INTERNATIONAL STANDARD

ISO/IEC 24778

First edition
2008-02-15

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

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

Reference number
ISO/IEC 24778:2008(E)

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

Published in Switzerland

Contents

Page

Foreword

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

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

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

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

ISO/IEC 24778 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

Introduction

Aztec Code is a two-dimensional matrix symbology whose symbols are nominally square, made up of square modules on a square grid, with a square bullseye pattern at their center. Aztec Code symbols can encode from small to large amounts of data with user-selected percentages of error correction.

Manufacturers of bar code equipment and users of the technology require publicly available standard symbology specifications to which they can refer when developing equipment and application standards. The publication of standardised symbology specifications is designed to achieve this.

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

1 Scope

This International Standard defines the requirements for the symbology known as Aztec Code. It specifies the Aztec Code symbology characteristics including data character encodation, rules for error control encoding, the graphical symbol structure, symbol dimensions and print quality requirements, a reference decoding algorithm, and user-selectable 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 15415:2004, *Information technology — Automatic identification and data capture techniques — Bar code print quality test specification — Two-dimensional symbols*

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

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

AIM Inc. International Technical Specification: Extended Channel Interpretations

– Part 1, Identification Schemes and Protocols

– Part 2, Registration Procedure for Coded Character Sets and Other Data Formats

– Character Set Register

3 Terms, definitions, symbols and functions

3.1 Terms and definitions

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

3.1.1
bullseye
set of concentric square rings used as the finder pattern in Aztec Code

3.1.2
checkword
codeword which is included in a symbol for error correction and/or error detection

3.1.3
dataword
codeword which is part of the data message encoded in a symbol

3.1.4
domino
2-module sub-structure of the symbol character in Aztec Code which is the elemental entity used in graphical encoding of the symbol

3.1.5
Mode Message
short fixed-length, error-corrected subsidiary message within an Aztec Code symbol which directly encodes the symbol's size and data message length

3.2 Symbols and functions

3.2.1 Mathematical symbols

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

B the number of bits in each codeword

C_b the symbol capacity in number of bits

C_w the symbol capacity in number of codewords

D the number of data (message) codewords in the symbol

K the number of error correction codewords in the symbol, equal to C_w - D

L the number of data layers (1 to 32) in the symbol, defining its size

m the symbology identifier modifier value

X the X-dimension or nominal square grid spacing

x a general variable used to express error correction polynomials

(x,y) Cartesian coordinates within the module grid

3.2.2 Mathematical functions and operations

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

abs() is the absolute value function

div is the integer division operator

max(a,b) is the greater of a and b

mod is the remainder after integer division

4 Symbology characteristics

4.1 Basic characteristics

Aztec Code is a two dimensional matrix symbology with the following basic characteristics:

a. Encodable character set:

1. All 8-bit values can be encoded. The default interpretation shall be:

   a. for values 0 to 127, in accordance with the U.S. national version of ISO/IEC 646;

   (NOTE: This version consists of the GO set of ISO/IEC 646 and the CO set of ISO/IEC 6429 with values 28 to 31 modified to FS, GS, RS and US respectively.)

   b. for values 128 - 255, in accordance with ISO/IEC 8859-1.

   This interpretation corresponds to ECI 000003.

2. Two non-data characters can be encoded, FNC1 for compatibility with some existing applications and ECI escape sequences for the standardized encoding of message interpretation information.

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

c. Symbol size:

1. The smallest Aztec Code symbol is 15 x 15 modules square, and the largest is 151 x 151.

2. No quiet zone is required outside the bounds of the symbol.

d. Data capacity (at recommended error correction level):

1. The smallest Aztec Code symbol encodes up to 13 numeric or 12 alphabetic characters or 6 bytes of data.

2. The largest symbol encodes up to 3832 numeric or 3067 alphabetic characters or 1914 bytes of data.

e. Selectable error correction:

1. User-selectable, from 5 % to 95 % of the data region, with a minimum of 3 codewords.

2. Recommended level is 23 % of symbol capacity plus 3 codewords.

f. Code type: Matrix

g. Orientation independent: Yes

4.2 Summary of additional features

The following summarizes additional features that are inherent or optional in Aztec Code:

a. Reflectance Reversal (Inherent): Though Aztec Code symbols are shown and described in this specification always with the finder's center dark and with dark modules encoding binary 1s throughout, symbols exhibiting the opposite reflectance characteristics are easily autodiscriminated and decoded with the standard reader.

b. Mirror Image (Inherent): Images which contain an Aztec Code symbol in mirror reversal, either because they are obtained using a reflected optical path, a reversed scan direction, or from behind through a clear substrate, are easily autodiscriminated and decoded with the standard reader.

c. Extended Channel Interpretation (Optional): The ECI mechanism enables characters from various character sets (e.g. Arabic, Cyrillic, Greek, Hebrew) and other data interpretations or industry-specific requirements to be represented.

d. Structured Append (Optional): Structured Append allows files of data to be represented logically and continually in up to 26 Aztec Code symbols. The symbols may be scanned in any sequence to enable the original data to be correctly reconstructed.

e. Reader Initialization Symbols (Optional): A distinct format of Aztec Code symbol is available for use in barcode menus for reader initialization. The encoded message in these special symbols is never passed on to an application.

f. Aztec "Runes" (Optional): a series of 256 small, machine-readable marks compatible with Aztec Code are available for special applications. See Annex A.

5 Symbol description

Aztec Code symbols are nominally square, made up of square modules on a square grid, with a square bullseye pattern at their center. Figure 1 shows two representative Aztec Code symbols, a small 1-layer symbol on the left which encodes 12 digits with 47 % error correction and a larger 6-layer symbol on the right which encodes 168 text characters with 30 % error correction.

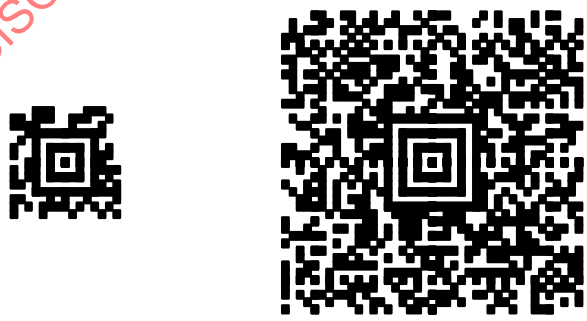

Figure 1 — Representative Aztec Code symbols

These symbols illustrate the two basic formats of Aztec Code symbols: on the left is a "compact" Aztec Code symbol, visually characterized by a 2-ring bullseye, useful for encoding shorter messages efficiently, while on the right is a "full-range" Aztec Code symbol, visually characterized by a 3-ring bullseye, which supports much larger symbols for longer data messages. Since encoders can autoselect and decoders autodiscriminate between the two formats, a seamless transition is achieved to cover the full spectrum of applications.

5.1 Symbol structure

The underlying structure of a compact Aztec Code symbol is shown in Figure 2, and that of a full-range Aztec Code symbol is shown in Figure 3. In both cases, the Aztec Code symbol has at its center a Core Symbol which is then surrounded by data fields on all four sides.

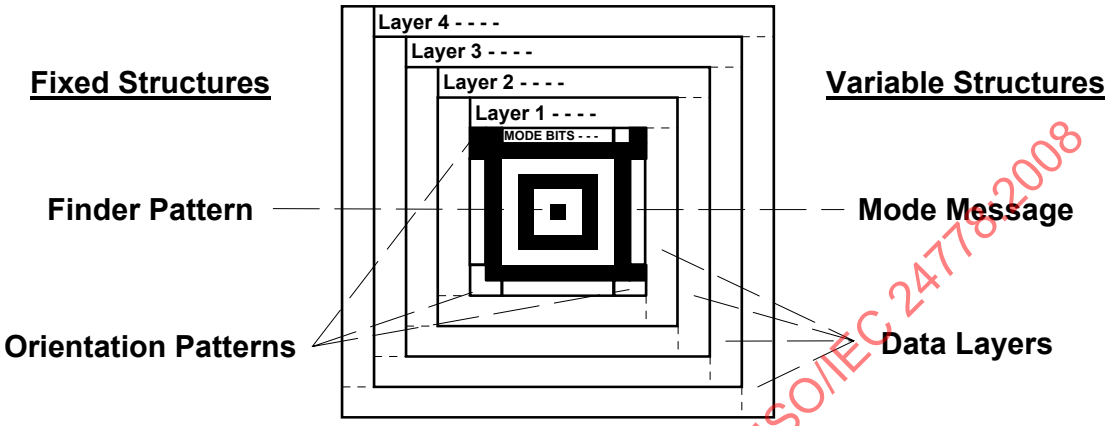

Figure 2 — Structure of a "compact" Aztec Code symbol

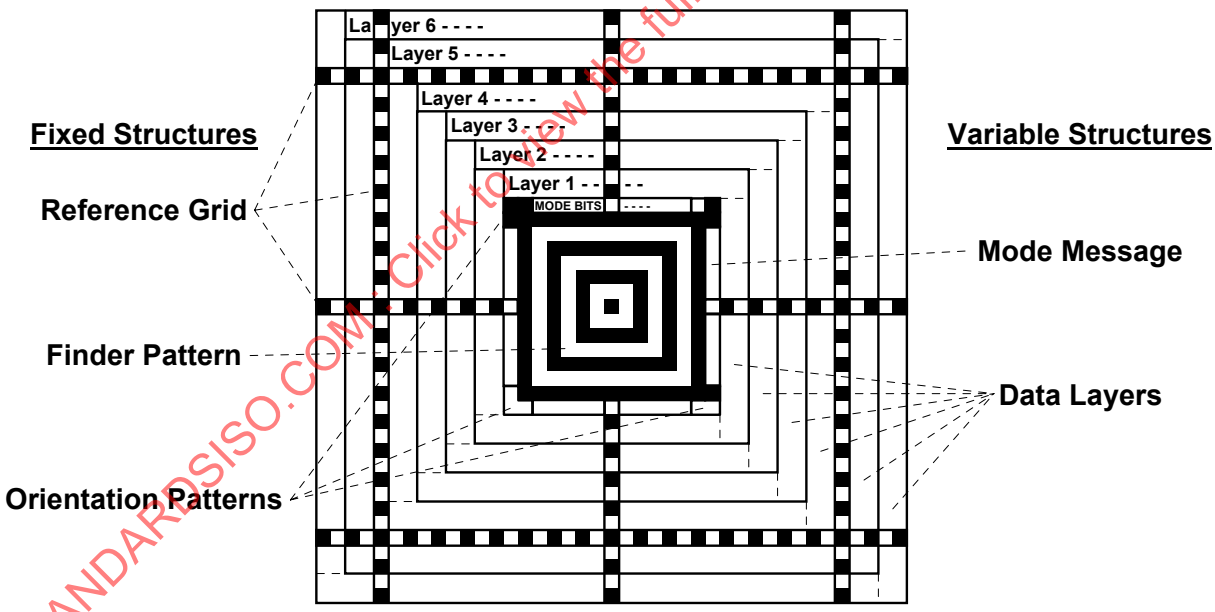

Figure 3 — Structure of a "full-range" Aztec Code symbol

5.1.1 Core Symbol

The Core Symbol, always square and at the exact center of an Aztec Code symbol, consists of a finder pattern, orientation patterns, and a Mode Message. This core covers an 11x11 module area in compact symbols and a 15x15 module area in full-range symbols. It is called the Core "Symbol" because it must be successfully found and decoded before decoding can proceed into the surrounding data fields.

5.1.1.1 Finder pattern

The finder pattern in Aztec Code is a set of concentric square rings. Centered on a single dark module, there is a ring of light modules surrounded by a larger ring of dark modules, and so forth outward to a second 9x9 module dark ring in compact symbols and to a third 13x13 module dark ring in full-range symbols.

5.1.1.2 Orientation patterns

Four 3-module chevron-shaped orientation patterns are located at the corners of the finder pattern. The upper lefthand pattern is all dark and the lower lefthand pattern is all light, while the upper righthand pattern has 2 modules dark and the lower right pattern has just 1 module dark, as shown in Figures 2 and 3.

5.1.1.3 Mode Message

The single layer of bits adjoining the finder pattern, excluding the orientation patterns and in full-range symbols also excluding the center bit along each side (which is part of the reference grid), comprises an error-corrected Mode Message wrapped in a clockwise direction starting from the upper left corner. This message explicitly encodes both the number of data layers in the overall symbol (and thus its size) and the number of datawords in those layers, the rest being error correction checkwords for that message. Encoding details for the Mode Message are given in 7.2.

5.1.2 Data fields

The data fields symmetrically surround the Core Symbol with one or more data layers. In full-range symbols, a reference grid also threads throughout the data fields.

5.1.2.1 Reference grid

The reference grid, clearly evident in Figure 3, is a ladder-like extension of the dark/light periodicity in the finder along every 16th row and column of the symbol, extending to the limit of the data fields. Its regular structure provides outlying reference points often needed to accurately map the data field in the larger full-range symbols. Compact Aztec Code symbols, which are of limited size, have no reference grid structure.

5.1.2.2 Data layers

The message data themselves plus their error correction words are laid into 2-module thick layers, spiralling clockwise from the upper left corner of the Core Symbol outward, and in full-range symbols necessarily skipping over module positions occupied by the reference grid. Compact Aztec Code symbols can have from one to four data layers, while full-range Aztec Code symbols can have from one up to 32 data layers. Details of the message encoding, codeword formation, error correction encoding, and the final laying of codewords into the data layers are given in 7.3.

5.2 Symbol character structure and sequence

In order to enhance Reed-Solomon error correction performance, the codewords, and thus the symbol characters, vary in size from 6-bits up to 12-bits depending on the overall symbol size, as illustrated in Figure 4.

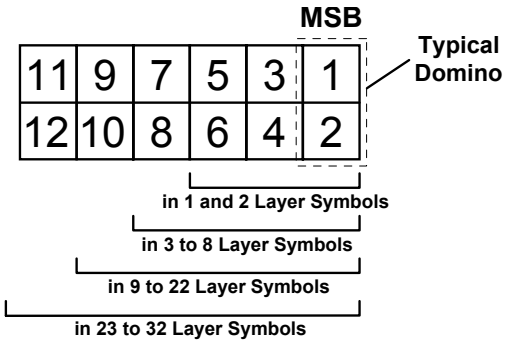

Figure 4 — Symbol character structure

While spiralling around the core, turning corners and occasionally skipping across the reference grid, the symbol characters' actual shapes vary widely. However, if they are regarded as sequences of "dominos", each 2 modules tall by 1 module wide, then the sequence of symbol characters becomes a sequence of dominos whose placement is highly systematic throughout the data fields, and thus easy to place during encoding and easy to map during decoding. Figure 5 shows how the sequence of dominos is positioned as it turns corners and transitions between data layers.

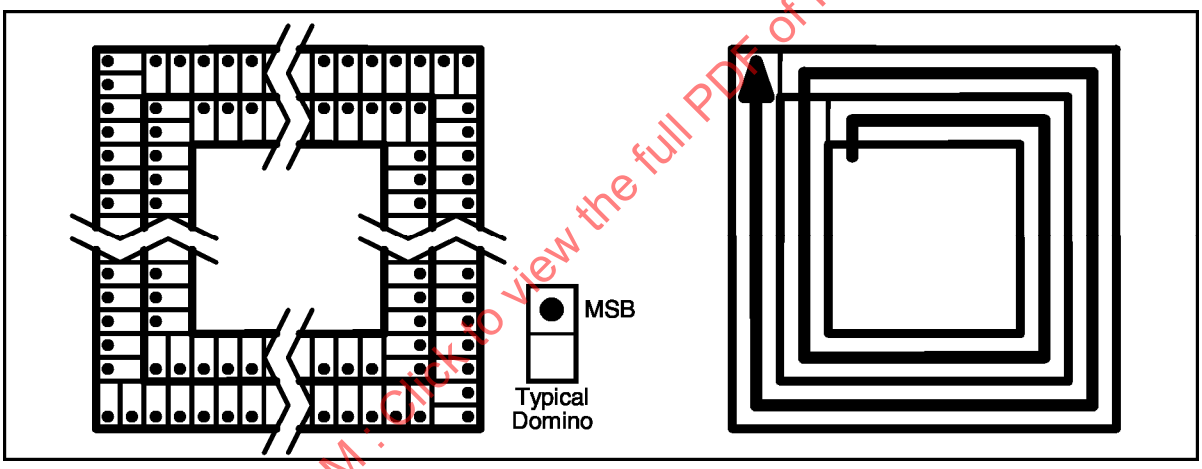

Figure 5 — "Domino" layout and sequencing

The sequence of codewords that spirals outward from the Core Symbol is in fact reversed from its natural order: the first codeword in the spiral is the last Reed-Solomon checkword, followed then by the immediately preceeding checkword, and so forth through the check words and then through the message codewords, until the final codeword in the outermost data layer is the first codeword of the encoded message. This arrangement enhances error correction by locating the data codewords (which have erasure detection because data codewords with all zeros or all ones are illegal) near the symbol's perimeter where erasures are more likely to occur.

5.3 Symbol size and capacity

Table 1 lists the overall size and capacities of the different sized Aztec Code symbols.

The data capacities shown are based on the recommended error correction levels. They represent approximate limits because the message encoding efficiency depends in a detailed way on message content and because the error correction level is user adjustable.

Table 1 — The size and capacities of Aztec Code symbols

# of Data Layers	Symbol Size (in x)	Codeword Count x Size	Symbol Bit Capacity	Symbol Data Capacities		
				Digits	Text	Bytes
1*	15 x 15	17 x 6	102	13	12	6
1	19 x 19	21 x 6	126	18	15	8
2*	19 x 19	40 x 6	240	40	33	19
2	23 x 23	48 x 6	288	49	40	24
3*	23 x 23	51 x 8	408	70	57	33
3	27 x 27	60 x 8	480	84	68	40
4*	27 x 27	76 x 8	608	110	89	53
4	31 x 31	88 x 8	704	128	104	62
5	37 x 37	120 x 8	960	178	144	87
6	41 x 41	156 x 8	1248	232	187	114
7	45 x 45	196 x 8	1568	294	236	145
8	49 x 49	240 x 8	1920	362	291	179
9	53 x 53	230 x 10	2300	433	348	214
10	57 x 57	272 x 10	2720	516	414	256
11	61 x 61	316 x 10	3160	601	482	298
12	67 x 67	364 x 10	3640	691	554	343
13	71 x 71	416 x 10	4160	793	636	394
14	75 x 75	470 x 10	4700	896	718	446
15	79 x 79	528 x 10	5280	1008	808	502
16	83 x 83	588 x 10	5880	1123	900	559
17	87 x 87	652 x 10	6520	1246	998	621
18	91 x 91	720 x 10	7200	1378	1104	687
19	95 x 95	790 x 10	7900	1511	1210	753
20	101 x 101	864 x 10	8640	1653	1324	824
21	105 x 105	940 x 10	9400	1801	1442	898
22	109 x 109	1020 x 10	10200	1956	1566	976
23	113 x 113	920 x 12	11040	2116	1694	1056
24	117 x 117	992 x 12	11904	2281	1826	1138
25	121 x 121	1066 x 12	12792	2452	1963	1224
26	125 x 125	1144 x 12	13728	2632	2107	1314
27	131 x 131	1224 x 12	14688	2818	2256	1407
28	135 x 135	1306 x 12	15672	3007	2407	1501
29	139 x 139	1392 x 12	16704	3205	2565	1600
30	143 x 143	1480 x 12	17760	3409	2728	1702
31	147 x 147	1570 x 12	18840	3616	2894	1806
32	151 x 151	1664 x 12	19968	3832	3067	1914

* An asterisk indicates a “compact” symbol; the rest are “full-range” symbols.

NOTE: Full range symbols with 1, 2, or 3 layers are useful only for reader initialization.

6 General encodation procedures

The following steps are required to convert data into the encoded form represented in an Aztec Code symbol. The following clauses of this specification specify all the rules and procedures. An encoding example is shown in Annex G.

1. Data from a 256 character set may be encoded in Aztec Code. The input message is presented in a stream of byte values reading from left to right. Special FNC1 or ECI flag characters may be inserted at any points in the stream.

2. Each message character is translated into 4, 5, or 8 bits, preceded by additional 4 or 5-bit shift and latch codes as needed, forming a long continuous data bit stream.

3. The minimum number of bits to be encoded is computed by taking the length of the data bit stream and adding as many bits as needed to reach either the default or user-specified error correction percentage. From this calculation, the format and minimum size (number of data layers L) of the symbol is selected using Table 1. This in turn establishes both the codeword size B and overall symbol capacity in codewords C_w.

4. The data bit stream is laid into codewords, systematically avoiding the formation of any codewords containing all 0's or all 1's, thus creating D message codewords.

5. The number K of checkwords becomes C_w minus D. Systematic Reed-Solomon encoding, based on a Galois Field of size 2^B and using a generator polynomial of order K, is employed to generate K additional check codewords which are appended to the sequence of message codewords.

6. The binary values of L and D are formed into a Mode Message, and systematic Reed-Solomon encoding based on GF(16) is employed to generate additional check bits.

7. Graphically, the L-layer symbol is constructed by placing modules first for the fixed structures of the finder, orientation patterns, and (if full-range) reference grid, then for the Mode Message wrapping around the finder, and finally for the spiralling layers of dominos which constitute the sequence of datawords and checkwords taken in reverse order.

7 Symbol structure

7.1 Fixed pattern structures

An Aztec Code symbol contains three types of fixed pattern - the finder, orientation bits, and if full-range a reference grid. These are all shown in Figures 2 and 3. Their specification is facilitated by regarding the symbol grid as a Cartesian plane with (0,0) at the symbol's center and with the x-axis pointing to the right and the y-axis pointing upward.

7.1.1 The finder

The finder pattern is a square bullseye target filling the central square region whose corner modules are centered at (-F,-F), (-F,F), (F,F), and (F,-F) where F is 4 in compact symbols and 6 in full-range symbols. Where the value "0" represents a light module and "1" represents a dark module, all the modules within the finder region can be defined as encoding:

((max(abs(x),abs(y))) + 1) mod 2

7.1.2 The orientation bits

The orientation bits are four groups of three modules attached to the corners of the finder. With F as defined above, six dark modules are located at (-F-1,F), (-F-1,F+1), (-F,F+1), (F+1,F+1), (F+1,F) and (F+1,-F) and six light modules are located at (F,F+1), (F+1,-F-1), (F,-F-1), (-F,-F-1), (-F-1,-F-1) and (-F-1,-F).

7.1.3 The reference grid

The reference grid extends throughout full-range Aztec Code symbols occupying every location where x is a multiple of 16 or y is a multiple of 16, i.e. ((x mod 16) = 0) or ((y mod 16) = 0). Again where the value "0" represents a light module and "1" represents a dark module, all the modules within the reference grid can be defined as encoding:

(x + y + 1) mod 2.

Where the reference grid and finder overlap, they have the same values.

7.2 Mode Message encoding and structure

The Mode Message is a sequence of bits which encircles the finder pattern. It includes bits designating the symbol size, followed by bits designating the message length, and finally bits which provide the Mode Message with error redundancy.

7.2.1 Symbol size designator

The symbol size designator bits directly encode in binary the number L of data layers in the overall symbol, minus one. In the compact format there are 2 such bits, whose binary value from 0 to 3 signifies symbols with from one up to 4 layers of data. In the full-range format there are 5 such bits, whose binary value from 0 to 31 signifies symbols with from one up to 32 layers of data. These bits in turn designate the symbol's overall size dimension and data capacity C_w, and they also establish the number of bits B in each data codeword, as given in Table 1.

7.2.2 Message length designator

The message length designator bits directly encode in binary the number D of the data codewords which actually encode the message, minus 1, with the rest of the codewords in the symbol being used for error correction. In the compact format there are 6 such bits, whose binary value from 0 to 63 signifies symbols whose message fills from one up to 64 codewords. In the full-range format there are 11 such bits, whose binary value from 0 to 2047 would signify symbols whose message fills from one up to 2048 codewords. However, the symbol capacity C_w presents a logical upper limit for message length, and the recommended upper limit allowing for adequate error protection is (0.77 C_w) - 3.

For special reader initialization symbols (see Clause 9), the most significant bit of the message length designator is artificially set to a one, thus indicating a message length which exceeds C_w.

7.2.3 Error encodation for the Mode Message

The symbol size plus message length bits are packed into several 4-bit codewords which become the "datawords" of the Mode Message. To these are then added several more "checkwords" computed by systematic Reed-Solomon encoding over the Galois Field GF(16) based on a prime modulus polynomial of $x^4 + x + 1$ (19 decimal).

In compact Aztec Code symbols, the Mode Message data of "ssmmmmmm" (where "ss" designates size and "mm..." designates message length) forms two 4-bit datawords, to which 5 checkwords are added using a generator polynomial of $(x-2^1)...(x-2^5)$ equal to:

$$x^5 + 11x^4 + 4x^3 + 6x^2 + 2x + 1$$

Together the 2 datawords plus the 5 checkwords, obtained as coefficients of the remainder when a polynomial of the datawords is scaled upwards by x^5 and then long divided by the generator polynomial within GF(16), form the 28-bit Mode Message for a compact symbol.

In full-range symbols, the Mode Message data of "sssssmmmmmmmmmmm" forms four 4-bit datawords, to which 6 checkwords are added using $(x-2^1)...(x-2^6)$ equal to:

$$x^6 + 7x^5 + 9x^4 + 3x^3 + 12x^2 + 10x + 12$$

Together the 4 datawords plus the 6 checkwords form the 40-bit Mode Message for a full-range Aztec Code symbol.

7.2.4 Module placement for the Mode Message

The bits of the Mode Message are graphically encoded using dark modules to represent a "1" and light modules to represent a "0".

The string of Mode Message bits, from the most significant size designation bit through to the least significant bit of the final checkword, is laid in a single layer of modules starting near the upper left corner of the finder and wrapping clockwise around it, skipping over positions occupied by the orientation patterns and reference grid. In a compact symbol, the 28 bits are essentially parsed into four 7-bit strings, with the first running left-to-right across the top of the finder, the next running down its right side, the third running right-to-left along its bottom, and the fourth running up its left side. In a full-range symbol, the 40-bit Mode Message is parsed into eight 5-bit strings, with the first two running left-to-right across the top of the the finder, and so forth.

7.3 Data message encoding and structure

The data message is a sequence of C_w B-bit codewords which spirals outward from the Core Symbol. These consist of D codewords, exactly the number needed to encode the source data message, plus as many check codewords as are needed to fill out the symbol, all taken in reverse order (see 7.3.3).

7.3.1 Source message encoding

The source sequence of data characters is translated into a sequence of message codewords in a two-step process. First the data characters in succession are converted to equivalent binary values, using intermediary shift and latch values as needed, producing a long binary stream of input data. Second this string is laid into a sequence of B-bit codewords using an exclusionary rule that avoids any occurrence of a codeword with all 0's or with all 1's.

7.3.1.1 Translation into a binary stream

Table 2 presents values assigned to several code sets of ASCII character values, and to shift and latch operations for jumping between code sets. The Upper and Lower code sets contain upper and lowercase alphabetic characters respectively; Mixed contains a mixture of ASCII control and punctuation characters; Punctuation contains common punctuation characters and combinations; and Digit contains the numerals plus some punctuation. Other than "FLG(n)" (see below), all those entries in Table 2 that are shown with no corresponding ASCII value are shift or latch operations for accessing the different code sets.

Message encoding starts in the Upper code set and can be latched (semi-permanently, via U/L, L/L, M/L, P/L, or D/L) or shifted (for one character only, via U/S or P/S) into the other modes as needed. In Upper, Lower, Mixed, and Punctuation code sets each character, latch or shift adds a 5-bit value to the data bitstream, while in the Digit code set each character, latch, or shift adds just a 4-bit value to the stream. In all cases, the binary values are strung together with their most significant bits leading.

Byte Shift (B/S) is a special case, shifting into a runlength-controlled string of literal 8-bit bytes. Following a B/S is a 5-bit binary value: if non-zero, it encodes the number of bytes (1 to 31) that follow, but if zero then the next 11 bits encode the number of bytes less 31. Byte Shift can thus encode either isolated extended ASCII or control characters or long strings of byte data, possibly filling the whole symbol. At the end of the byte string, encoding returns to the mode from which B/S was invoked.

Table 2 — Aztec Code high-level (message) encoding

	Upper		Lower		Mixed		Punct.		Digit	
Value	Char	ASCII	Char	ASCII	Char	ASCII	Char	ASCII	Char	ASCII
0	P/S		P/S		P/S		FLG(n)		P/S	
1	SP	32	SP	32	SP	32	CR	13	SP	32
2	A	65	a	97	SOH	1	CR LF	13, 10	0	48
3	B	66	b	98	STX	2	. SP	46, 32	1	49
4	C	67	c	99	ETX	3	, SP	44,32	2	50
5	D	68	d	100	EOT	4	: SP	58, 32	3	51
6	E	69	e	101	ENQ	5	!	33	4	52
7	F	70	f	102	ACK	6	"	34	5	53
8	G	71	g	103	BEL	7	#	35	6	54
9	H	72	h	104	BS	8	$	36	7	55
10	I	73	i	105	HT	9	%	37	8	56
11	J	74	j	106	LF	10	&	38	9	57
12	K	75	k	107	VT	11	'	39	,	44
13	L	76	l	108	FF	12	(	40	.	46
14	M	77	m	109	CR	13	)	41	U/L	
15	N	78	n	110	ESC	27	*	42	U/S	
16	O	79	o	111	FS	28	+	43		
17	P	80	p	112	GS	29	,	44		
18	Q	81	q	113	RS	30	-	45		
19	R	82	r	114	uS	31	.	46		
20	S	83	s	115	@	64	/	47		
21	T	84	t	116	\	92	:	58		
22	U	85	u	117	^	94	;	59		
23	V	86	v	118	_	95	<	60		
24	W	87	w	119	`	96	=	61		
25	X	88	x	120	|	124	>	62		
26	Y	89	y	121	~	126	?	63		
27	Z	90	z	122	DEL	127	[	91		
28	L/L		U/S		L/L		]	93		
29	M/L		M/L		U/L		{	123		
30	D/L		D/L		P/L		}	125		
31	B/S		B/S		B/S		U/L			

The character designated “FLG(n)” in the Punctuation column of Table 2 is a special in-place flag used to represent a variety of non-data characters supported by many standard symbologies. In the bit stream, the FLG(n) value is followed by 3 extra bits encoding its argument “n” in binary; thus n is valued between 0 and 6 (7 is invalid).

FLG(0) represents “FNC1”, a non-data flag deriving from Code 128. When FNC1 precedes the first message character, it signals the use of GS1 data formatting rules using Application Identifiers and affects the symbology identifier modifier value. When FNC1 immediately follows a single upper or lower case letter or two digits at the

beginning of the message, then it signals the use of some other industry-standard format, identified by the preceeding data, and also affects the modifier value. When FNC1 occurs at any other location in the data, it serves as a field separator and causes an ASCII 29 (<GS>) to be inserted in its place in the output data string.

FLG(1) through FLG(6) represent the Extended Channel Interpretation (ECI) flag, which in an ECI-compliant reader causes "\nnnnnn", a backslash followed by a 6-digit number, to be inserted into the output data string. (An ECI-compliant reader also doubles all encoded data backslashes in the output string and sets an appropriate symbology identifier modifier value.) The argument n in this case indicates how many of the 6 digits are explicitly encoded in the symbol, using Digit mode, with leading zeros being assumed for the rest. ECI 000123, for example, is encoded FLG(3)123, after which encoding reverts to the mode from which FLG(n) was invoked.

Annex H presents an algorithm for finding that sequence of code sets, latchs and shifts that will minimize the number of bits needed to encode any input sequence of bytes, ultimately minimizing symbol size and maximizing its error redundancy.

7.3.1.2 Formation of data codewords

In the second step of source encoding, once the overall symbol size has been selected and the codeword size B thus determined, then the stream of message bits is laid into the sequence of B-bit (B = 6, 8, 10, or 12) message codewords in a generally direct fashion, starting at the most significant bit of the first codeword, with two key exceptions: whenever the first B-1 bits placed in a codeword are all "0"s, then a dummy "1" is inserted into that codeword's LSB and the following message bit starts off the next codeword. Similarly a message codeword that starts with B-1 "1"s has a dummy "0" inserted into its LSB. In this manner, message codewords of all "0"s or all "1"s are illegal and can be deemed erasures during the decoding process.

In the end, the character and byte boundaries in the original message have no well-defined relationship with the codeword boundaries. Up to B-1 bits may remain unfilled in the final message codeword, and they are to be padded out with "1"s (and possibly a final dummy "0" if necessary) to eliminate any ambiguity. (Note: because this pad pattern can appear to be the start of a Binary Shift sequence, readers must take care to terminate message interpretation with the final data codeword.)

7.3.2 Error encodation for the data message

Source encoding produces D codewords. To these are appended $K = C_w - D$ checkwords which are computed by systematic Reed-Solomon encoding over $GF(2^B)$ based on the corresponding prime modulus polynomial listed in Table 3. The checkword values are the coefficients of the remainder when the polynomial formed by message codewords is first scaled upwards by x^K and then long-divided by the generator polynomial of $(x-2^1)...(x-2^K)$. See Annex B.1.

Table 3 — Data codeword sizes and prime modulus polynomials

# Layers	Codeword	Galois Field	Prime Modulus Polynomial	Binary Equiv.	Decimal Equiv.
1 – 2	6-bits	GF(64)	$x^6 + x + 1$	1000011	67
3 – 8	8-bits	GF(256)	$x^8 + x^5 + x^3 + x^2 + 1$	100101101	301
9 – 22	10-bits	GF(1024)	$x^{10} + x^3 + 1$	10000001001	1033
23 – 32	12-bits	GF(4096)	$x^{12} + x^6 + x^5 + x^3 + 1$	1000001101001	4201

7.3.3 Module placement for the data message

The bits of the data message are graphically encoded using dark modules to represent a "1" and light modules to represent a "0".

For placement within an Aztec Code symbol, the sequence of D + K codewords is taken in reverse order and wrapped into L layers each 2-modules thick, starting adjacent to the upper left corner of the Core Symbol and spiralling outward in a clockwise direction. The exact placement of individual modules is best realized by first regarding each codeword as a 2-module high "brick" as pictured in Figure 4, and then breaking that brick into B/2 dominos which are each 1 module wide with a more significant bit sitting atop a less significant bit. These dominos are then placed within data layers starting at the "L" of "Layer 1 ..." in Figure 2 or 3 and wrapping clockwise, skipping over positions occupied by the reference grid, with the less significant bit of each domino always nearer the symbol's center. Dominos along the bottom will thus seem to be upside down.

Figure 5 illustrates the sequence and orientation of the dominos when turning the corners in any layer and when transitioning between layers. The symbol characters are often irregular in shape at the four corners or non-contiguous when skipping across the reference grid, but they nonetheless represent integral binary codeword values. Within Layers 12 and 27 of full-range symbols, even the dominos themselves are bisected by a reference grid cell, but still each domino's more significant bit is positioned directly "above" its less significant counterpart (that is, directly outward in line from the Core Symbol) following the orientations shown in Figure 5.

In the outermost layer of each symbol, at the end of the spiral, there may be several dominos left over and they are left with both modules light ("0").

8 Structured Append

In order to fit a non-square area or to handle larger messages than are practical in a single symbol, a data message can be distributed across several Aztec Code symbols. Although this can be achieved by a user-defined header structure, the following is a standard method that shall be supported by Aztec Code readers.

All symbols which are part of a Structured Append sequence shall start message encoding with the sequence:

M/L U/L [space I...D space] M N ...

which consists of the Structured Append flag sequence "M/L U/L" where M/L and U/L are latch characters, an optional space-delimited message ID field "I...D", and a 2-character M-of-N sequencing field. This header is then followed directly by the actual message segment to be encoded.

An Aztec Code reader, upon detecting an Upper Latch ahead of any stored data, shall either (1) interpret the header and store the message segment for later transmission as part of the complete assembled message, or (2) signal that this is an ordered append symbol and transmit it with appropriate sequencing information. The ECI protocol provides a symbology independent way to convey this information within ECI-compliant applications. Alternately, the Aztec Code reader shall signal that this is an ordered append symbol by setting an appropriate symbology identifier modifier value, then transmit the header and message segment intact.

The optional message ID field is any number of data characters starting and ending with a space character, and shall be the same for all symbols which comprise the same message. Though the "I...D" string may be made up of any characters (except additional spaces), the most efficient encoding will result if it is a string of uppercase letters.

The sequencing field is two uppercase letters, the first encoding which symbol this is in a sequence and the second encoding the total number of symbols in the sequence, where "A" = 1, "B" = 2, and so forth. For example, a four symbol sequence would have sequencing fields of "AD", "BD", "CD", and "DD". Up to 26 symbols can be linked together using this Structured Append protocol.

9 Reader initialization symbols

It is often useful to have a class of symbols which function solely to deliver initialization or configuration information into a reader. A 1-layer compact mode Aztec Code symbol or a 1 to 22-layer full-range symbol can be encoded for reader initialization purposes by artificially setting the most significant bit of the message length designator, with its lesser bits encoding the real message length. Because such a symbol's total capacity is less than the designated message length, this otherwise illegal setting signals that the encoded message shall not be transmitted but may

be used instead to configure the reader. The format and meaning of the encoded data shall be determined by the reader manufacturer.

10 Extended Channel Interpretation

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

a. international character sets (or code pages)

b. general purpose interpretations such as encryption and compaction

c. user defined interpretations for closed systems

d. control information for Structured Append in unbuffered mode.

The Extended Channel Interpretation protocol is fully specified in AIM Inc. International Technical Specification - Extended Channel Interpretations Part 1. The protocol provides a consistent method to specify particular interpretations on byte values before printing and after decoding.

The Extended Channel Interpretation is identified by a 6-digit number which is encoded in the Aztec Code symbol by the ECI character followed by one to six 4-bit values representing digits. Specific interpretations are listed in AIM Inc. Extended Channel Interpretations Character Set Register.

The Extended Channel Interpretation can only be used with readers enabled to transmit the symbology identifiers. Readers that are not enabled to transmit the symbology identifier shall not transmit the data from any symbol containing an ECI. An exception can be made if the ECI(s) can be handled entirely within the reader.

10.1 Encoding ECIs in Aztec Code

The ECI assignment is invoked by encoding the FLG(n) character in the Punctuation code set, which is followed by 3 bits encoding “n” valued between one and six in binary. FLG(1-6) is then followed by one to six 4-bit values which, using the Digit code set, represent the ECI number. Leading zeros which were dropped during the encode are reinserted in the output message in the decode to expand the ECI number to six digits.

10.2 Code sets and ECIs

The code set used is determined strictly by the 8-bit data values being encoded and does not depend on the Extended Channel Interpretation in force. For example, a sequence of values in the range 48 to 57 (decimal) would be most efficiently encoded in the Digit code set even if the sequence is not to be interpreted as numbers.

10.3 ECIs and Structured Append

ECIs may be encoded anywhere in the message encoded in a single symbol or Structured Append set of Aztec Code 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.

10.4 Post-decode protocol

The protocol for transmitting ECI data shall be as defined in 16.3. When using ECIs, symbology identifiers (see 16.4) shall be fully implemented and the appropriate symbology identifier transmitted as a preamble.

11 User considerations

Potential users of Aztec Code have several decisions regarding the data to be encoded, the amount of error correction needed, and the possible distribution of the message over several symbols that they may want to consider.

11.1 User selection of encoded message

Though for many barcode applications the encoded data messages will be strictly defined, those in the planning stages may offer some leeway to adopt data formats optimized for efficient encoding. In general the most efficient encoding is achieved when the message is made up of long strings of characters from a single code set (see Table 2). Uppercase text, all numeric data, or efficiently packed byte data are three types of message which encode most efficiently in Aztec Code symbols.

11.2 User selection of minimum error correction level

The design of Aztec Code technically allows a symbol to include as little as none or as much as 99 % in error correction codewords, though both limits are unsound. The recommended level of error correction for normal use is 23 % of symbol capacity plus 3 codewords more. Thus a 5-layer symbol which holds 120 codewords should normally include at least 28 + 3 checkwords, leaving up to 89 codewords for encoding the data message. Print programs should enforce this level as the default.

Users, judging their applications to be especially benign or critical, may choose to specify an alternate minimum error correction percentage, ranging from below 10 % to above 50 % plus always, for data security, 3 additional checkwords. This is called a “minimum” percentage because, depending on message length, the symbology will typically have to add extra checkwords above this minimum to fill out the symbol.

Some applications will be best served by specifying a fixed size (number of data layers) to be used for all Aztec Code symbols regardless of their data content. In this case, the user should specify a size which includes adequate error correction for the longest message anticipated; then typically shorter messages will be encoded with excess error correction, creating more robust symbols for this application.

11.3 User selection of Structured Append

Structured Append allows data messages to be spread across multiple Aztec Code symbols, then reconstructed when they have all been read, in any order. This option allows data to be encoded into a non-square region. It is also available for encoding data messages which exceed the practical capacity of a single Aztec Code symbol.

11.4 User selection of optional symbol formats

Referring to Table 1, printing programs will generally print the smallest symbol for a given data set and error correction percentage in the data layer sequence 1*, 2*, 3*, 4*, 4, 5, 6, ... , 32. For Reader Initialization symbols, the printing program will generally print the smallest symbol in the data layer sequence 1*, 1, 2, 3, 4, 5, ... , 22. Alternately, the user may specify the size and format of the desired symbol, overriding the default settings.

Users will need to explicitly specify the printing of Runes for small data set applications.

12 Dimensions

The basic Aztec Code symbol dimension is X, the center-to-center spacing of its nominally square grid, also nominally equal to the height and width of the square modules within that grid.

This symbology specification places no limits on the size of X. However, the horizontal and vertical grid spacings shall be constant throughout a symbol and within 10 % of each other. Furthermore, each dark module’s width shall equal the horizontal grid spacing within ±20 %, and its height shall equal the vertical grid spacing within ±20 %.

13 User guidelines

13.1 Human readable interpretation

A human readable interpretation of the encoded data message may accompany an Aztec Code symbol in any region beyond the outermost data layer (see Figure G.2). The font and size are not specified.

13.2 Autodiscrimination capability

Aztec Code may be read by suitably programmed bar code decoders which have been designed to autodiscriminate it from other symbologies.

To maximise reading security, a decoder's valid set of symbologies should be limited to those needed by a given application.

13.3 User-defined application parameters

Application standards shall define parameters of Aztec Code symbols specified herein as variable, as follows:

a. The data to be encoded.

b. The X-dimension used in printing the symbol.

c. The placement of the symbol on a label, object, or document.

d. The level of symbol print quality required.

They may also choose to override the default error correction level, specifying either an alternate minimum error correction percentage or a fixed symbol format and size.

14 Reference decode algorithm

This reference decode algorithm finds the symbols in an image and decodes them. This is the decode algorithm used in the determination of symbol print quality. It may also be used by practical reader implementations.

Decoding Aztec Code from a stored image involves the steps of:

1. locating the symbol within the image,

2. processing the bullseye to pinpoint its center and determine the symbol's major axes and nominal module spacings,

3. decoding the Core Symbol, first determining if the symbol is black/white reversed, then its format, then its orientation, and then mapping, decoding, and checking the Mode Message,

4. mapping, decoding and checking the data message, and finally

5. translating the datawords into an output stream of characters.

This decode algorithm presumes to be working on a binarized image, where every pixel value is either "0" for dark or "1" for light. (This is not to be confused with the convention of module value assignments.) If the stored image is in grayscale, then some thresholding technique must first be used to obtain an equivalent binary image. For evaluating symbol print quality, a global threshold shall be selected midway between the maximum and minimum reflectance values in the image, then each pixel assigned a value of "1" if it exceeds that threshold, "0" otherwise. Practical readers may well use a nonglobal threshold to handle uneven illumination and viewing conditions, and

may even employ inter-pixel interpolation to create a binary image with higher spatial resolution than that of the grayscale image, all to minimize the information content lost in this preprocessing step.

14.1 Finding candidate symbols

The “bullseye” finder pattern in Aztec Code has two distinct characteristics which may make it stand out from other objects in an image. First, topologically, its center module is highly “isolated” from the rest of the symbol, like an island within a lake that is within an island within a lake, and so forth, and thus can be made to stand out in a scan of the image which identifies connected regions. Annex C describes a scanning method for such a topology. Each horizontal line of the topological search algorithm which contains a section where there are at least 3 steps of monotonically increasing pattern followed by at least three steps of monotonically decreasing pattern (see row 9 of Figure C.7) shall be deemed to be a candidate for locating the bullseye. The highest value within this section shall be a candidate for being within the bullseye (pixel value 4 in row 9 of Figure C.7). This shall be the reference method for verification.

Equivalently, a scan that traces borders in the image would find the bullseye as one border wholely enclosed within another, which in turn is wholely enclosed within another, and so forth. Either of these approaches will work reliably to find the centers of candidate symbols, even in badly distorted images, so long as their finders are undamaged.

A second characteristic of the square bullseye which could be used by a practical reader is its assemblage of aligned rightangle corners all pointing directly outward from the center module. A scan through the image which detects clusters of such corners, and especially those attached to longer arms of other corners, should be able to find the bullseye even if partially damaged. In this manner, a bullseye that is 25 % obliterated but with at least two sets of corners intact can be reliably found.

14.2 Processing the bullseye image

The image of the finder shall be processed to locate as many of its corners as possible and matching them up with other corners in the same ring of the bullseye. The center module position shall be pinpointed as the average position of all opposing pairs of corners. The two major axes of the symbol shall be determined as the directions between adjacent pairs of corners. The module spacing along those axes shall be determined by the distances between adjacent corners.

It will be appreciated that the five critical starting parameters - the symbol’s exact center, the directions of its two major axes, and the nominal module spacings along those axes - are each indicated in multiple ways by features of the square bullseye finder, and can thus be determined for practical readers even if the finder is somewhat damaged.

14.3 Decoding the Core Symbol

Decoding the Core Symbol begins by mapping all the module centers within a square region centered on the bullseye and determining if they are light or dark. This information is used first to determine the video sign (normal or reversed) and format (compact or full-range) of the symbol, and then its orientation (that is, which direction is toward the top of the symbol) and possible mirror image reversal, and then to assemble, check, and parse the Mode Message.

14.3.1 Mapping and sampling module centers

The reference method for mapping the module centers over an Aztec Code symbol may be generally described as “two-dimensional crystal growing”. Starting with the center module’s position, a first layer of 8 module centers is projected surrounding it, and these positions are fine adjusted to fall evenly within the innermost (normally light) ring of the bullseye, then a next layer of 16 module centers is projected and fine adjusted to fall evenly within the next (normally dark) ring, and so forth. This process of projecting and adjusting module center positions relative to adjacent boundaries may conveniently be done in a spiral fashion outward one module layer at a time until the entire Core Symbol is mapped. The binary state at each of the module centers is sampled and stored in a bit map of the symbol.

14.3.2 Determining video sign and symbol format

If the 8 modules surrounding the finder's center are predominantly light, then decoding proceeds normally with dark regions equated to a binary "1"; otherwise decoding shall proceed presuming video reversal with light regions equated to a "1".

The fifth (11x11 square) ring of modules surrounding the finder's center is used to determine the symbol format: if it contains four or more binary "1" (normally dark) modules, then this is a compact Aztec Code symbol and mapping of the Core Symbol is done. Otherwise this is a full-range Aztec Code symbol and mapping of the Core Symbol must be extended outward for two more layers of modules.

14.3.3 Determining symbol orientation and mirror image reversal

In the outer layer of the Core Symbol, the 12 orientation bits at the corners are bitwise compared against the specified pattern in each of four possible orientations and their four mirror inverse orientations as well. If in any of the 8 cases checked as many as 9 of the 12 bits correctly match, that is deemed to be the correct orientation, otherwise decoding fails.

14.3.4 Decoding the Mode Message

Knowing the symbol format and orientation, the bits of the Mode Message are extracted, parsed into 4-bit binary values, and error decoded using GF(16). All of the syndrome values can be used for error correction since the tiny chance of an undetected error in the Mode Message will result in an eventual non-read, not a symbol misread. If error correction fails, decoding fails; otherwise, the values for the number of data layers L and for the number of datawords D are extracted from the Mode Message. In the special case where the most significant bit of D is set and D exceeds the capacity of the L data layers, then this symbol is identified as a reader initialization symbol and the most significant bit in D is cleared.

14.4 Decoding the data message

Like the Core Symbol, the data message is decoded first by mapping and sampling all the remaining module centers out to the limits of the symbol, then assembling and checking the spiralling sequence of codewords.

14.4.1 Mapping the data layers

For Aztec Code symbols with 4 or fewer layers of data, including all compact Aztec Code symbols, the two-dimensional crystal growing shall be extended outward to map all L layers of the data regions.

In larger full-range symbols, a method which maps first the reference grid's center positions is substantially stronger. The centers of the reference grid modules shall be plotted using the linear crystal growing algorithm described in Annex D. Once mapped, these grids provide sets of localized reference coordinates from which the center positions of all data modules are calculated and sampled.

Either method, two-dimensional crystal growing or reference grid mapping, ends up filling in a bit map which now covers the entire area of the symbol.

14.4.2 Assembling the codewords

As given in Table 1, the size and number of codewords in the symbol are determined by L, the number of data layers, which is encoded in the Mode Message. Those codewords can be directly assembled in reverse order - that is, starting with the final checkword and moving forward eventually to the first dataword - by sampling the symbol's bit map in an outward moving clockwise spiral which starts at the upper left corner of the Core Symbol. A simple systematic routine which samples two module layers in parallel, and for full-range symbols jumps past all modules where either the x or y coordinate is a multiple of 16, can assemble the entire codeword sequence for any size Aztec Code symbol.

14.4.3 Checking the codewords

The sequence of C_w codewords, of which D are data and K = C_w - D are checks, is conveniently checked and corrected using Massey-Berlekamp/Chien/Forney procedures, augmented to handle erasures efficiently, within the appropriate Galois Field GF(2^B) as indicated in Table 3 (See Annex B.3). Any datawords of all “0”s or all “1”s, as well as any codewords including modules which fall outside the image, shall be deemed erasures. The Massey-Berlekamp procedure starts by evaluating K syndrome values, and thus adapts easily to whatever number of checkwords a symbol happens to have. Two (or more, when errors predominate -- see B.2) checkwords shall be reserved for error detection, so that at most K-2 (or fewer) checkwords are utilized for error correction. If this error correction fails, then decoding fails.

14.5 Translating the datawords

In reverse of the message encoding procedure, translation of the datawords involves first converting them into a bit stream and then interpreting successive segments in that stream into output data characters.

14.5.1 Creating the data bit stream

Starting with the most significant bit of the first dataword, a stream of bits is assembled through to the least significant bit of the last dataword, taking care to omit the least significant bit of any word where the more significant bits are all “0” or all “1”.

14.5.2 Interpreting the bit stream

Interpretation into data characters is accomplished using Table 2 and starting in the Upper code set.

The bit stream is parsed into successive 5-bit, 4-bit, and occasional 11 and 8-bit values (under Byte Shift) as it latches and shifts between different code sets. When the remaining databits in the final codeword are all ones they shall be ignored. The output sequence of data characters ends when the bit stream ends.

If at the start of interpretation an Upper Latch is encoded before any data characters, then this symbol is identified as having a Structured Append header, which can be either processed within the reader or transmitted intact and signalled in the symbology identifier.

15 Symbol quality

Aztec Code symbols shall be assessed for quality using the two-dimensional matrix symbol print quality guidelines defined in ISO/IEC 15415, as augmented and modified below.

15.1 Symbol quality parameters

15.1.1 Fixed Pattern Damage

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

15.1.2 Axial nonuniformity

Aztec Code symbols are always nominally square, therefore the overall height H and width W of a symbol (instead of average grid spacing) as determined during the reference decode shall be used to evaluate Axial Nonuniformity, AN. The formula for AN becomes simply:

$$AN = abs(H - W)/((H + W)/2)$$

and this parameter shall be graded exactly as described in Clause 7.8.6 of ISO/IEC 15415:2004.

15.1.3 Unused error correction

The Reed-Solomon fields for the Mode Message and for the data message shall be graded independently as specified in Clause 7.8.8 of ISO/IEC 15415:2004, using p = 0 for the Mode Message and p = 2 for the data message. The Unused Error Correction grade shall be the lower achieved by those two fields.

15.1.4 "Print" Growth

"Print" Growth shall be assessed by checking that the "duty cycle" of lines extending through the finder and along the reference grid patterns (where available), nominally about 50 %, is between 30 % to 70 % inclusively.

Working independently from left to right through the symbol and then from top to bottom, proceed along lines passing through the symbol's center and aligned with each major axis summing the number of light (N_L) and dark (N_D) pixels encountered. In full-range symbols, traverse the full width or height of the symbol; in compact symbols, span just the finder pattern. Normalize the pixel counts by the number of light or dark modules they should have crossed, producing N_L' and N_D'. The resulting measure of Print Growth in each direction is:

$$D = N_D' / (N_L' + N_D')$$

which shall be graded against $D_{nom} = 0.50$ with $D_{min} = 0.30$ and $D_{max} = 0.70$ as follows:

$$D' = (D - D_{nom}) / (D_{max} - D_{nom}) \text{ if } D > D_{nom},$$

$$= (D - D_{nom}) / (D_{nom} - D_{min}) \text{ otherwise.}$$

The corresponding Print Quality grade is then:

A (4.0) if $-0.50 \le D' \le 0.50$

B (3.0) if $-0.70 \le D' \le 0.70$

C (2.0) if $-0.85 \le D' \le 0.85$

D (1.0) if $-1.00 \le D' \le 1.00$

F (0.0) if $D' < -1.00$ or $D' > 1.00$

The Print Growth grade shall be the lower of those achieved horizontally and vertically.

15.2 Symbol print quality grading

15.2.1 Image grade

The print quality grade for each image captured is the lowest of the parameter grades achieved per ISO/IEC 15415 and for Print Growth per 15.1.4 above. This is referred to as the Image Grade.

15.2.2 Symbol grade

Five or more Image Grades shall be collected for each symbol. These images should be of the symbol rotated through a full 360 degrees about the center point of the finder pattern. The increment of rotation for each image should be approximately equal.

The Symbol Grade is the average of the Image Grades from 15.2.1.

15.3 Process control measurements

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

1. Symbol Contrast readings from a linear bar code verifier.

2. Special reference symbols which facilitate visual inspection for grid irregularities and the measurement of Print Growth with a linear verifier.

3. Determination of Axial Nonuniformity by physical measurement.

4. Visual inspection of the finder pattern for critical defects.

These tools and methods are described in Annex I.

16 Transmitted data

16.1 Basic interpretation

No data shall be transmitted from special reader initialization symbols. The data from symbols within a Structured Append sequence may be buffered for later transmission within the full assembled message or may be transmitted with the appropriate symbology identifier modifier and sequencing header.

For all other Aztec Code symbols, the full data message string resulting from a proper decode shall be transmitted.

More complex interpretations are addressed below.

16.2 Protocol for FNC1

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

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

When FNC1 appears in any later position, it acts as a field separator and shall be represented in the transmitted message by the ASCII character <GS> (value 29).

16.3 Protocol for ECIs

In systems where ECIs are supported, the use of a symbology identifier prefix is required with every transmission. Whenever an ECI (FLG(n)) character is encountered, it shall be transmitted as the escape character 92 (or $5C_{HEX}$), which represents the character "\" (backslash or reverse solidus) in the default interpretation. The following "n" encoded digits are expanded with leading zeros to a 6-digit string which is transmitted as the appropriate ASCII values (48-57).

Application software recognizing "\nnnnnn" should interpret all subsequent characters as being from the ECI defined by that 6-digit sequence. This interpretation remains in effect until the end of the encoded data or until another ECI sequence is encountered.

If the byte value 92 occurs within encoded data (whether representing "\" or not), then two bytes of that value shall be transmitted. Thus a single occurrence always signals the ECI escape sequence and a double occurrence indicates true data.

Example:

Encoded data: A\\B\C

Transmission: A\\\\B\\C

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

16.4 Symbology identifier

Once the structure of the data (including the use of any ECI) has been identified, the appropriate symbology identifier shall be added by the decoder as a preamble to the transmitted data. The symbology identifier is required if ECIs appear within the transmitted data, if FNC1 is used as defined in 16.2, or if unbuffered Structured Append is required. See Annex F.

16.5 Transmitted data example

In this example, the two-character message " ¶ Ж " is to be encoded in Aztec Code. " ¶ " is represented by a byte value of 182 in Aztec Code's default character set (ECI 000003, which is equivalent to ISO/IEC 8859-1). " Ж " is a Cyrillic character not available in ECI 000003, but which can be represented in ISO/IEC 8859-5 (ECI 000007) by the same byte value of 182. The complete message can therefore be represented by inserting a switch to ECI 000007 after the first character, as follows:

The symbol encodes the message <¶> <Switch to ECI 000007> <Ж> using the following series of Aztec Code characters:

[B/S(1)],[182],[P/S],[FLG(1)],["7"],[B/S(1)],[182]

whose binary stream becomes (mimicking the notation above):

[11111(00001)],[10110110],[00000],[00000(001)],[1001],[11111(00001)],[10110110]

The decoder transmits the following bytes (including the symbology identifier prefix with an option value of 3, which indicates use of the ECI protocol):

93, 122, 51, 182, 92, 48, 48, 48, 48, 48, 55, 182

which, if viewed entirely in the default interpretation, would appear graphically as:

]z3¶\000007¶

Note that the decoder is responsible for signalling the switch to ECI 000007, but not for interpreting the result.

ECI-aware software in the receiving application would delete the ECI escape sequence \000007, and the Cyrillic character " Ж " would be represented in a system-dependent manner (e.g. by changing the font in a desktop-publishing file). The final result would match the original message of " ¶ Ж ".

Annex A
(normative)

Aztec Runes

Aztec Runes are a series of small but distinct machine-readable marks designed to be graphically compatible with Aztec Code. They are in fact just the Core Symbol of a compact Aztec Code symbol with a numerically distinct Mode Message which in this case conveys 8 bits of actual data. Thus they comprise 256 11x11 module square marks which are conveniently found and read by an Aztec Code reader.

A.1 Symbol description

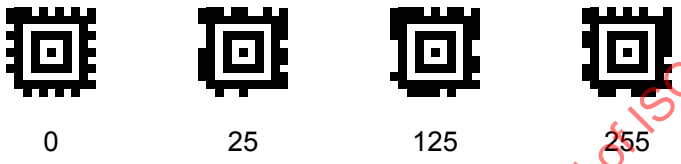

Figure A.1 — Representative Aztec Runes

Figure A.1 shows several representative runes, and the underlying structure of a Rune is shown in Figure A.2. Each Rune consists of a bullseye finder pattern, orientation patterns, and a Reed-Solomon encoded data message.

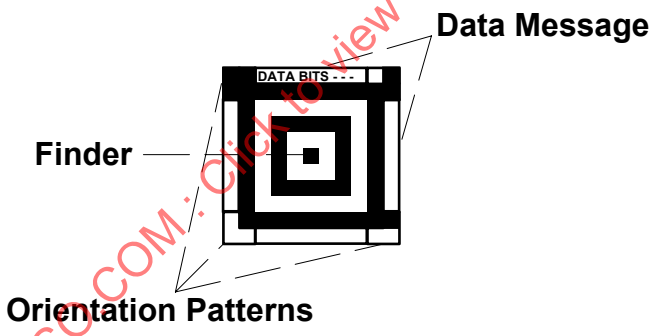

Figure A.2 — Structure of an Aztec Rune

The finder and orientation patterns in an Aztec Rune are exactly as defined for compact Aztec Code symbols in 5.1.1. The data message fills out the single layer of bits adjoining the finder exactly like the Mode Message described in 5.1.1.3. Like Aztec Code, no surrounding quiet zone is required.

A.2 Data message encoding

The 28 bits comprising a Rune's data message are subdivided into seven 4-bit words, of which two convey data and the other five are Reed-Solomon checkwords computed using GF(16), again exactly as defined for compact Aztec Code Mode Messages in 7.2.3. However, they are then distinguished from a normal Mode Message by inverting every other bit at the graphical level, equivalent to performing an Exclusive-OR of each word's value with the binary pattern "1010".

Thus when encoding, the 8-bit value to be encoded is parsed into two 4-bit words, which are then augmented by 5 checkwords using the procedures for a compact Mode Message, and then an exclusive OR operation is performed

on each of the seven word values with the value “1010” before being laid clockwise in a single layer wrapping around the finder with dark modules representing binary “1”s and light modules representing binary “0”s.

A.3 Data message decoding

The reference decode algorithm for Aztec Code, presented in Clause 14, is adequate for finding also Aztec Runes. The finder-based methods for detecting video reversal as well as the compact vs. wide-range format, and the use of the orientation patterns to determine image orientation as well as possible mirror-image reversal, shall all be applied. Rune decoding starts when error correction for the Mode Message of a compact symbol fails, and is very simple:

(1) Exclusive-OR each of the seven 4-bit “Mode Message” words with “1010”.

(2) Try Reed-Solomon decoding again.

If the Reed-Solomon decoding succeeds, then the leading eight bits of the data message in its current form (that is, with alternate bits inverted from what was represented graphically) convey the Rune’s encoded value.

A.4 Transmitted data

The value encoded in an Aztec Rune shall be transmitted as a 3-digit decimal value, with leading zeroes inserted as needed.

The symbology identifier, if used, shall be “]zC” where the modifier character of “C” signifies decoding an Aztec Rune instead of a normal Aztec Code symbol.

Annex B
(normative)

Error detection and correction

The error correction codewords in Aztec Code symbols shall be computed using Reed-Solomon error control encoding. All codewords are B bits in size, and the polynomial arithmetic within GF(2^B) is calculated using bit-wise modulo 2 arithmetic and word-wise modulo P arithmetic (where P is the equivalent value of the prime modulus polynomial). The values for B and P are as specified in 7.2.3 for the Mode Message and in 7.3.2 for the data message.

B.1 Generating the error correction codewords

The error correction codewords are the remainder after the polynomial formed by the "nd" data codewords is scaled upward by x^{nc} and then long divided by a generator polynomial of order "nc". It is impractical to list all the possible generator polynomials used with the data messages in Aztec Code; rather, they are generated directly within the appropriate software.

The C-language functions presented in Figure B.1 perform the needed error encoding. Operating on "nd" data codeword values stored in the integer array wd[], the function ReedSolomon() first generates log and antilog tables for the Galois Field of size "gf" with prime modulus "pp", then uses them in the function prod() to first calculate coefficients of the generator polynominal of order "nc", then to calculate "nc" additional check codewords which end up following the data in wd(). ReedSolomon() works equally well for the Mode Message as it does for the data message.

B.2 Error correction capacity

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

$$e + 2t < d - p$$

where:

e = Number of erasures

t = Number of errors

d = Number of error correction codewords

p = Number of ec codewords reserved for error detection

For Aztec Code's Mode Message, p = 0 (to maximize the possibility of continuing with the decode).

For the data message, p is normally equal to 2. However, if most of the error correction capacity is used to correct erasures, the possibility of undetected errors is increased. Whenever there are fewer than ten errors and the number of erasures is more than half the number of error correction codewords, then p is increased to 4.

Otherwise, the symbol cannot be decoded without risking a misdecode.

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

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

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

/* allocate, then generate the generator polynomial coefficients: */
  c = malloc(sizeof(int) * (nc)+1);
  for (i=1; i<=nc; i++) c[i] = 0; c[0] = 1;
  for (i=1; i<=nc; i++) {
     c[i] = c[i-1];
     for (j=i-1; j>=1; j--) {
        c[j] = c[j-1] ^ prod(c[j],alog[i],log,alog,gf);
     }
     c[0] = prod(c[0],alog[i],log,alog,gf);
  }

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

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


Figure B.1 — C function which generates Reed-Solomon checkwords

B.3 Error correction method

When the total number of erasures, e, is less than or equal to the error correction capacity, the recovery scheme is invoked. The erasures are substituted by zeros and the position of the q'th unknown codeword is j_q , for q = 1,2,...,e.

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

characters. Calculate i syndrome values S_0 through S_{i-1} by evaluating C(x) at $x = 2^k$ for k = 1 through i and where i is the number of error correction characters in the symbol. A circuit to generate the syndromes is shown in Figure B.2.

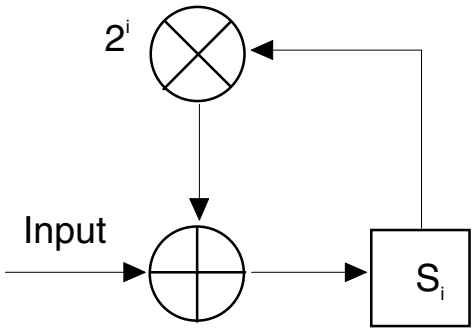

Figure B.2 — Symbol syndrome divider

Since the locations of erasures are known, j_q for q = 1, 2,...,e, the error location polynomial for these known positions can be computed:

$$\Lambda(x) = (1 - xX_1)\,(1 - xX_2)...(1 - xX_e)$$

$$= 1 + \Lambda_1 x + ... + \Lambda_e x^e$$

where $X_q = 2^{j_q}$

The error location polynomial, Λ(x), can be updated to include the position of errors. This can be done by using the Berlekamp-Massey Algorithm, an error and erasure decoder for BCH codes, as discussed in ***Theory and Practice of Error Control Codes***, by Richard E. Blahut (Addison Wesley, 1983). At this point verify that the number of erasures and errors satisfy the appropriate error correction capacity equation in Annex B.2.

Solving Λ(x) = 0 yields the position of the t errors, where t > 0; if t = 0 there is no error. Now, we only need to compute the error value, Y_{jp} for location j_q, where q = 1,...,e+t. To compute the error values, one auxiliary polynomial is needed, the Ω-polynomial, defined by:

$$\Omega(x) = 1 + (s_1 + \Lambda_1)x + (s_2 + \Lambda_1 s_1 + \Lambda_2)x^2 + ... + (s_\eta + \Lambda_1 s_{\eta-1} + \Lambda 2 s_{\eta-2} + ... + \Lambda_{\eta-1} s_1 + \Lambda_\eta)x^\eta$$

where η = e + t.

The error value at location j_q is thus given by:

$$Y_{jq} = \Omega(X_q^{-1}) \,/\, \left(X_q \prod_{i=1,\, i \neq q}^{\eta} (1 - X_i / X_q) \right)$$

After solving successfully for the error values, the complements of the error values are added to the corresponding locations of erroneous codewords.

Annex C
(normative)

Topological bullseye search algorithm

The topology of a bullseye can be made to stand out in a binary image using the following scanning method. It measures the "isolation" of each pixel from the image's white border, that is, a count of how many black/white and white/black edges must be crossed along any path to each border. Clearly, a bullseye's center is a region of high isolation.

The completely rigorous determination of pixel isolation for any arbitrary geometry (and the worst case would be establishing that the center of a long spiral connects to the border) requires multidirectional scanning which can be quite time consuming. Described here is an abbreviated single-pass method which is adequate for making a bullseye geometry stand out.

For illustration we will examine the steps in scanning the 24x21 pixel binary image section shown in Figure C.1. The basic approach is to create a secondary image with the same dimensions as the binary image but whose value at each pixel location indicates its isolation from the border. This can in fact be done one row at a time, using just a one dimensional integer array of the same width as the image, and sweeping downward through the image. Nonetheless, the isolation values will be shown in the following figures overlying the pixel locations in a two dimensional array.

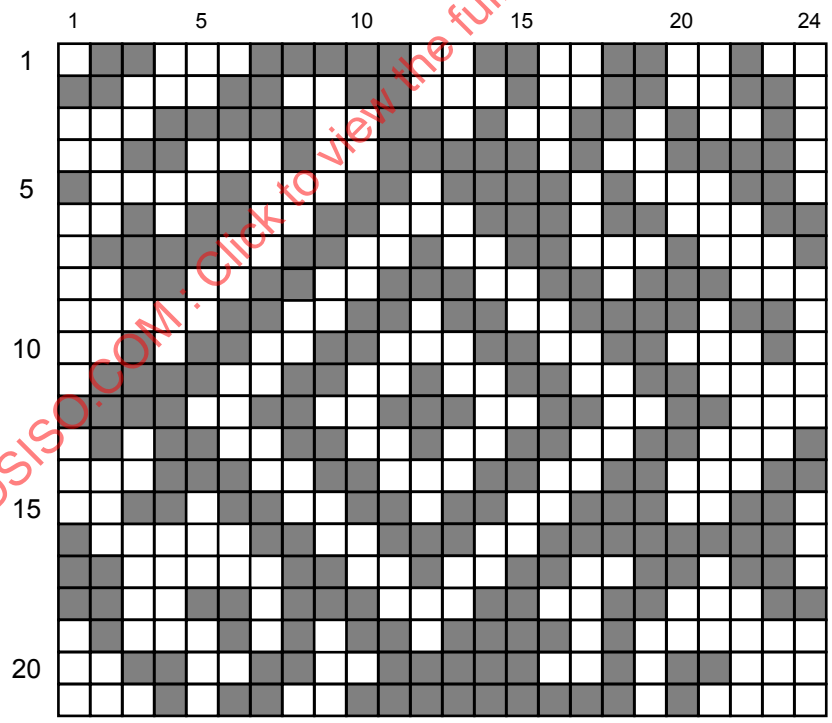

Figure C.1 — The binary image to be searched

The key topological rule is that the isolation values must always be even on light pixels and odd on dark pixels. The initial step then is to fill the integer array with the values "0" for a light pixel and "1" for a dark pixel for the pixels along the top row of the image, as shown in Figure C.2.

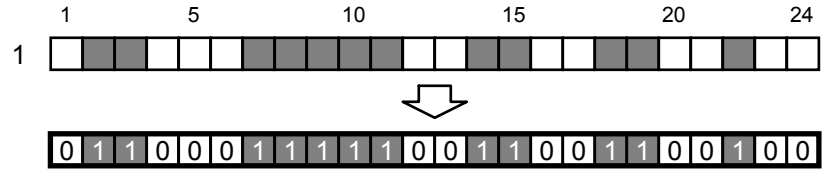

Figure C.2 — Initial integer array contents

Each successive row is then scanned using a three step process. First the integer array's contents are updated to the new row by incrementing any entry whose color changes. This is shown in the second row of Figure C.3, where several "0"s have become "1"s and several "1"s have become "2"s. Next the leftmost value in the array is reset to "0" or "1", then the array is swept from left to right and any value more than 1 above its lefthand neighbor is reduced by 2s until this is not so; as shown in the middle isolated row, the "2" in column 14 has been reduced to "0". Lastly the rightmost value in the array is reset to "0" or "1", then the array is swept from right to left and any value more than 1 above its righthand neighbor is reduced by multiples of 2 so that its value is equal to or less than the previous pixel value plus one; in the lower isolated row, the "2" in column 3 has also been lowered to "0".

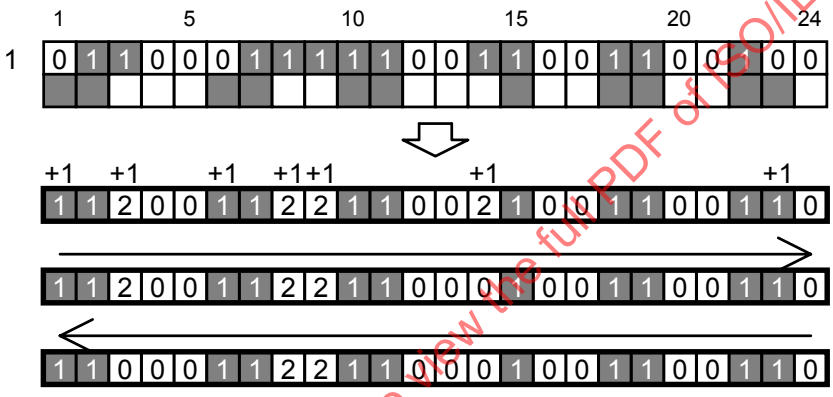

Figure C.3 — The steps and results of scanning row 2

Figures C.4 through C.6 show row-by-row progress of the scan.

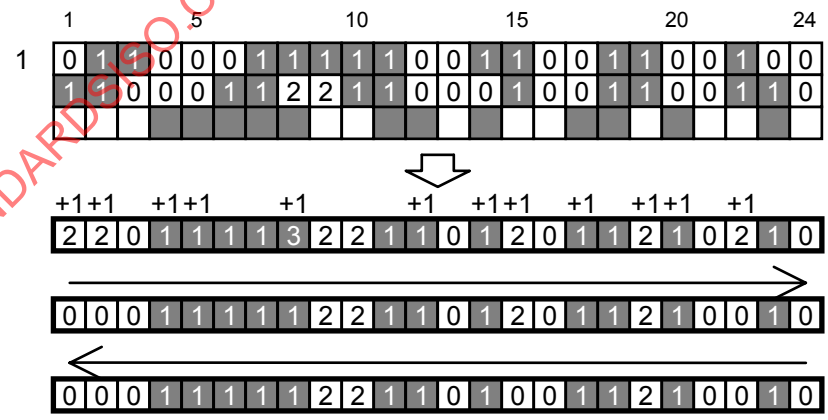

Figure C.4 — The steps and results of scanning row 3

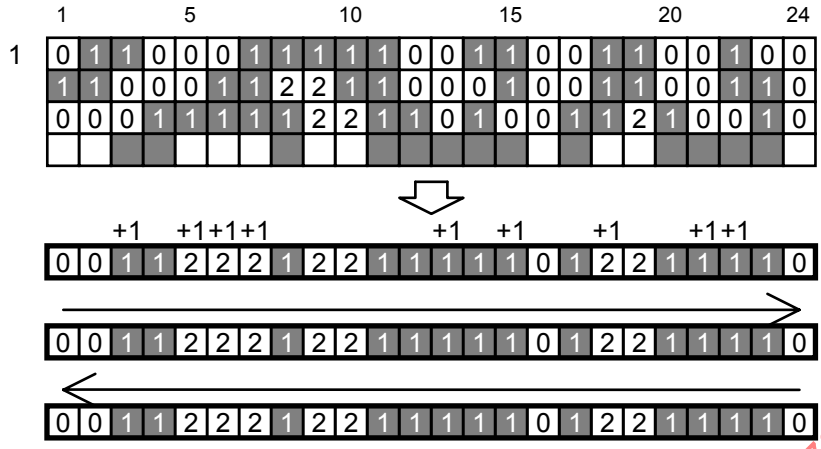

Figure C.5 — The steps and results of scanning row 4

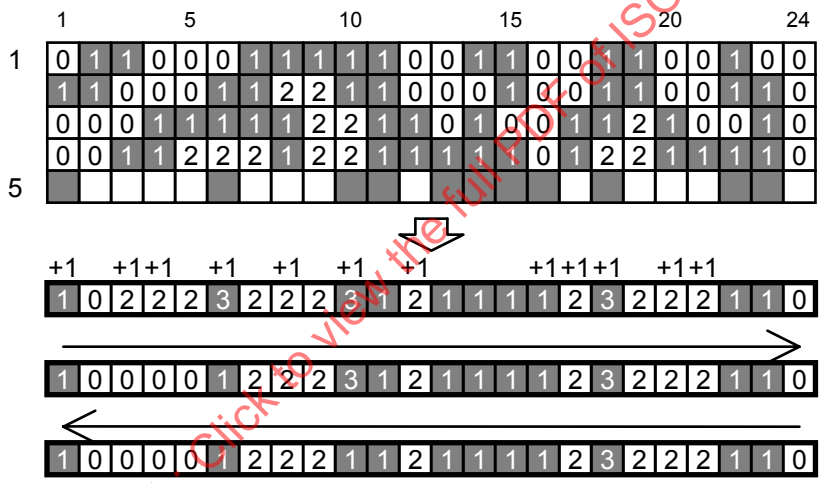

Figure C.6 — The steps and results of scanning row 5

In course of processing row 3 a value of “3” briefly appears in column 8 but it is “swept away” in the rightward sweep. Also note the leading 2 is changed to a zero in row 3.

In the course of processing row 5 a “3” briefly appears in column 6 but is swept away after the “2” guarding its left side is also swept away. The “3” in column 18, on the other hand, persists.

Figure C.7 shows the integer array’s contents after six more iterations. The bullseye topology has emerged in the middle as a region of relatively high isolation values. Although high values do identify its center, the most reliable indication of the “height” of the bullseye is the number of steps monotonically leading up to the peak value and then leading down from it. A state machine that monitors successive integer values during the final right-to-left sweep on each row can detect such bullseye centers and measure their height.

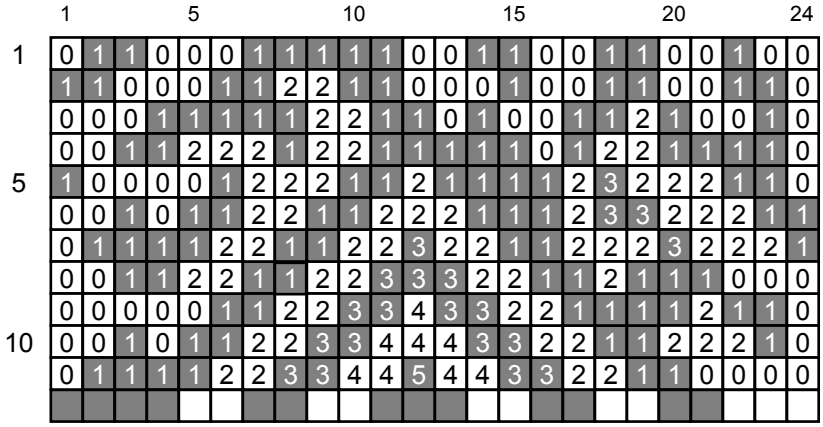

Figure C.7 — The steps and results through row 11

Figure C.8 shows the integer array's contents after six more iterations. The peak (i.e. center) of this bullseye has clearly occurred in columns 11 to 13 of rows 11 to 13, though other lesser local "elevations" (such as around column 20 in row 7) may be found in the rest of the image. The scan of a full image is likely to locate the center positions of several candidate bullseyes, and further two-dimensional graphical analysis is needed qualify them as an Aztec Code finder pattern.

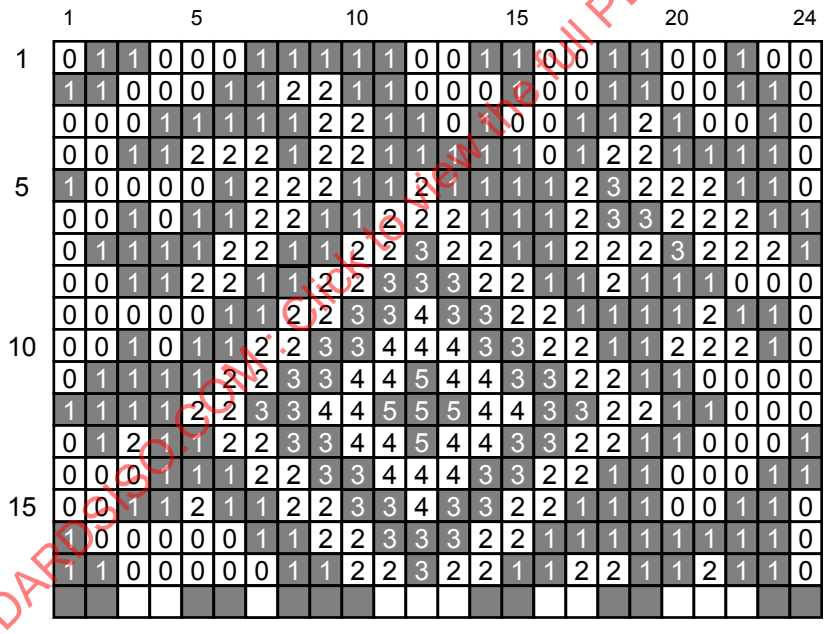

Figure C.8 — The steps and results through row 17

Figure C.8 also illustrates one of the inspirations for the name "Aztec Code". The square central bullseye of an Aztec Code symbol jumps numerically out of this image much as the ancient Aztec pyramids seem to rise vertically out of the undergrowth. The bullseye structures in some other matrix codes will also be detected in this manner, although this is not the reference finding method for those symbols.

Annex D
(normative)

Linear crystal growing algorithm

Use the measured vertical and horizontal pitch derived from the bullseye to set an initial estimate of both the vertical and horizontal X dimensions. Using the direction of the vertical and horizontal edges in the bullseye, determine an initial estimate of both the vertical and horizontal axis directions.

For each of the four reference grids originating at the center of the bullseye, plot the center points of the grid's modules using the “crystal growing” technique starting with the center of the bullseye and proceeding to the edge of the symbol.

First plot the estimated center of the next module one X from the center of the bullseye along the appropriate axis. Plot four 1X long sample lines projecting to the “compass points” from the estimated module center along the axes. Conditionally adjust the center both vertically and horizontally using the following method for each axis.

Along the two lines parallel to the axis:

a. If neither line crosses a color edge appropriate for the module’s nominal color (e.g. a dark to light transition for a dark module), do not adjust the center along that axis.

b. If only one line crosses an appropriate color edge, adjust the center 5 % of the distance from its original position to a point one-half X from the edge.

c. Otherwise, both lines cross an appropriate color edge, adjust the center 5 % of the distance from its original position to the mid-point of the two edges.

After calculating all 16 modules in a reference pattern subsection, recalculate the axis direction and X dimension, using those module centers, before proceeding into the next subsection using the previously calculated positions.

Starting from the adjusted point, plot a new point one X further along the axis and adjust it using the method above. Repeat this step to plot the center point for each module.

For reference grid segments not passing through the bullseye plot their centers starting from the module which intersects the appropriate one of the four originally plotted reference grids.

First, revise the direction of the axis and X dimension along the axis to match the center points of the closest already plotted reference pattern’s subsection. Then, starting with the estimated center of the intersecting module, repeat the crystal growing in both directions along the new reference pattern to the ends of the symbol.

Annex E
(normative)

Fixed Pattern Damage grading

E.1 Features to be assessed

For Fixed Patterns, “Compact” Aztec Code symbols have a 2-ring 9-module square Bullseye with 3-module Orientation Patterns in each corner, totaling 93 modules at the symbol’s center. Similarly, “Full-Range” Aztec Code symbols have a 3-ring 13-module square Bullseye and the Orientation Patterns, totaling 181 modules, plus a Reference Grid that runs throughout the data regions on 16-module centers.

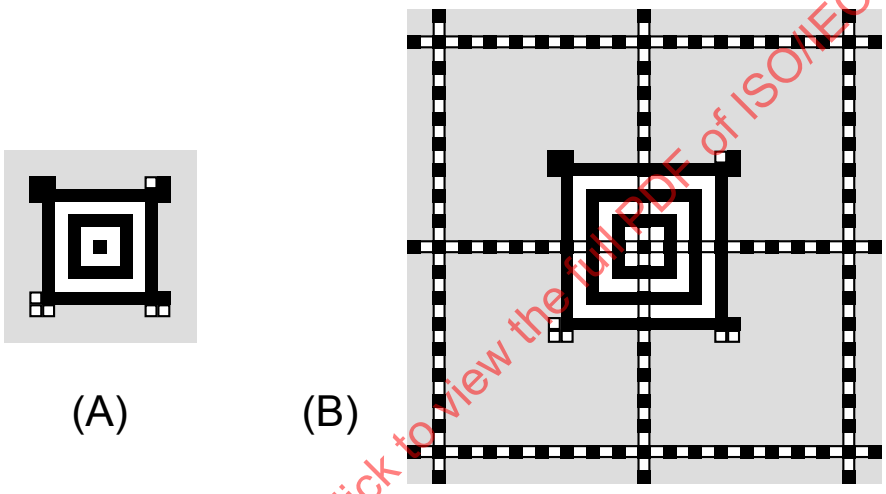

Figure E.1 — The Fixed Patterns within (A) compact and (B) full-range Aztec Code symbols

In both cases, the Bullseye & Orientation Patterns are taken together as Segment A. In addition, each of the light and dark rings of the Bullseye, and also its center module, are regarded as individual sub-segments. (The orientation patterns are not part of any sub-segment.)

In Full-Range symbols, the Reference Grid (including those of its modules within the Bullseye) is broken into individual vertical and horizontal “ladders”, each of which is regarded as a Segment B. The modules at the crossing points are counted in both ladders. The example in Figure E.1(B) has six such ladders or segments.

E.2 Grading criteria and assignments

Grading is performed based on a count of the number of erroneous (wrong color) modules in each segment or sub-segment as follows:

Segment A (Bullseye & Orientation patterns) is graded as:

A (4) if 0 (zero) modules are in error

B (3) if 1 module is in error

C (2) if 2 modules are in error, at most within a single sub-segment

D (1) if 3 modules are in error, at most within a single sub-segment

F (0) if 4 or more modules are in error -OR- errors occur in more than one sub-segment

Each Segment B (within the Reference grid) is graded as:

A (4) if 0 (zero) modules are in error, else

B (3) if 7 % or fewer modules are in error, else

C (2) if 11 % or fewer modules are in error, else

D (1) if 14 % or fewer modules are in error, else

F (0) .

The overall Fixed Pattern Damage grade is the lowest of the Segment grades achieved.