111414	11111444
9	11111516	51111422	21111452
10	21111524	41111513	11111543
11	11111615	51111521	61112114
12	21112136	41111612	11112155
13	31112144	41112125	21112163
14	41112152	51112133	61112213
15	21112235	61112141	11112254
16	31112243	31112216	21112262
17	41112251	41112224	61112312
18	11112326	51112232	11112353
19	21112334	31112315	21112361
20	11112425	41112323	61112411
21	11113136	51112331	11112452
22	21113144	31112414	51113114
23	31113152	41112422	61113122
24	11113235	31112513	11113163
25	21113243	41112521	51113213
26	31113251	31112612	61113221
27	11113334	31113125	11113262
28	21113342	41113133	51113312
29	11114144	51113141	11113361
30	21114152	21113216	51113411
31	11114243	31113224	41114114
32	21114251	41113232	51114122
33	11115152	21113315	41114213
34	51116111	31113323	51114221
35	31121135	41113331	41114312
36	41121143	21113414	41114411
37	51121151	31113422	31115114

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
38	21121226	21113513	41115122
39	31121234	31113521	31115213
40	41121242	21113612	41115221
41	21121325	21114125	31115312
42	31121333	31114133	31115411
43	11121416	41114141	21116114
44	21121424	11114216	31116122
45	31121432	21114224	21116213
46	11121515	31114232	31116221
47	21121523	11114315	21116312
48	11121614	21114323	11121146
49	21122135	31114331	21121154
50	31122143	11114414	31121162
51	41122151	21114422	11121245
52	11122226	11114513	21121253
53	21122234	21114521	31121261
54	31122242	11115125	11121344
55	11122325	21115133	21121352
56	21122333	31115141	11121443
57	31122341	11115224	21121451
58	11122424	21115232	11121542
59	21122432	11115323	61122113
60	11123135	21115331	11122154
61	21123143	11115422	21122162
62	31123151	11116133	61122212
63	11123234	21116141	11122253
64	21123242	11116232	21122261
65	11123333	11116331	61122311
66	21123341	41121116	11122352
67	11124143	51121124	11122451
68	21124151	61121132	51123113
69	11124242	41121215	61123121

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
70	11124341	51121223	11123162
71	21131126	61121231	51123212
72	31131134	41121314	11123261
73	41131142	51121322	51123311
74	21131225	41121413	41124113
75	31131233	51121421	51124121
76	41131241	41121512	41124212
77	11131316	41121611	41124311
78	21131324	31122116	31125113
79	31131332	41122124	41125121
80	11131415	51122132	31125212
81	21131423	31122215	31125311
82	11131514	41122223	21126113
83	11131613	51122231	31126121
84	11132126	31122314	21126212
85	21132134	41122322	21126311
86	31132142	31122413	11131145
87	11132225	41122421	21131153
88	21132233	31122512	31131161
89	31132241	31122611	11131244
90	11132324	21123116	21131252
91	21132332	31123124	11131343
92	11132423	41123132	21131351
93	11132522	21123215	11131442
94	11133134	31123223	11131541
95	21133142	41123231	61132112
96	11133233	21123314	11132153
97	21133241	31123322	21132161
98	11133332	21123413	61132211
99	11134142	31123421	11132252
100	21141125	21123512	11132351
101	31141133	21123611	51133112

Codeword	Bar-space sequence		
	Cluster 0	Cluster 3	Cluster 6
	BSBSBSBS	BSBSBSBS	BSBSBSBS
102	41141141	11124116	11133161
103	11141216	21124124	51133211
104	21141224	31124132	41134112
105	31141232	11124215	41134211
106	11141315	21124223	31135112
107	21141323	31124231	31135211
108	31141331	11124314	21136112
109	11141414	21124322	21136211
110	21141422	11124413	11141144
111	11141513	21124421	21141152
112	21141521	11124512	11141243
113	11142125	11125124	21141251
114	21142133	21125132	11141342
115	31142141	11125223	11141441
116	11142224	21125231	61142111
117	21142232	11125322	11142152
118	11142323	11125421	11142251
119	21142331	11126132	51143111
120	11142422	11126231	41144111
121	11142521	41131115	31145111
122	21143141	51131123	11151143
123	11143331	61131131	21151151
124	11151116	41131214	11151242
125	21151124	51131222	11151341
126	31151132	41131313	11152151
127	11151215	51131321	11161142
128	21151223	41131412	11161241
129	31151231	41131511	12111146
130	11151314	31132115	22111154
131	21151322	41132123	32111162
132	11151413	51132131	12111245
133	21151421	31132214	22111253

Codeword	Bar-space sequence		
	Cluster 0	Cluster 3	Cluster 6
	BSBSBSBS	BSBSBSBS	BSBSBSBS
134	11151512	41132222	32111261
135	11152124	31132313	12111344
136	11152223	41132321	22111352
137	11152322	31132412	12111443
138	11161115	31132511	22111451
139	31161131	21133115	12111542
140	21161222	31133123	62112113
141	21161321	41133131	12112154
142	11161511	21133214	22112162
143	32111135	31133222	62112212
144	42111143	21133313	12112253
145	52111151	31133321	22112261
146	22111226	21133412	62112311
147	32111234	21133511	12112352
148	42111242	11134115	12112451
149	22111325	21134123	52113113
150	32111333	31134131	62113121
151	42111341	11134214	12113162
152	12111416	21134222	52113212
153	22111424	11134313	12113261
154	12111515	21134321	52113311
155	22112135	11134412	42114113
156	32112143	11134511	52114121
157	42112151	11135123	42114212
158	12112226	21135131	42114311
159	22112234	11135222	32115113
160	32112242	11135321	42115121
161	12112325	11136131	32115212
162	22112333	41141114	32115311
163	12112424	51141122	22116113
164	12112523	41141213	32116121
165	12113135	51141221	22116212

Codeword	Bar-space sequence		
	Cluster 0	Cluster 3	Cluster 6
	BSBSBSBS	BSBSBSBS	BSBSBSBS
166	22113143	41141312	22116311
167	32113151	41141411	21211145
168	12113234	31142114	31211153
169	22113242	41142122	41211161
170	12113333	31142213	11211236
171	12113432	41142221	21211244
172	12114143	31142312	31211252
173	22114151	31142411	11211335
174	12114242	21143114	21211343
175	12115151	31143122	31211351
176	31211126	21143213	11211434
177	41211134	31143221	21211442
178	51211142	21143312	11211533
179	31211225	21143411	21211541
180	41211233	11144114	11211632
181	51211241	21144122	12121145
182	21211316	11144213	22121153
183	31211324	21144221	32121161
184	41211332	11144312	11212145
185	21211415	11144411	12121244
186	31211423	11145122	22121252
187	41211431	11145221	11212244
188	21211514	41151113	21212252
189	31211522	51151121	22121351
190	22121126	41151212	11212343
191	32121134	41151311	12121442
192	42121142	31152113	11212442
193	21212126	41152121	12121541
194	22121225	31152212	11212541
195	32121233	31152311	62122112
196	42121241	21153113	12122153
197	21212225	31153121	22122161

Codeword	Bar-space sequence		
	Cluster 0	Cluster 3	Cluster 6
	BSBSBSBS	BSBSBSBS	BSBSBSBS
198	31212233	21153212	61213112
199	41212241	21153311	62122211
200	11212316	11154113	11213153
201	12121415	21154121	12122252
202	22121423	11154212	61213211
203	32121431	11154311	11213252
204	11212415	41161112	12122351
205	21212423	41161211	11213351
206	11212514	31162112	52123112
207	12122126	31162211	12123161
208	22122134	21163112	51214112
209	32122142	21163211	52123211
210	11213126	42111116	11214161
211	12122225	52111124	51214211
212	22122233	62111132	42124112
213	32122241	42111215	41215112
214	11213225	52111223	42124211
215	21213233	62111231	41215211
216	31213241	42111314	32125112
217	11213324	52111322	31216112
218	12122423	42111413	32125211
219	11213423	52111421	31216211
220	12123134	42111512	22126112
221	22123142	42111611	22126211
222	11214134	32112116	11221136
223	12123233	42112124	21221144
224	22123241	52112132	31221152
225	11214233	32112215	11221235
226	21214241	42112223	21221243
227	11214332	52112231	31221251
228	12124142	32112314	11221334
229	11215142	42112322	21221342

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
230	12124241	32112413	11221433
231	11215241	42112421	21221441
232	31221125	32112512	11221532
233	41221133	32112611	11221631
234	51221141	22113116	12131144
235	21221216	32113124	22131152
236	31221224	42113132	11222144
237	41221232	22113215	12131243
238	21221315	32113223	22131251
239	31221323	42113231	11222243
240	41221331	22113314	21222251
241	21221414	32113322	11222342
242	31221422	22113413	12131441
243	21221513	32113421	11222441
244	21221612	22113512	62132111
245	22131125	22113611	12132152
246	32131133	12114116	61223111
247	42131141	22114124	11223152
248	21222125	32114132	12132251
249	22131224	12114215	11223251
250	32131232	22114223	52133111
251	11222216	32114231	51224111
252	12131315	12114314	42134111
253	31222232	22114322	41225111
254	32131331	12114413	32135111
255	11222315	22114421	31226111
256	12131414	12114512	22136111
257	22131422	12115124	11231135
258	11222414	22115132	21231143
259	21222422	12115223	31231151
260	22131521	22115231	11231234
261	12131612	12115322	21231242

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
262	12132125	12115421	11231333
263	22132133	12116132	21231341
264	32132141	12116231	11231432
265	11223125	51211115	11231531
266	12132224	61211123	12141143
267	22132232	11211164	22141151
268	11223224	51211214	11232143
269	21223232	61211222	12141242
270	22132331	11211263	11232242
271	11223323	51211313	12141341
272	12132422	61211321	11232341
273	12132521	11211362	12142151
274	12133133	51211412	11233151
275	22133141	51211511	11241134
276	11224133	42121115	21241142
277	12133232	52121123	11241233
278	11224232	62121131	21241241
279	12133331	41212115	11241332
280	11224331	42121214	11241431
281	11225141	61212131	12151142
282	21231116	41212214	11242142
283	31231124	51212222	12151241
284	41231132	52121321	11242241
285	21231215	41212313	11251133
286	31231223	42121412	21251141
287	41231231	41212412	11251232
288	21231314	42121511	11251331
289	31231322	41212511	12161141
290	21231413	32122115	11252141
291	31231421	42122123	11261132
292	21231512	52122131	11261231
293	21231611	31213115	13111145

Codeword	Bar-space sequence		
	Cluster 0	Cluster 3	Cluster 6
	BSBSBSBS	BSBSBSBS	BSBSBSBS
294	12141116	32122214	23111153
295	22141124	42122222	33111161
296	32141132	31213214	13111244
297	11232116	41213222	23111252
298	12141215	42122321	13111343
299	22141223	31213313	23111351
300	32141231	32122412	13111442
301	11232215	31213412	13111541
302	21232223	32122511	63112112
303	31232231	31213511	13112153
304	11232314	22123115	23112161
305	12141413	32123123	63112211
306	22141421	42123131	13112252
307	11232413	21214115	13112351
308	21232421	22123214	53113112
309	11232512	32123222	13113161
310	12142124	21214214	53113211
311	22142132	31214222	43114112
312	11233124	32123321	43114211
313	12142223	21214313	33115112
314	22142231	22123412	33115211
315	11233223	21214412	23116112
316	21233231	22123511	23116211
317	11233322	21214511	12211136
318	12142421	12124115	22211144
319	11233421	22124123	32211152
320	11234132	32124131	12211235
321	11234231	11215115	22211243
322	21241115	12124214	32211251
323	31241123	22124222	12211334
324	41241131	11215214	22211342
325	21241214	21215222	12211433

Codeword	Bar-space sequence		
	Cluster 0	Cluster 3	Cluster 6
	BSBSBSBS	BSBSBSBS	BSBSBSBS
326	31241222	22124321	22211441
327	21241313	11215313	12211532
328	31241321	12124412	12211631
329	21241412	11215412	13121144
330	21241511	12124511	23121152
331	12151115	12125123	12212144
332	22151123	22125131	13121243
333	32151131	11216123	23121251
334	11242115	12125222	12212243
335	12151214	11216222	22212251
336	22151222	12125321	12212342
337	11242214	11216321	13121441
338	21242222	12126131	12212441
339	22151321	51221114	63122111
340	11242313	61221122	13122152
341	12151412	11221163	62213111
342	11242412	51221213	12213152
343	12151511	61221221	13122251
344	12152123	11221262	12213251
345	11243123	51221312	53123111
346	11243222	11221361	52214111
347	11243321	51221411	43124111
348	31251122	42131114	42215111
349	31251221	52131122	33125111
350	21251411	41222114	32216111
351	22161122	42131213	23126111
352	12161213	52131221	21311135
353	11252213	41222213	31311143
354	11252312	51222221	41311151
355	11252411	41222312	11311226
356	23111126	42131411	21311234
357	33111134	41222411	31311242

Codeword	Bar-space sequence		
	Cluster 0	Cluster 3	Cluster 6
	BSBSBSBS	BSBSBSBS	BSBSBSBS
358	43111142	32132114	11311325
359	23111225	42132122	21311333
360	33111233	31223114	31311341
361	13111316	32132213	11311424
362	23111324	42132221	21311432
363	33111332	31223213	11311523
364	13111415	41223221	21311531
365	23111423	31223312	11311622
366	13111514	32132411	12221135
367	13111613	31223411	22221143
368	13112126	22133114	32221151
369	23112134	32133122	11312135
370	33112142	21224114	12221234
371	13112225	22133213	22221242
372	23112233	32133221	11312234
373	33112241	21224213	21312242
374	13112324	31224221	22221341
375	23112332	21224312	11312333
376	13112423	22133411	12221432
377	13112522	21224411	11312432
378	13113134	12134114	12221531
379	23113142	22134122	11312531
380	13113233	11225114	13131143
381	23113241	12134213	23131151
382	13113332	22134221	12222143
383	13114142	11225213	13131242
384	13114241	21225221	11313143
385	32211125	11225312	12222242
386	42211133	12134411	13131341
387	52211141	11225411	11313242
388	22211216	12135122	12222341
389	32211224	11226122	11313341

Codeword	Bar-space sequence		
	Cluster 0	Cluster 3	Cluster 6
	BSBSBSBS	BSBSBSBS	BSBSBSBS
390	42211232	12135221	13132151
391	22211315	11226221	12223151
392	32211323	51231113	11314151
393	42211331	61231121	11321126
394	22211414	11231162	21321134
395	32211422	51231212	31321142
396	22211513	11231261	11321225
397	32211521	51231311	21321233
398	23121125	42141113	31321241
399	33121133	52141121	11321324
400	43121141	41232113	21321332
401	22212125	51232121	11321423
402	23121224	41232212	21321431
403	33121232	42141311	11321522
404	12212216	41232311	11321621
405	13121315	32142113	12231134
406	32212232	42142121	22231142
407	33121331	31233113	11322134
408	12212315	32142212	12231233
409	22212323	31233212	22231241
410	23121422	32142311	11322233
411	12212414	31233311	21322241
412	13121513	22143113	11322332
413	12212513	32143121	12231431
414	13122125	21234113	11322431
415	23122133	31234121	13141142
416	33122141	21234212	12232142
417	12213125	22143311	13141241
418	13122224	21234311	11323142
419	32213141	12144113	12232241
420	12213224	22144121	11323241
421	22213232	11235113	11331125

Codeword	Bar-space sequence		
	Cluster 0	Cluster 3	Cluster 6
	BSBSBSBS	BSBSBSBS	BSBSBSBS
422	23122331	12144212	21331133
423	12213323	11235212	31331141
424	13122422	12144311	11331224
425	12213422	11235311	21331232
426	13123133	12145121	11331323
427	23123141	11236121	21331331
428	12214133	51241112	11331422
429	13123232	11241161	11331521
430	12214232	51241211	12241133
431	13123331	42151112	22241141
432	13124141	41242112	11332133
433	12215141	42151211	12241232
434	31311116	41242211	11332232
435	41311124	32152112	12241331
436	51311132	31243112	11332331
437	31311215	32152211	13151141
438	41311223	31243211	12242141
439	51311231	22153112	11333141
440	31311314	21244112	11341124
441	41311322	22153211	21341132
442	31311413	21244211	11341223
443	41311421	12154112	21341231
444	31311512	11245112	11341322
445	22221116	12154211	11341421
446	32221124	11245211	12251132
447	42221132	51251111	11342132
448	21312116	42161111	12251231
449	22221215	41252111	11342231
450	41312132	32162111	11351123
451	42221231	31253111	21351131
452	21312215	22163111	11351222
453	31312223	21254111	11351321
454	41312231	43111115	12261131
455	21312314	53111123	11352131
456	22221413	63111131	11361122
457	32221421	43111214	11361221
458	21312413	53111222	14111144
459	31312421	43111313	24111152
460	22221611	53111321	14111243
461	13131116	43111412	24111251
462	23131124	43111511	14111342
463	33131132	33112115	14111441
464	12222116	43112123	14112152
465	13131215	53112131	14112251
466	23131223	33112214	54113111
467	33131231	43112222	44114111
468	11313116	33112313	34115111
469	12222215	43112321	24116111
470	22222223	33112412	13211135
471	32222231	33112511	23211143
472	11313215	23113115	33211151
473	21313223	33113123	13211234
474	31313231	43113131	23211242
475	23131421	23113214	13211333
476	11313314	33113222	23211341
477	12222413	23113313	13211432
478	22222421	33113321	13211531
479	11313413	23113412	14121143
480	13131611	23113511	24121151
481	13132124	13114115	13212143
482	23132132	23114123	14121242
483	12223124	33114131	13212242
484	13132223	13114214	14121341
485	23132231	23114222	13212341

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
486	11314124	13114313	14122151
487	12223223	23114321	13213151
488	22223231	13114412	12311126
489	11314223	13114511	22311134
490	21314231	13115123	32311142
491	13132421	23115131	12311225
492	12223421	13115222	22311233
493	13133132	13115321	32311241
494	12224132	13116131	12311324
495	13133231	52211114	22311332
496	11315132	62211122	12311423
497	12224231	12211163	22311431
498	31321115	52211213	12311522
499	41321123	62211221	12311621
500	51321131	12211262	13221134
501	31321214	52211312	23221142
502	41321222	12211361	12312134
503	31321313	52211411	13221233
504	41321321	43121114	23221241
505	31321412	53121122	12312233
506	31321511	42212114	13221332
507	22231115	43121213	12312332
508	32231123	53121221	13221431
509	42231131	42212213	12312431
510	21322115	52212221	14131142
511	22231214	42212312	13222142
512	41322131	43121411	14131241
513	21322214	42212411	12313142
514	31322222	33122114	13222241
515	32231321	43122122	12313241
516	21322313	32213114	21411125
517	22231412	33122213	31411133

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
518	21322412	43122221	41411141
519	22231511	32213213	11411216
520	21322511	42213221	21411224
521	13141115	32213312	31411232
522	23141123	33122411	11411315
523	33141131	32213411	21411323
524	12232115	23123114	31411331
525	13141214	33123122	11411414
526	23141222	22214114	21411422
527	11323115	23123213	11411513
528	12232214	33123221	21411521
529	22232222	22214213	11411612
530	23141321	32214221	12321125
531	11323214	22214312	22321133
532	21323222	23123411	32321141
533	13141412	22214411	11412125
534	11323313	13124114	12321224
535	12232412	23124122	22321232
536	13141511	12215114	11412224
537	12232511	13124213	21412232
538	13142123	23124221	22321331
539	23142131	12215213	11412323
540	12233123	22215221	12321422
541	13142222	12215312	11412422
542	11324123	13124411	12321521
543	12233222	12215411	11412521
544	13142321	13125122	13231133
545	11324222	12216122	23231141
546	12233321	13125221	12322133
547	13143131	12216221	13231232
548	11325131	61311113	11413133
549	31331114	11311154	12322232

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
550	41331122	21311162	13231331
551	31331213	61311212	11413232
552	41331221	11311253	12322331
553	31331312	21311261	11413331
554	31331411	61311311	14141141
555	22241114	11311352	13232141
556	32241122	11311451	12323141
557	21332114	52221113	11414141
558	22241213	62221121	11421116
559	32241221	12221162	21421124
560	21332213	51312113	31421132
561	31332221	61312121	11421215
562	21332312	11312162	21421223
563	22241411	12221261	31421231
564	21332411	51312212	11421314
565	13151114	52221311	21421322
566	23151122	11312261	11421413
567	12242114	51312311	21421421
568	13151213	43131113	11421512
569	23151221	53131121	11421611
570	11333114	42222113	12331124
571	12242213	43131212	22331132
572	22242221	41313113	11422124
573	11333213	51313121	12331223
574	21333221	43131311	22331231
575	13151411	41313212	11422223
576	11333312	42222311	21422231
577	12242411	41313311	11422322
578	11333411	33132113	12331421
579	12243122	43132121	11422421
580	11334122	32223113	13241132
581	11334221	33132212	12332132

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
582	41341121	31314113	13241231
583	31341311	32223212	11423132
584	32251121	33132311	12332231
585	22251212	31314212	11423231
586	22251311	32223311	11431115
587	13161113	31314311	21431123
588	12252113	23133113	31431131
589	11343113	33133121	11431214
590	13161311	22224113	21431222
591	12252311	23133212	11431313
592	24111125	21315113	21431321
593	14111216	22224212	11431412
594	24111224	23133311	11431511
595	14111315	21315212	12341123
596	24111323	22224311	22341131
597	34111331	21315311	11432123
598	14111414	13134113	12341222
599	24111422	23134121	11432222
600	14111513	12225113	12341321
601	24111521	13134212	11432321
602	14112125	11316113	13251131
603	24112133	12225212	12342131
604	34112141	13134311	11433131
605	14112224	11316212	11441114
606	24112232	12225311	21441122
607	14112323	11316311	11441213
608	24112331	13135121	21441221
609	14112422	12226121	11441312
610	14112521	61321112	11441411
611	14113133	11321153	12351122
612	24113141	21321161	11442122
613	14113232	61321211	12351221

Codeword	Bar-space sequence		
	Cluster 0	Cluster 3	Cluster 6
	BSBSBSBS	BSBSBSBS	BSBSBSBS
614	14113331	11321252	11442221
615	14114141	11321351	11451113
616	23211116	52231112	21451121
617	33211124	12231161	11451212
618	43211132	51322112	11451311
619	23211215	52231211	12361121
620	33211223	11322161	11452121
621	23211314	51322211	15111143
622	33211322	43141112	25111151
623	23211413	42232112	15111242
624	33211421	43141211	15111341
625	23211512	41323112	15112151
626	14121116	42232211	14211134
627	24121124	41323211	24211142
628	34121132	33142112	14211233
629	13212116	32233112	24211241
630	14121215	33142211	14211332
631	33212132	31324112	14211431
632	34121231	32233211	15121142
633	13212215	31324211	14212142
634	23212223	23143112	15121241
635	33212231	22234112	14212241
636	13212314	23143211	13311125
637	14121413	21325112	23311133
638	24121421	22234211	33311141
639	13212413	21325211	13311224
640	23212421	13144112	23311232
641	14121611	12235112	13311323
642	14122124	13144211	23311331
643	24122132	11326112	13311422
644	13213124	12235211	13311521
645	14122223	11326211	14221133

Codeword	Bar-space sequence		
	Cluster 0	Cluster 3	Cluster 6
	BSBSBSBS	BSBSBSBS	BSBSBSBS
646	24122231	61331111	24221141
647	13213223	11331152	13312133
648	23213231	11331251	14221232
649	13213322	52241111	13312232
650	14122421	51332111	14221331
651	14123132	43151111	13312331
652	13214132	42242111	15131141
653	14123231	41333111	14222141
654	13214231	33152111	13313141
655	32311115	32243111	12411116
656	42311123	31334111	22411124
657	52311131	23153111	32411132
658	32311214	22244111	12411215
659	42311222	21335111	22411223
660	32311313	13154111	32411231
661	42311321	12245111	12411314
662	32311412	11336111	22411322
663	32311511	11341151	12411413
664	23221115	44111114	22411421
665	33221123	54111122	12411512
666	22312115	44111213	12411611
667	23221214	54111221	13321124
668	33221222	44111312	23321132
669	22312214	44111411	12412124
670	32312222	34112114	13321223
671	33221321	44112122	23321231
672	22312313	34112213	12412223
673	23221412	44112221	22412231
674	22312412	34112312	12412322
675	23221511	34112411	13321421
676	22312511	24113114	12412421
677	14131115	34113122	14231132

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
678	24131123	24113213	13322132
679	13222115	34113221	14231231
680	14131214	24113312	12413132
681	33222131	24113411	13322231
682	12313115	14114114	12413231
683	13222214	24114122	21511115
684	23222222	14114213	31511123
685	24131321	24114221	41511131
686	12313214	14114312	21511214
687	22313222	14114411	31511222
688	14131412	14115122	21511313
689	12313313	14115221	31511321
690	13222412	53211113	21511412
691	14131511	63211121	21511511
692	13222511	13211162	12421115
693	14132123	53211212	22421123
694	24132131	13211261	32421131
695	13223123	53211311	11512115
696	14132222	44121113	12421214
697	12314123	54121121	22421222
698	13223222	43212113	11512214
699	14132321	44121212	21512222
700	12314222	43212212	22421321
701	13223321	44121311	11512313
702	14133131	43212311	12421412
703	13224131	34122113	11512412
704	12315131	44122121	12421511
705	41411114	33213113	11512511
706	51411122	34122212	13331123
707	41411213	33213212	23331131
708	51411221	34122311	12422123
709	41411312	33213311	13331222

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
710	41411411	24123113	11513123
711	32321114	34123121	12422222
712	42321122	23214113	13331321
713	31412114	24123212	11513222
714	41412122	23214212	12422321
715	42321221	24123311	11513321
716	31412213	23214311	14241131
717	41412221	14124113	13332131
718	31412312	24124121	12423131
719	32321411	13215113	11514131
720	31412411	14124212	21521114
721	23231114	13215212	31521122
722	33231122	14124311	21521213
723	22322114	13215311	31521221
724	23231213	14125121	21521312
725	33231221	13216121	21521411
726	21413114	62311112	12431114
727	22322213	12311153	22431122
728	32322221	22311161	11522114
729	21413213	62311211	12431213
730	31413221	12311252	22431221
731	23231411	12311351	11522213
732	21413312	53221112	21522221
733	22322411	13221161	11522312
734	21413411	52312112	12431411
735	14141114	53221211	11522411
736	24141122	12312161	13341122
737	13232114	52312211	12432122
738	14141213	44131112	13341221
739	24141221	43222112	11523122
740	12323114	44131211	12432221
741	13232213	42313112	11523221

Codeword	Bar-space sequence		
	Cluster 0	Cluster 3	Cluster 6
	BSBSBSBS	BSBSBSBS	BSBSBSBS
742	23232221	43222211	21531113
743	11414114	42313211	31531121
744	12323213	34132112	21531212
745	22323221	33223112	21531311
746	14141411	34132211	12441113
747	11414213	32314112	22441121
748	21414221	33223211	11532113
749	13232411	32314211	12441212
750	11414312	24133112	11532212
751	14142122	23224112	12441311
752	13233122	24133211	11532311
753	14142221	22315112	13351121
754	12324122	23224211	12442121
755	13233221	22315211	11533121
756	11415122	14134112	21541112
757	12324221	13225112	21541211
758	11415221	14134211	12451112
759	41421113	12316112	11542112
760	51421121	13225211	12451211
761	41421212	12316211	11542211
762	41421311	11411144	16111142
763	32331113	21411152	16111241
764	42331121	11411243	15211133
765	31422113	21411251	25211141
766	41422121	11411342	15211232
767	31422212	11411441	15211331
768	32331311	62321111	16121141
769	31422311	12321152	15212141
770	23241113	61412111	14311124
771	33241121	11412152	24311132
772	22332113	12321251	14311223
773	23241212	11412251	24311231
774	21423113	53231111	14311322
775	22332212	52322111	14311421
776	23241311	51413111	15221132
777	21423212	44141111	14312132
778	22332311	43232111	15221231
779	21423311	42323111	14312231
780	14151113	41414111	13411115
781	24151121	34142111	23411123
782	13242113	33233111	33411131
783	23242121	32324111	13411214
784	12333113	31415111	23411222
785	13242212	24143111	13411313
786	14151311	23234111	23411321
787	11424113	22325111	13411412
788	12333212	21416111	13411511
789	13242311	14144111	14321123
790	11424212	13235111	24321131
791	12333311	12326111	13412123
792	11424311	11421143	23412131
793	13243121	21421151	13412222
794	11425121	11421242	14321321
795	41431211	11421341	13412321
796	31432112	12331151	15231131
797	31432211	11422151	14322131
798	22342112	11431142	13413131
799	21433112	11431241	22511114
800	21433211	11441141	32511122
801	13252112	45111113	22511213
802	12343112	45111212	32511221
803	11434112	45111311	22511312
804	11434211	35112113	22511411
805	15111116	45112121	13421114

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
806	15111215	35112212	23421122
807	25111223	35112311	12512114
808	15111314	25113113	22512122
809	15111413	35113121	23421221
810	15111512	25113212	12512213
811	15112124	25113311	13421312
812	15112223	15114113	12512312
813	15112322	25114121	13421411
814	15112421	15114212	12512411
815	15113132	15114311	14331122
816	15113231	15115121	13422122
817	24211115	54211112	14331221
818	24211214	14211161	12513122
819	34211222	54211211	13422221
820	24211313	45121112	12513221
821	34211321	44212112	31611113
822	24211412	45121211	41611121
823	24211511	44212211	31611212
824	15121115	35122112	31611311
825	25121123	34213112	22521113
826	14212115	35122211	32521121
827	24212123	34213211	21612113
828	25121222	25123112	22521212
829	14212214	24214112	21612212
830	24212222	25123211	22521311
831	14212313	24214211	21612311
832	24212321	15124112	13431113
833	14212412	14215112	23431121
834	15121511	15124211	12522113
835	14212511	14215211	13431212
836	15122123	63311111	11613113
837	25122131	13311152	12522212

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
838	14213123	13311251	13431311
839	24213131	54221111	11613212
840	14213222	53312111	12522311
841	15122321	45131111	11613311
842	14213321	44222111	14341121
843	15123131	43313111	13432121
844	14214131	35132111	12523121
845	33311114	34223111	11614121
846	33311213	33314111	31621112
847	33311312	25133111	31621211
848	33311411	24224111	22531112
849	24221114	23315111	21622112
850	23312114	15134111	22531211
851	33312122	14225111	21622211
852	34221221	13316111	13441112
853	23312213	12411143	12532112
854	33312221	22411151	13441211
855	23312312	12411242	11623112
856	24221411	12411341	12532211
857	23312411	13321151	11623211
858	15131114	12412151	31631111
859	14222114	11511134	22541111
860	15131213	21511142	21632111
861	25131221	11511233	13451111
862	13313114	21511241	12542111
863	14222213	11511332	11633111
864	15131312	11511431	16211132
865	13313213	12421142	16211231
866	14222312	11512142	15311123
867	15131411	12421241	25311131
868	13313312	11512241	15311222
869	14222411	11521133	15311321

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
870	15132122	21521141	16221131
871	14223122	11521232	15312131
872	15132221	11521331	14411114
873	13314122	12431141	24411122
874	14223221	11522141	14411213
875	13314221	11531132	24411221
876	42411113	11531231	14411312
877	42411212	11541131	14411411
878	42411311	36112112	15321122
879	33321113	36112211	14412122
880	32412113	26113112	15321221
881	42412121	26113211	14412221
882	32412212	16114112	23511113
883	33321311	16114211	33511121
884	32412311	45212111	23511212
885	24231113	36122111	23511311
886	34231121	35213111	14421113
887	23322113	26123111	24421121
888	33322121	25214111	13512113
889	22413113	16124111	23512121
890	23322212	15215111	13512212
891	24231311	14311151	14421311
892	22413212	13411142	13512311
893	23322311	13411241	15331121
894	22413311	12511133	14422121
895	15141113	22511141	13513121
896	25141121	12511232	32611112
897	14232113	12511331	32611211
898	24232121	13421141	23521112
899	13323113	12512141	22612112

Codeword	Bar-space sequence		
	Cluster 0 BSBSBSBS	Cluster 3 BSBSBSBS	Cluster 6 BSBSBSBS
900	14232212	11611124	23521211
901	15141311	21611132	22612211
902	12414113	11611223	14431112
903	13323212	21611231	13522112
904	14232311	11611322	14431211
905	12414212	11611421	12613112
906	13323311	12521132	13522211
907	15142121	11612132	12613211
908	14233121	12521231	32621111
909	13324121	11612231	23531111
910	12415121	11621123	22622111
911	51511112	21621131	14441111
912	51511211	11621222	13532111
913	42421112	11621321	12623111
914	41512112	12531131	16311122
915	42421211	11622131	16311221
916	41512211	11631122	15411113
917	33331112	11631221	25411121
918	32422112	14411141	15411212
919	33331211	13511132	15411311
920	31513112	13511231	16321121
921	32422211	12611123	15412121
922	31513211	22611131	24511112
923	24241112	12611222	24511211
924	23332112	12611321	15421112
925	24241211	13521131	14512112
926	22423112	12612131	15421211
927	23332211	12621122	14512211
928	21514112	12621221	33611111

Annex B
(normative)

The default character set for Byte Compaction mode

B	C	B	C	B	C	B	C	B	C	B	C	B	C	B	C
0	NUL	32	space	64	@	96	`	128		160	NBSP	192	À	224	à
1	SOH	33	!	65	A	97	a	129		161	¡	193	Á	225	á
2	STX	34	"	66	B	98	b	130		162	¢	194	Â	226	â
3	ETX	35	#	67	C	99	c	131		163	£	195	Ã	227	ã
4	EOT	36	$	68	D	100	d	132		164	¤	196	Ä	228	ä
5	ENQ	37	%	69	E	101	e	133		165	¥	197	Å	229	å
6	ACK	38	&	70	F	102	f	134		166	¦	198	Æ	230	æ
7	BEL	39	'	71	G	103	g	135		167	§	199	Ç	231	ç
8	BS	40	(	72	H	104	h	136		168	¨	200	È	232	è
9	HT	41	)	73	I	105	i	137		169	©	201	É	233	é
10	LF	42	*	74	J	106	j	138		170	ª	202	Ê	234	ê
11	VT	43	+	75	K	107	k	139		171	«	203	Ë	235	ë
12	FF	44	,	76	L	108	l	140		172	¬	204	Ì	236	ì
13	CR	45	-	77	M	109	m	141		173	SHY	205	Í	237	í
14	SO	46	.	78	N	110	n	142		174	®	206	Î	238	î
15	SI	47	/	79	O	111	o	143		175	¯	207	Ï	239	ï
16	DLE	48	0	80	P	112	p	144		176	°	208	Ð	240	ð
17	DC1	49	1	81	Q	113	q	145		177	±	209	Ñ	241	ñ
18	DC2	50	2	82	R	114	r	146		178	²	210	Ò	242	ò
19	DC3	51	3	83	S	115	s	147		179	³	211	Ó	243	ó
20	DC4	52	4	84	T	116	t	148		180	´	212	Ô	244	ô
21	NAK	53	5	85	U	117	u	149		181	µ	213	Õ	245	õ
22	SYN	54	6	86	V	118	v	150		182	¶	214	Ö	246	ö
23	ETB	55	7	87	W	119	w	151		183	·	215	×	247	÷
24	CAN	56	8	88	X	120	x	152		184	¸	216	Ø	248	ø
25	EM	57	9	89	Y	121	y	153		185	¹	217	Ù	249	ù
26	SUB	58	:	90	Z	122	z	154		186	º	218	Ú	250	ú
27	ESC	59	;	91	[	123	{	155		187	»	219	Û	251	û
28	IS4/FS	60	<	92	\	124	|	156		188	¼	220	Ü	252	ü
29	IS3/GS	61	=	93	]	125	}	157		189	½	221	Ý	253	ý
30	IS2/RS	62	>	94	^	126	~	158		190	¾	222	Þ	254	þ
31	IS1/US	63	?	95	_	127	DEL	159		191	¿	223	ß	255	ÿ

NOTE This table corresponds to the character set defined in ISO/IEC 8859-1, with the addition of the control characters (byte values 00 – 31) defined in ISO/IEC 646:1991, International Reference Version.

Annex C
(normative)

Byte Compaction mode encoding algorithm

This base 256 to base 900 conversion converts six data bytes to five PDF417 data codewords. The conversion equation is:

$$\boldsymbol{b_5 \times 256^5 + b_4 \times 256^4 + b_3 \times 256^3 + b_2 \times 256^2 + b_1 \times 256^1 + b_0 \times 256^0}$$

$$\boldsymbol{= d_4 \times 900^4 + d_3 \times 900^3 + d_2 \times 900^2 + d_1 \times 900^1 + d_0 \times 900^0}$$

where b = data byte value as a decimal (0 to 255)

where d = data codeword

The following algorithm may be used for a base 256 to base 900 conversion.

1. Designate ***t*** as a temporary value
2. Calculate $\boldsymbol{t = b_5 \times 256^5 + b_4 \times 256^4 + b_3 \times 256^3 + b_2 \times 256^2 + b_1 \times 256^1 + b_0 \times 256^0}$
3. Calculate each codeword as follows:

```
For each data codeword $d_i = d_0 ... d_4$
BEGIN
    $d_i$  =  $t$ mod 900
    $t$    =  $t$ div 900
END
```

EXAMPLE:

Encode the Byte Compaction characters b_5 ... b_0 **{231, 101, 11, 97, 205, 2}**

Calculate the sum t using the decimal values of the six Byte Compaction characters:

t = 231 × 256^5 + 101 × 256^4 + 11 × 256^3 + 97 × 256^2 + 205 × 256^1 + 2 × 256^0

= 254 421 168 672 002

Calculate codeword 0

d_0 = 254 421 168 672 002 mod 900 = 302

t = 254 421 168 672 002 div 900 = 282 690 187 413

Calculate codeword 1

d_1 = 282 690 187 413 mod 900 = 213

t = 282 690 187 413 div 900 = 314 100 208

Calculate codeword 2

d_2	= 314 100 208 mod 900	= 208
t	= 314 100 208 div 900	= 349 000

Calculate codeword 3

d_3	= 349 000 mod 900	= 700
t	= 349 000 div 900	= 387

Calculate codeword 4

d_4	= 387 mod 900	= 387
t	= 387 div 900	= 0

The codeword sequence d_4 ... d_0 is 387, 700, 208, 213, 302

Annex D
(normative)

Numeric Compaction mode encoding algorithm

This base 10 to base 900 conversion converts groups of up to 44 consecutive numeric digits to 15 or fewer PDF417 data codewords.

The following algorithm may be used for a base 10 to base 900 conversion.

1. Designate ***t*** as a temporary value.
2. Set the initial value of ***t*** to be the group of up to 44 consecutive numeric digits, preceded by the digit 1.
3. Calculate each codeword as follows:

```
    For each data codeword $d_i = d_o \ldots d_{n-1}$
    BEGIN
        $d_i$   =   $t$ mod 900
        $t$     =   $t$ div 900
        If $t$ = 0, then stop encoding
    END
```

EXAMPLE:

Encode the fifteen digit sequence **000213298174000**

Prefix the numeric string with a 1 and set the initial value of

t = 1 000 213 298 174 000

Calculate codeword 0

d_0 = 1 000 213 298 174 000 mod 900 = 200

t = 1 000 213 298 174 000 div 900 = 1 111 348 109 082

Calculate codeword 1

d_1 = 1 111 348 109 082 mod 900 = 282

t = 1 111 348 109 082 div 900 = 1 234 831 232

Calculate codeword 2

d_2 = 1 234 831 232 mod 900 = 632

t = 1 234 831 232 div 900 = 1 372 034

Calculate codeword 3

d_3	=	1 372 034 mod 900	=	434
t	=	1 372 034 div 900	=	1 524

Calculate codeword 4

d_4	=	1 524 mod 900	=	624
t	=	1 524 div 900	=	1

Calculate codeword 5

d_5	=	1 mod 900	=	1
t	=	1 div 900	=	0

The codeword sequence d_5 ... d_0 is 1, 624, 434, 632, 282, 200

Annex E
(normative)

Error correction

Unlike PDF417, which has user-selectable levels of error correction, a MicroPDF417 symbol with a given number of rows and columns has a fixed amount of error correction. All of the specified versions of MicroPDF417 contain at least 28% error correction codewords (see Table 1). A symbol version giving a higher percentage of error correction codewords may be used where significant symbol damage or degradation is anticipated. It is, however, preferable to maintain symbol quality rather than to compensate for poor print quality by this means.

Annex F
(normative)

Tables of coefficients for calculating MicroPDF417 error correction codewords

Tables of coefficients are on the MicroPDF417 development diskette accompanying this specification, as a convenience to the reader. These tables list the coefficients for each of the values of *k* that are used by MicroPDF417. These coefficients can be derived, for any value of *k*, by means of the procedures given in 5.10 of this specification. The coefficients are given below, for all values of *k* used by MicroPDF417. Annex P provides a worked example of the calculation.

Coefficient table for *k* = 3

j	0	1	2
α_j	200	351	890

Coefficient table for *k* = 4

j	0	1	2	3
α_j	522	568	723	809

Coefficient table for *k* = 5

j	0	1	2	3	4
α_j	427	919	460	155	566

Coefficient table for *k* = 6

j	0	1	2	3	4	5
α_j	861	285	19	803	17	766

Coefficient table for *k* = 7

j	0	1	2	3	4	5	6
α_j	76	925	537	597	784	691	437

Coefficient table for *k* = 8

j	0	1	2	3	4	5	6	7
α_j	237	308	436	284	646	653	428	379

Coefficient table for *k* = 9

j	0	1	2	3	4	5	6	7	8
α_j	567	527	622	257	289	362	501	441	205

Coefficient table for *k* = 10

j	0	1	2	3	4	5	6	7	8	9
α_j	377	457	64	244	826	841	818	691	266	612

Coefficient table for *k* = 11

j	0	1	2	3	4	5	6	7	8	9	10
α_j	462	45	565	708	825	213	15	68	327	602	904

Coefficient table for *k* = 12

j	0	1	2	3	4	5	6	7	8	9	10	11
α_j	597	864	757	201	646	684	347	127	388	7	69	851

Coefficient table for *k* = 13

j	0	1	2	3	4	5	6	7	8	9	10	11	12
α_j	764	713	342	384	606	583	322	592	678	204	184	394	692

Coefficient table for *k* = 14

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
α_j	669	677	154	187	241	286	274	354	478	915	691	833	105	215

Coefficient table for *k* = 15

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
α_j	460	829	476	109	904	664	230	5	80	74	550	575	147	868	642

Coefficient table for *k* = 16

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
α_j	274	562	232	755	599	524	801	132	295	116	442	428	295	42	176	65

Coefficient table for *k* = 18

j	0	1	2	3	4	5	6	7	8
α_j	279	577	315	624	37	855	275	739	120
j	9	10	11	12	13	14	15	16	17
α_j	297	312	202	560	321	233	756	760	573

Coefficient table for *k* = 21

j	0	1	2	3	4	5	6	7	8	9	10
α_j	108	519	781	534	129	425	681	553	422	716	763
j	11	12	13	14	15	16	17	18	19	20	
α_j	693	624	610	310	691	347	165	193	259	568	

Coefficient table for *k* = 26

j	0	1	2	3	4	5	6	7	8	9	10	11	12
α_j	443	284	887	544	788	93	477	760	331	608	269	121	159
j	13	14	15	16	17	18	19	20	21	22	23	24	25
α_j	830	446	893	699	245	441	454	325	858	131	847	764	169

Coefficient table for *k* = 32

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
α_j	361	575	922	525	176	586	640	321	536	742	677	742	687	284	193	517
j	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
α_j	273	494	263	147	593	800	571	320	803	133	231	390	685	330	63	410

Coefficient table for *k* = 38

j	0	1	2	3	4	5	6	7	8	9	10	11	12
α_j	234	228	438	848	133	703	529	721	788	322	280	159	738
j	13	14	15	16	17	18	19	20	21	22	23	24	25
α_j	586	388	684	445	680	245	595	614	233	812	32	284	658
j	26	27	28	29	30	31	32	33	34	35	36	37	
α_j	745	229	95	689	920	771	554	289	231	125	117	518	

Coefficient table for *k* = 44

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
α_j	476	36	659	848	678	64	764	840	157	915	470	876	109	25	632
j	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
α_j	405	417	436	714	60	376	97	413	706	446	21	3	773	569	267
j	30	31	32	33	34	35	36	37	38	39	40	41	42	43	
α_j	272	213	31	560	231	758	103	271	572	436	339	730	82	285	

Coefficient table for *k* = 50

j	0	1	2	3	4	5	6	7	8	9	10	11	12
α_j	923	797	576	875	156	706	63	81	257	874	411	416	778
j	13	14	15	16	17	18	19	20	21	22	23	24	25
α_j	50	205	303	188	535	909	155	637	230	534	96	575	102
j	26	27	28	29	30	31	32	33	34	35	36	37	38
α_j	264	233	919	593	865	26	579	623	766	146	10	739	246
j	39	40	41	42	43	44	45	46	47	48	49		
α_j	127	71	244	211	477	920	876	427	820	718	435		

Annex G
(normative)

Text Compaction mode encoding algorithm

In Text Compaction mode, pairs of data characters are represented in a single codeword. The values assigned to the data characters are in the range 0 to 29 (i.e. base 30) as defined in Table 6. For each pair of base 30 values, the first or left value shall be designated the more significant value **h**, the other shall be designated the less significant value **l**.

The encoded PDF417 codeword is defined using the following formula:

$$d = h \times 30 + l$$

where: *d* is as defined in clause 4.

h and *l* are respectively the more and the less significant base 30 values

The formula shall also apply to the base 30 values for shifts and latches within the Text Compaction mode. Appropriate latch and shift values shall be used between sub-modes. If the encoding of the character sequence does not result in an even number of base 30 values, see 5.4.2.4 for the specific mechanism to use.

The following example illustrates how compaction is achieved in Text Compaction mode.

EXAMPLE:

Data to be encoded: **MicroPDF417**

Table G.1 — Example of Text Compaction encoding

Character Pairs	*h*	*l*	*h* × 30 + *l*	Codeword
M **ll**	12	27	12 × 30 + 27	387
i c	8	2	8 × 30 + 2	242
r o	17	14	17 × 30 + 14	524
TC Mode Latch				900
P D	15	3	15 × 30 + 3	453
F **ml**	5	28	5 × 30 + 28	178
4 1	4	1	4 × 30 + 1	121
7 **ps**	7	29	7 × 30 + 29	239

NOTE 1 **ll** (latch to lower sub-mode) is used to switch to encode the lower-case characters

NOTE 2 **ml** (latch to mixed sub-mode) is used to switch to encode the numeric characters

NOTE 3 The Text Compaction mode latch 900 is used to re-initialise encoding to Text Compaction Alpha sub-mode (see 5.4.2.5) to encode the upper-case characters

NOTE 4 **ps** is used as a pad value in this example, other shift and latch values can be used (see 5.4.2.4)

The data **MicroPDF417** is represented by codewords 387, 242, 524, 900, 453, 178, 121, 239.

Annex H
(normative)

Structured Append MicroPDF417 symbols

H.1 Structured Append overview

Structured Append provides a standard mechanism for creating a distributed representation of files too large to be represented by a single MicroPDF417 symbol. Structured Append symbols differ from ordinary MicroPDF417 symbols in that they contain additional control information in a Structured Append Control Block (identical in structure to the Macro PDF417 Control Block defined in the PDF417 specification (ISO/IEC 15438)).

Using Structured Append, large files are split into several file segments and encoded into individual symbols. The Control Block defines the file ID, the concatenation sequence and optionally other information about the file. The Structured Append decoder uses the Control Block's information to reconstruct the file correctly independently of symbol scanning order. When multiple Structured Append symbols appear on the same label or page, their layout should ensure that only one symbol at a time appears within a scanner's field of view.

H.2 Structured Append syntax

Each Structured Append symbol shall encode a Structured Append Control Block containing control information. The Control Block begins with the Structured Append marker codeword (928). The Control Block follows the data block with which it is associated, and the number of codewords in the control block is counted as data. The beginning of the error correction codewords identifies the end of the Control Block. The Control Block shall contain at least the two mandatory fields: a segment index and file ID. It may also contain a number of optional fields, as described in Annex H.2.3.

NOTE A symbol containing no user data, other than a Structured Append Control Block, is a valid symbol.

Figure H.1 illustrates the position of the Control Block in a Structured Append symbol.

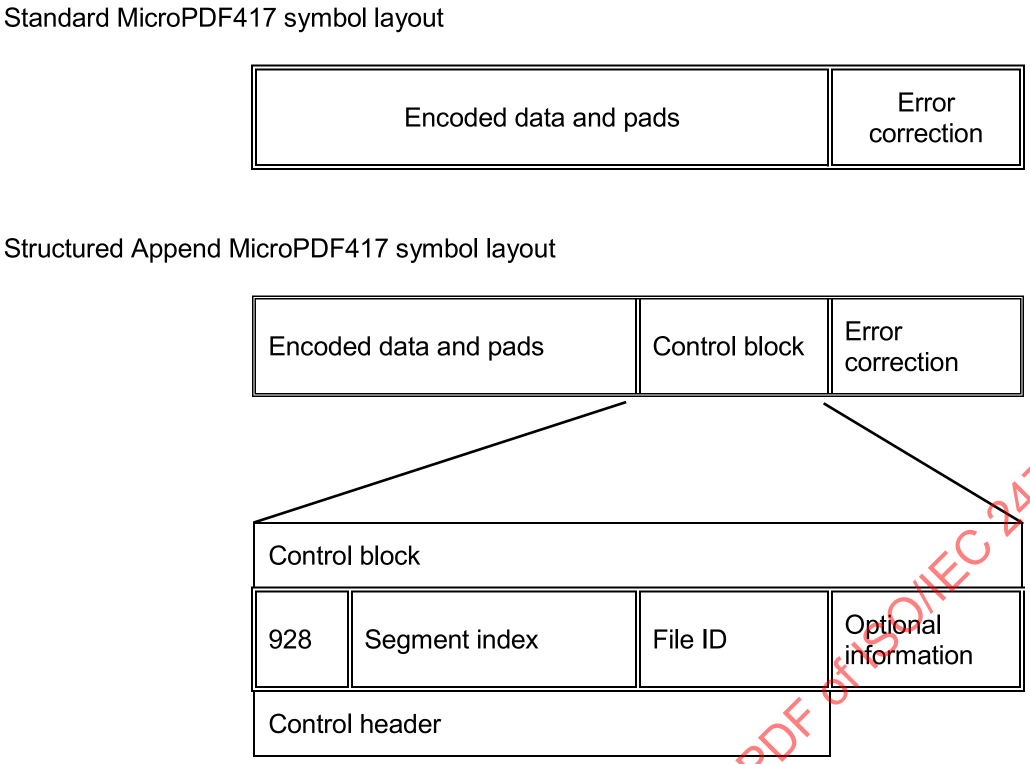


Figure H.1 — MicroPDF417 symbol layouts

H.2.1 The segment index

In Structured Append, each symbol represents a segment of the whole file. To reconstruct the whole file, the segments need to be placed in the correct order. Control information in the Control Block facilitates this reassembly process. For a file divided into a set of j Structured Append symbols, the segment index field in each symbol's Control Block contains a value between 0 and *j* - 1, corresponding to the relative position of that symbol's content within the distributed representation.

The segment index field is two codewords in length and is encoded using Numeric Compaction mode as defined in 5.4.4. The segment index value shall be padded with leading zeros to five digits before Numeric Compaction shall be applied, and the switch to Numeric Compaction shall not require an explicit mode latch (codeword 902). The largest allowed value in the segment index field is 99 998. Thus, up to 99 999 Structured Append symbols may comprise the distributed representation of a data file.

NOTE This translates to a capacity of nearly 14 million bytes of data in Byte Compaction mode, or 24 million characters in Text Compaction mode, or over 35 million characters in Numeric Compaction mode.

H.2.2 File ID field

For each related Structured Append symbol, the file ID field contains the same value. This ensures that all re-assembled symbol data belongs to the same distributed file representation. The file ID is a variable length field which begins with the first codeword following the segment index and extends to the start of the optional fields (if present) or to the end of the Control Block (if not).

Each codeword in the file ID can have a value between 0 and 899, effectively making the file ID a series of base 900 numbers. Each codeword of the series is transmitted as the 3-digit ASCII representation of its decimal value.

NOTE The effectiveness of the file identification scheme is influenced by both the length of the file ID field and the suitability of the algorithm used to generate its value.

H.2.3 Optional fields

Optional fields may follow the file ID. Each optional field begins with a specific tag sequence and extends until the start of the next optional field (if present) or the end of the Control Block (if not). The tag sequence consists of codeword 923 followed by a single codeword field designator. In each optional field, data following the tag sequence has a field-specific interpretation. Empty optional fields shall not be used. Table H.1 shows the correspondence between currently defined field designators and optional field contents. Each optional field begins with an implied reset to the compaction mode shown in the table and with an implied reset to ECI 000002 (or GLI 0 for encoders complying with early PDF417 standards). ECI escape sequences and mode latches and shifts may be used, but only in the optional fields initially in Text Compaction mode.

These fields shall always represent global file attributes and so need not be present in the Control Block of more than one Structured Append symbol within the distributed file representation, with the exception of the segment count field, as described below. The segment which contains these fields is defined by the specific encoder implementation. If a particular field is to appear in more than one segment, it shall appear identically in every segment. There is no required order for the optional fields.

Table H.1 — Structured Append optional field designators

Field designator	Byte value transmitted	Contents	Initial compaction mode	Fixed compaction mode	Total number of codewords
0	48	File Name	Text Compaction	N	Variable
1	49	Segment Count	Numeric Compaction	Y	4
2	50	Time Stamp	Numeric Compaction	Y	6
3	51	Sender	Text Compaction	N	Variable
4	52	Addressee	Text Compaction	N	Variable
5	53	File Size	Numeric Compaction	Y	Variable
6	54	Checksum	Numeric Compaction	Y	4

NOTE 1 A ‘Y' in the ‘Fixed Compaction Mode' column means that no ECIs and no compaction mode latches and shifts are allowed in that field.

NOTE 2 The totals shown in the last column include the two-codeword tag sequence.

As shown in Table H.1, all optional fields use standard MicroPDF417 high-level encoding. At the beginning of each field, the default mode in effect shall be defined by Table H.1, regardless of mode shifts and latches earlier in the symbol.

Specific construction of optional fields shall be as follows:

- The segment count field (identifying the total number of Structured Append symbols in the distributed file) can contain values from 1 to 99 999 and shall be encoded as two codewords. If the optional segment count field is used, that field shall appear in every segment.

- The time stamp field shall be interpreted in Numeric Compaction mode. It indicates the time stamp on the source file expressed as the elapsed time in seconds since 1970:01:01:00:00:00 GMT (i.e. 00:00:00 GMT on 1 January 1970). Using this format, four codewords can encode any date over the next two hundred centuries.

- The file size field contains the size in bytes of the entire source file.
- The checksum field contains the value of the 16-bit (2 bytes) CRC checksum using the CCITT-16 polynomial $x^{16} + x^{12} + x^5 + 1$ computed over the entire source file.

NOTE 1 The file size and checksum shall be calculated from the original source file, prior to the addition of any ECI escape sequences for Extended Channel Interpretation encoding. This implies that, if the receiver is to verify the checksum after reception, the original source file must be reconstructed verbatim. This requires, for the purposes of this optional checksum verification only, that no user-selectable or optional transformations of the byte stream be performed, even if these would normally be done in ECI decode processing.

NOTE 2 If the CRC is used, the calculation may be performed either before the data is sent to the printer, or in the printer, based on the capabilities of the printer.

Field designator values greater than 6 are not currently defined. However, MicroPDF417 decoding equipment shall decode and transmit any optional fields encountered with a field designator of 7 to 9 (byte 55 to 57) or A to Z (byte 65 to 90) by treating the field's data as being initially in Text Compaction mode and being variable length.

H.2.4 Structured Append terminator

The Control Block in the symbol representing the last segment of a Structured Append file contains a special marker, consisting of the codeword 922 at the end of the Control Block. The Control Block for every other symbol shall end after any optional fields with no special terminator.

H.3 High level encoding considerations

While Structured Append provides a mechanism for logically associating a set of symbols, it is important to realise that, with respect to MicroPDF417 high-level encoding, each symbol shall remain a distinct entity. Thus, the scope of a mode switch shall be confined to the symbol in which it occurs. Each symbol shall implicitly begin in the Byte Compaction mode (see 5.4.3).

The two mandatory fields are encoded as follows: the segment index is encoded in Numeric Compaction mode and the file ID field is encoded as a sequence of base 900 numbers.

In the context of a Control Block optional field, the compaction modes indicated in Table H.1 shall supersede the mode currently set by the mode identifier codewords within the data codeword region of the symbol. The scope of the current ECI(s), however, skips over the Structured Append Control Block to the start of the next Structured Append symbol. Each Structured Append Control Block field begins with an implied reset to ECI 000002 (or GLI 0 for encoders complying with the early PDF417 standards). It shall also be possible to set a different ECI within an optional Text Compaction mode Structured Append Control Block field, for example, to represent properly a Greek addressee's name. The ECI escape sequence may be placed in any permitted position (see 5.5.3) after the tag codeword (923).

H.4 Encodation example

To illustrate the encodation of a Structured Append Control Block, the following example is used:

A Structured Append series encodes a total of 400 bytes of user defined data in four MicroPDF417 symbols (or file segments). Other 'header' data to be encoded are:

- File ID = $17_{\text{base 900}}$ $53_{\text{base 900}}$
- Segment count to be used
- Sender: CEN BE
- Addressee: ISO CH

NOTE The segment count, sender and addressee are three optional fields selected by the user.

On the assumption that the encoder places optional fields in the first symbol, the encodation of the Structured Append Control Block would be as follows for that symbol.

... [last data codeword] $[928]_A$ [111] $[100]_B$ [017] $[053]_C$ [923] $[001]_D$

$[111]$ $[104]_E$ [923] $[003]_F$ [064] [416] $[034]_G$ [923] $[004]_H$ [258] [446] $[067]_I$

[first error correcting codeword]...

The last symbol of four would have the following Structured Append Control Block:

... [last data codeword] $[928]_A$ [111] $[103]_B$ [017] $[053]_C$

[923] $[001]_D$ [111] $[104]_E$ $[922]_J$ [first error correcting codeword]

where: A = Structured Append Marker Codeword

B = File Segment ID

File segments are numbered from 0 to j - 1, and are encoded using Numeric Compaction

1st Segment = 00000 = codewords 111, 100

4th Segment = 00003 = codewords 111, 103

C = File ID to base 900

D = Tag for segment count field

E = Segment count

F = Tag for sender field

G = Sender field encoding CEN BE

H = Tag for addressee field

I = Addressee field encoding ISO CH

J = Structured Append Terminator

H.5 Structured Append and the Extended Channel Interpretation protocol

The symbology-independent Extended Channel Interpretation (ECI) protocol was developed after PDF417 was specified as a symbology. PDF417 supported its own Global Label Identifier (GLI) system, the precursor and basis of the ECI protocol, from the first publication of the symbology specification in 1994. Therefore, previous 'GLI' implementations have to be taken into account. There are two different conditions which need to be taken into account:

- GLI 0 and 1 which were the only interpretations specified in the original AIM USA (1994) and AIM Europe (1994) PDF417 specifications. These are equivalent to ECI 000000 and ECI 000001. The precise rules for Structured Append are defined in Annex H.5.1

- All other ECI assignments, whose usage with Structured Append is defined in Annex H.5.2

H.5.1 Structured Append with ECI 000000 and 000001 (GLI 0 and 1)

As GLIs were intrinsically part of the original AIM USA (1994) and AIM Europe (1994) PDF417 specifications, it is logical to have a GLI encoder and Structured Append encoder combined in one unit. These specifications called for an implied 'reset-to-GLI 0' logic at the beginning of the second and subsequent Structured Append symbols, thus every symbol is expected to start at the default interpretation. For GLI 0 and 1 (equivalent to ECI 000000 and 000001), this has no inherent effect on the encodation. However for some complex ECIs, the reset-to-GLI 0 logic is difficult to implement in a symbology-independent manner.

Encoding software compliant with the definitions of Macro PDF417 and GLI 0 and 1 in the original AIM USA (1994) and AIM Europe (1994) PDF417 specifications is completely suitable for pre-existing applications. So too are pre-existing applications of user defined GLIs (now called ECIs) because by definition the domain of the system is constrained.

All ECIs numbered 000002 or higher shall not be defined with the reset-to-GLI 0 logic. Therefore, MicroPDF417 symbols shall not mix ECI 000000 and 000001 with any higher numbered ECI (except in closed systems).

H.5.2 Structured Append and other ECIs

An ECI encoder could be symbology independent and create a byte stream as input to a MicroPDF417 symbology encoder. The ECI encoder should behave as if there is a single data stream, irrespective of the size of the file. Thus, an ECI once invoked would persist across segments until another ECI or the end of the encoded data. This is essential if, for example, the ECI assignment represents an encryption scheme, where returning to GLI 0 would not be appropriate.

Structured Append encoders compliant with this standard need not encode the prevailing ECI at the beginning of subsequent Structured Append symbols.

NOTE There may need to be some iteration to produce a logical end-of-symbol encodation, for example: Numeric Compaction mode shall not straddle two segments, but two separate Numeric Compaction blocks can be encoded at the end of one symbol and at the beginning of the next. These conditions are related to Structured Append and High Level Encoding *(*see Annex H.3*)* and not Structured Append and ECIs.

H.6 Structured Append data transmission

The transmission of Structured Append Control Block information shall be treated in a similar manner to that of interpretative ECIs. The symbology-independent ECI protocol is defined below; the original PDF417 protocol is defined in Annex M. Although the Structured Append Control Block is encoded at the end of the symbol's data, it is transmitted before the symbol's data when using the ECI protocol.

Three codewords (922, 923 and 928) signal the encodation of a Structured Append Control Block or one of its constituent parts. Decoding is as follows:

1. If the Structured Append marker codeword (928) begins the sequence:

    a. Codeword 928 is transmitted as the escape sequence 92, 77, 73, which represents '\MI' in the default interpretation.

    b. The next two codewords identify the segment index. These are encoded in Numeric Compaction mode and decode as a 5-digit number in the range 00000 to 99998.

    c. The next codewords encode the file ID field, which shall be the same for all related Structured Append symbols. The end point of the file ID field is either codeword 922, codeword 923, or the end of the encoded data in the symbol. Each codeword is converted to a 3-digit number in the range 000 to 899 (i.e. the codeword number) and transmitted as three byte values (in the range decimal 48 to 57) following the escape header: 92, 77, 70, which represents '\MF' in the default interpretation.

2. If the Structured Append sequence tag codeword (923) begins the sequence:

   a. Codeword 923 is transmitted as the escape sequence 92, 77, 79, which represents '\MO' in the default interpretation.

   b. The next codeword represents one of the optional field designators in Table H.1 transmitted as a single byte representing the ASCII value of the designator.

   c. The next codewords carry the data content of the optional field designator. The end point of the optional field is either codeword 922, codeword 923, or the end of the encoded data in the symbol. The intervening codewords should be converted according to the decode rules of the relevant compaction mode defined in Table H.1. The resultant data may be variable length.

3. If the Structured Append Terminator (codeword 922) is identified, the escape sequence 92, 77, 90, which represents '\MZ' in the default interpretation, shall be transmitted.

4. At the end of the Structured Append Control Block, as defined by the end of encoded data in the symbol, the escape sequence 92, 77, 89, which represents '\MY' in the default interpretation, shall be transmitted.

NOTE This escape sequence is not explicitly encoded in the symbol.

All the Structured Append Control Block fields for a symbol (segment) shall be transmitted as a single block starting with \MI... and ending with \MY. The transmission of the Structured Append Control Block shall precede the transmission of the remainder of the encoded file segment, even though it is encoded at the end of the symbol.

EXAMPLE:

The Structured Append Control Header of the first symbol, Segment Index = 0, with a File ID (100, 200, 300) would be encoded in the symbol as the codeword sequence:

[928] [111] [100] [100] [200] [300]

It would be transmitted as:

Data transmission (byte):

92, 77, 73, 48, 48, 48, 48, 48, 92, 77, 70, 49, 48, 48, 50, 48, 48, 51, 48, 48, 92, 77, 89

ASCII interpretation:

\MI00000\MF100200300\MY

As the Structured Append symbols are scanned, the de-packetising function reconstructs the original message, bearing in mind that the symbols may be scanned out of sequence. If the system is operating in buffered mode, the de-packetising function is in the decoder; if operating in unbuffered mode it is in the receiving system.

Decoders should provide a decoder-specific means whereby the processing of a given Structured Append file ID may be aborted, thus allowing the decoder to begin processing a new File ID. This is necessary to prevent a deadlock condition should one or more symbols of a given File ID be missing or undecodable.

H.6.1 Operating in buffered mode

In buffered mode, de-packetising shall be performed in the decoder/reader. Depending on the equipment configuration it will either:

- send the reconstructed data with no Structured Append Control Block.

or

- send one Structured Append Control Block (which itself may have been reconstructed to include all optional fields included in any symbols) to precede the entire encoded message. The resulting Structured Append Control Block shall have its Structured Append Index field set to 0 and shall include the Structured Append end-of-file field (in effect, to mark the entire reconstructed message as the first and only Structured Append segment of the pseudo-series).

H.6.2 Operating in unbuffered mode

In unbuffered mode, de-packetising shall be performed in the receiving system. Each transmitted Structured Append Control block shall represent all of the required and optional fields actually encoded in the symbol.

When configured in unbuffered mode, a decoder may optionally be configured not to require successive symbols to be of the same File ID. This procedure would only be appropriate if the decoder is configured to transmit the Structured Append Control Block to the receiving system, and this receiving system is designed to monitor the File ID portion of the Control Block to determine when the entire file has been processed. Symbols with a different File ID or no File ID (e.g. a single symbol not part of a Structured Append set) shall be dealt with as determined by the receiving system.

To facilitate checking that all symbols in a Structured Append set are received in an unbuffered operation, the optional Segment Count field should be used whenever possible as part of the encoded Structured Append Control Block.

H.6.3 Reset-to-GLI 0 transmissions

Because the 1994 USS for PDF417 defined GLI 0 and GLI 1 to have rules slightly different from the rules for ECIs, a reader compliant with this International Standard must, in two situations, transmit extra escape sequences when transmitting symbols containing explicit GLI 1 invocations:

1. The decoder shall transmit either a GLI 0 escape sequence or an ECI 000000 escape sequence (depending upon which transmission protocol it is programmed to use) after transmitting the data of any Structured Append symbol whose data ends in a GLI 1 (ECI 000001) interpretation.

2. The decoder shall transmit a GLI 1 (ECI 000001) at the start of each variable length optional field encoded in Text Compaction mode in the Structured Append Control Block, if the data preceding that field ends in a GLI 1 (ECI 000001) interpretation.

This requirement applies whether operating in buffered or unbuffered mode, and whether the decoder is programmed to transmit using either the ECI protocol, or the original PDF417 transmission protocol.

Annex I
(normative)

Testing MicroPDF417 symbol quality

I.1 Overview of methodology

As specified in 5.14.4, the quality of MicroPDF417 symbols is evaluated according to the methodology defined in ISO/IEC 15415 for the assessment of multi-row symbologies with cross-row scanning ability.

In summary, MicroPDF417 symbols are graded in respect of the following:

— Analysis of the scan reflectance profile applied to the Row Address Patterns only (see I.2).

— Codeword Yield, applied to the data and error correction codewords only, which measures the efficiency with which linear scans can recover data from the symbol. The Codeword Yield is the number of validly decoded codewords expressed as a percentage of the maximum number of codewords that could have been decoded, i.e. the number of data columns in the symbol multiplied by the number of "qualified" scans (after adjusting for tilt).

— Unused Error Correction, applied to the data and error correction codewords only, which expresses the number of errors and erasures as a function of the error correction capacity of the symbol .

— Codeword print quality, applied to the data and error correction codewords only, which enables the Decodability, Defects and Modulation parameters of scan reflectance profiles covering the entire data region of the symbol to be graded; these grades are then modified to allow for the effect of error correction in masking less than perfect attributes of the symbol that influence symbol quality.

— The Decode parameter tests, on a Pass/Fail basis, whether the symbol has all its features sufficiently correct to be readable using the reference decode algorithm in Annex J.

The overall symbol grade shall be the lowest of each of the above grades.

I.2 Test scans for scan reflectance profile

Ten target test scan lines are first selected evenly spaced across the symbol. For instance for a five row symbol, two target test scan lines will be assigned to each row.

The analysis of the signal representing a MicroPDF417 Row Address Pattern (see 5.2.5) shall make allowance for the fact that, in the vicinity of the one module within each Row Address Pattern that differs from the row above, or in the vicinity of the module that differs from the row below, signal crosstalk may be present from the Row Address Pattern either above or below the one nominally being scanned. Note that, if the effective optical aperture of the verifying equipment is appropriate for the X- and Y-dimensions of the symbol, significant crosstalk from both adjacent rows will not occur on the same measurement scan line. Thus, a single poor measurement (such as unusually low decodability, compared with the rest of the Row Address Pattern), if at either of the two possible crosstalk locations within a given Row Address Pattern, shall be discarded when computing the grade for that scan.

Each target test scan line to be evaluated shall contain at least two Row Address Patterns (and may contain three, if a Center Row Address Pattern is present); if such scans cannot be obtained, then the symbol shall receive a grade of 0 on this measure. Each of these Row Address Patterns provides 5 E-distances that can be measured (except that the outermost E-distance of the Left Row Address Pattern shall not be used for measurement purposes). For any specific Row Address Pattern to be measured, two subsets of these E-distances shall be enumerated. One subset excludes the one or two of the E-distances that could be distorted by the Row Address Pattern above the one being assessed, and the other subset excludes the E-distances that could be distorted by the Row Address Pattern below the one being assessed. For each Row Address Pattern measurement, the subset yielding the better grade shall be used.

Because of the possibility of a scan line crossing the corner of a RAP edge at the point where it differs from the adjacent RAP, possibly creating an extraneous transition through the global threshold and resulting in an incorrect number of elements being detected, the verifier shall ignore RAP defects at either of the predicted adjacent RAP edge changes. This requires that the codeword adjacent to the RAP must be decoded so that the two RAP edge changes can be predicted. If the scan line decodes neither the RAP nor its adjacent codeword, the scan shall be ignored. If there is damage detected to the RAP that was not at the RAP edge change, the target test scan line shall be graded 0 and that target test scan shall be considered complete. Once all of the target test scan lines have been assigned a grade, the grading of the scan reflectance profile is complete. However, if the RAP and adjacent codeword of one or more target scan lines are never decoded after overscanning according to the codeword yield criteria in ISO/IEC 15415, these particular target test scan line(s) shall be graded 0.

Annex J
(normative)

Reference decode algorithm for MicroPDF417

This Annex describes the reference decode algorithm used in the computation of decodability when assessing the symbol quality using the method described in ISO/IEC 15415.

When assessing symbol quality through the use of this reference decode algorithm, a MicroPDF417 symbol shall be decoded in a series of scan lines running across the symbol that cross at least one Row Address Pattern, but not necessarily row by row. It is possible to decode the symbol if the scan line crosses two or more rows by using the cluster number. The decoding of symbol character bar-space sequences, for the Row Address Patterns as well as the PDF417 symbol characters, shall be achieved by using edge to similar edge (e) measurements.

The MicroPDF417 symbol shall be decoded in three phases:

- Initialization - to establish the symbol matrix.
- Filling the matrix
- Interpretation.

Phases 1 and 2 both perform line-by-line decoding using the reference line-decode algorithm, but with slightly different requirements (as described in J.1 and J.2). The first two phases also differ in that this reference decode algorithm does not require that any decoded codewords be stored in the matrix until phase 1 has been completed (although commercial implementations may choose to store the decoded results of phase 1, at the expense of additional algorithmic complexity).

Many stacked-linear symbologies (such as PDF417, MicroPDF417, and Stacked Channel Code) do not specify horizontal separator bars between the stacked rows, and thus do not provide a physical mechanism to prevent distortions and errors in the scan data near row boundaries. Practical implementations of linear-scanning readers of stacked 2D symbologies will include some method of "voting," or counting of successful scan results, to ensure that many correctly-decoded scans of a given codeword or area of the symbol will not be discarded in favour of the results of a single misdecoded scan. Voting algorithms are applicable during both Phase 1 and Phase 2 of the MicroPDF417 decode algorithm.

J.1 Phase 1: Initialization

A sufficient number of line decodes (see J.4 below) shall be performed at the start of the decode process to establish the MicroPDF417 symbol's “family”, that is, the number of columns ***c*** of the symbol, and the RAP rotation ***D*** of its Row Address Patterns. This is accomplished by collecting scans until either:

a) at least three more rows have been identified from the same family than have been identified from any other single family, or

b) at least three different rows from one family have been scanned and the number of successful scans from that family outnumbers the total number of successful scans from all other families.

In the absence of misdecodes, both of these criteria will be met as soon as three different rows of the symbol have been successfully decoded. During this phase, each useable scan line must contain at least two valid Row Address Patterns, in order to establish ***D***. For each such scan to be considered useable, at least one valid PDF417 symbol character must be adjacent to at least one of these patterns; furthermore, the values of these two address patterns, and the cluster number(s) of the intervening symbol character(s), must be consistent with each other. If during Phase 1, successful candidates are found from the same family, but indicate rows that are inconsistent with a single symbol version of that family, then these are counted separately (but any row shared by two versions increments the count for both versions). Phase 1 concludes as soon as the counts for one of these versions (including its shared rows) meet either of the above criteria, compared to any other families decoded.

If a valid center Row Address Pattern was detected as one of the decoded patterns, then the specific combination of decoded Row Address Patterns is sufficient to determine that the scan crossed the left and/or right halves of a three- or four-column symbol rather than a one- or two-column symbol (by checking for a valid pairing of patterns within Table 12). When both a left and a right Row Address Pattern were decoded, but no center pattern was decoded, then the relative physical distance between the left and right patterns indicates the number of columns. Specifically, the distance between the patterns will (before allowing for possible acceleration of the scanning beam) be either equal to the width of a symbol character (for a one-column symbol), two symbol characters (for a two-column symbol), 3.6 symbol characters (for a three-column symbol), or 4.6 symbol characters (for a four-column symbol).

An individual scan that contains a center and right Row Address Pattern, but that contains neither a left Row Address Pattern nor a decodable PDF417 character in the first column position, could correspond to either a three or a four-column symbol. If after several scans a left Row Address Pattern has yet to be decoded, but the requirements of Phase 1 have otherwise been met, the decoder shall proceed to Phase 2, using the procedures described below, and begin storing decoded codewords under the assumption of a four-column symbol (i.e., $c = 4$). The determination whether $c = 3$ or 4 will be made during Phase 2.

After the RAP rotation D and the number of columns c have thus been determined, a matrix shall be established which has a number of columns equal to the number of data columns of the symbol being decoded, and which has 52 rows (so that codewords can be placed in the matrix based upon their address pattern numbers, even before the actual row numbering for the symbol version being scanned has been determined).

J.2 Phase 2: Filling the matrix

The following procedure shall be used to fill the matrix of 52 rows by 1, 2, 3, or 4 columns established in the initialization phase. During this phase, after each scan line has been processed (see J.4), decoded codewords are placed into this matrix at a matrix row address (addresses defined such that the first row of the matrix is row 1, not row 0) equal to the symbol's left Row Address Pattern number (which, using Tables 10a to 10c, may be calculated from the decoded data of that scan, even if the left Row Address Pattern was not actually decoded). During this second phase, it is possible to utilize scans that contain only one RAP (exception: 2 RAPs may still be needed, if the 3- vs 4-column decision is still pending, because if under this condition a new scan were to contain only a left RAP and the single data codeword next to it, then the decoder would not know whether to place that codeword in the first or the second column of the matrix).

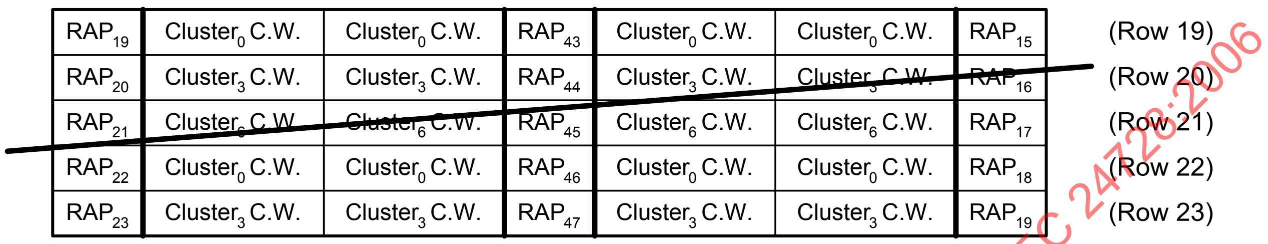
1. Initialize multiple erasure count variables v_i each equal to $r_i \times c$, where each r_i corresponds to the number of rows of one of the defined MicroPDF417 versions from the family determined in Phase 1 (i.e., with exactly c data columns and the RAP rotation D).

2. For each scan, attempt to decode as many codewords as the number of columns c of the matrix, following the procedures of J.4

3. Valid decode results are placed in the row of the matrix determined by the Row Address Pattern numbers and the cluster values. As an example of a voting mechanism that can be used in this phase, an up-down counter, initially set to zero, is associated with each matrix position. When a codeword is decoded for a matrix position whose count is zero, the codeword is stored at that position and the count is incremented by one. When a codeword is decoded for a position with a non-zero count, the count is incremented if the new codeword matches the stored codeword, otherwise the count is decremented.

If row crossing occurs, the cluster number shall be used to interpolate the correct Row Address Pattern number for each individual valid codeword.

EXAMPLE A left-to-right scan of a 4-column by 44-row symbol is shown below; the scan entered the symbol on the boundary between left Row Address Patterns 21 and 22. The left Row Address Pattern was decoded as 22. Although there are 4 columns in the matrix, this scan line has only three decodable codewords because it did not remain entirely in the one row for the full scan; however, the position of the missing codeword is known from element timings. The clusters of the symbol characters were decoded as follows: 6, 6, unknown, 3.



RAP_{19}	$Cluster_0$ C.W.	$Cluster_0$ C.W.	RAP_{43}	$Cluster_0$ C.W.	$Cluster_0$ C.W.	RAP_{15}	(Row 19)
RAP_{20}	$Cluster_3$ C.W.	$Cluster_3$ C.W.	RAP_{44}	$Cluster_3$ C.W.	$Cluster_3$ C.W.	RAP_{16}	(Row 20)
RAP_{21}	$Cluster_6$ C.W.	$Cluster_6$ C.W.	RAP_{45}	$Cluster_6$ C.W.	$Cluster_6$ C.W.	RAP_{17}	(Row 21)
RAP_{22}	$Cluster_0$ C.W.	$Cluster_0$ C.W.	RAP_{46}	$Cluster_0$ C.W.	$Cluster_0$ C.W.	RAP_{18}	(Row 22)
RAP_{23}	$Cluster_3$ C.W.	$Cluster_3$ C.W.	RAP_{47}	$Cluster_3$ C.W.	$Cluster_3$ C.W.	RAP_{19}	(Row 23)

Figure J.1 — Schematic showing a scan line crossing rows

Using matrix notation of (*row*, *column*), where *row* = left Row Address Pattern number (from 1 to 52) and where *column* is numbered from 1 to 4, the codewords are filled in the positions:

(21, 1), (21,2), (unknown), (20, 4).

NOTE Because its cluster number was 6, rather than 0, the first codeword was placed at matrix address 21, even though the left row address was decoded as 22.

4. As the matrix is being filled, one or more of the erasure counts v_i shall be reduced by one for each valid codeword placed in a row that is a member of one or more of the versions of this MicroPDF417 family.

5. If Phase 2 was entered without having decoded any left Row Address Patterns (i.e. assuming a four-column symbol), then once three or more left Row Address Patterns have been decoded in a position that indicates a three-column symbol, the decoder shall decrement the erasure counts v_i accordingly, and relocate the codewords of the last three columns of the matrix to its first three columns.

6. Error recovery may be attempted as soon as the number of unknown codewords (for any of the erasure counts v_i) satisfies the equations in 5.7.2 of the MicroPDF417 specification (with $v_i = v = L$ and $f = 0$). If at this point it is still true that fewer than three left Row Address Patterns were ever seen (i.e., the matrix was filled by assuming a four-column symbol), then the decoder shall adjust the erasure counts v_i to correspond to a three-column symbol, only if less than three data codewords were placed in the first column of the matrix. If error recovery fails, then more codewords shall be collected.

For more details on error detection and correction see Annex K.

J.3 Phase 3: Interpretation

Beginning from an initial state of the Byte Compaction mode 901, the data codewords shall be interpreted according to the compaction mode that resulted from initialization, as modified by any other compaction mode switches. If the first codeword is less than 900, then it shall be interpreted as if it were a three-codeword sequence representing a Transformation ECI whose number is (000900 + d_{n-1}).

J.4 Reference line-decode algorithm

The requirements for successful processing of a scan line differ slightly, depending upon whether the decoding process is in its first or its second phase; these requirements were defined in the phase 1 and 2 descriptions. During either phase, every scan shall contain one or more symbol characters in the data region, and at least one of these data region characters shall be adjacent to a left or right Row Address Pattern. During either phase, a decodable scan line may cross more than one row. The algorithm contains the following steps to decode the line:

1. Find a candidate pairing of a Left or Right Row Address Pattern adjacent to a PDF417 symbol character. A potential candidate consists of a sequence of 14 bars and spaces (starting with either a bar or a space) where either the first 6 elements have a nominal 10 : 17 ratio compared to the next 8, or the first 8 elements have a nominal 17 : 10 ratio compared to the next 6. To confirm a valid candidate, decode the selected 6 elements as a Row Address Pattern (using the steps of J.4.1 below), and decode the selected 8 elements as a PDF417 symbol character (using the steps of J.4.2 below). Assume a forward scan if the candidate string begins with a bar, and a reverse scan if it begins with a space; confirm this assumption using the fact that the first bar of a MicroPDF417 symbol will always be 2*X* or greater on a forward scan, but will be 1*X* on a reverse scan (if misinterpreted as a forward scan). Confirm the presence of the Address Pattern's adjacent quiet zone, and for a right pattern, the single-module stop bar as well (by taking an extra edge-to-edge measurement, which must normalize as 1 module larger than the rightmost space of the right RAP). Use the information in Tables 10a to 10c to reject scans containing inconsistent Row Address Pattern numbers.

2. In processing the entire scan line, more than one valid candidate pattern/character pairing may be found, possibly at overlapping locations. For each candidate (i.e., for each Row Address Pattern adjacent to a PDF417 symbol character) found using the procedures of step 1 above:

   a) During Phase 1, when the number of columns is unknown, attempt to decode in parallel both an additional symbol character and a second Row Address Pattern at every valid starting position (relative to the address pattern/data codeword pair found in step 1 above), until a second Row Address Pattern, consistent with the results of step 1, is confirmed. If no second Row Address Pattern was found during this Phase 1 processing, discard the candidate. If the pair of Row Address Patterns included a center Row Address Pattern, then attempt to decode the additional symbol character(s), and the remaining Row Address Pattern, of the symbol.

   b) During Phase 2, when the number of columns is known, attempt to decode as many additional symbol characters as is appropriate for the number of data columns in the matrix; also, attempt to decode the remaining Row Address Pattern(s) at the appropriate location(s) for this number of data columns.

   c) In performing either a) or b), use the scan direction derived from step 1 above. Decode the Row Address Pattern bar-space sequences as per Annex J.4.1; decode the symbol character bar-space sequences as per Annex J.4.2.

   d) In performing either a) or b), if two or three Row Address Patterns were found within the candidate string, check that their pattern numbers are consistent with each other (using Tables 10a to 10c). The decoded pattern numbers of two neighboring RAPs may each vary by plus or minus one from those of a defined MicroPDF417 row. These numbers must be