

INTERNATIONAL STANDARD

ISO/IEC
8632-2

Second edition
1992-10-01

Information technology — Computer graphics — Metafile for the storage and transfer of picture description information —

Part 2: Character encoding

*Technologies de l'information — Infographie — Métafichier de stockage
et de transfert des informations de description d'images —*

Partie 2: Codage des caractères



Reference number
ISO/IEC 8632-2:1992(E)



Documents
Document Sharing Hub

CONTENTS

1	Scope	1
2	Normative references	2
3	Notational conventions	3
3.1	7-Bit and 8-Bit code tables	3
3.2	Code extension techniques vocabulary	4
3.2.1	C0 sets	4
3.2.2	C1 sets	4
3.2.3	G-sets	4
4	Entering and leaving the metafile environment	7
4.1	Implicitly entering the metafile environment	7
4.2	Designating and invoking the CGM coding environment from ISO 2022	7
5	Method of encoding opcodes	8
5.1	Encoding technique of the basic opcode set	8
5.2	Extension mechanism	8
5.3	Opcode assignments	9
6	Method of encoding parameters	14
6.1	Basic format	14
6.2	Bitstream format	15
6.3	Coding integers	16
6.4	Coding real numbers	16
6.5	Coding VDCs and points	18
6.6	Coding point list parameters	18
6.6.1	Displacement mode	18
6.6.2	Incremental mode	19
6.6.3	Incremental mode encoding	22
6.7	Colour specifiers	23
6.8	Colour lists	24
6.8.1	Normal format (coding type=0)	25
6.8.2	Bitstream format (coding type=1)	25
6.8.3	Runlength format (coding type=2)	25
6.8.4	Runlength bitstream format (coding type=3)	25
6.8.5	Examples	26
6.9	String parameters	27
6.9.1	Overall string parameter format	27
6.9.2	Bit combinations permitted within string parameters of text elements	27
6.9.3	C0 control within string parameters	28
6.9.4	Using G-sets in string parameters	28

© ISO/IEC 1992

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland
Printed in Switzerland

6.10	Enumerated parameters	29
6.11	Index parameters	29
6.12	Data record parameters	29
6.13	Coding VCs and viewport point parameters	29
6.14	Name parameters	30
6.15	Compressed bitstream operands	30
6.16	Structured data record operands	30
6.17	Glyph mapping	30
7	Character substitution	31
8	Representation of each element	33
8.1	Delimiter elements	35
8.1.1	BEGIN METAFILE	35
8.1.2	END METAFILE	35
8.1.3	BEGIN PICTURE	35
8.1.4	BEGIN PICTURE BODY	35
8.1.5	END PICTURE	35
8.1.6	BEGIN SEGMENT	35
8.1.7	END SEGMENT	36
8.1.8	BEGIN FIGURE	36
8.1.9	END FIGURE	36
8.1.10	BEGIN PROTECTION REGION	36
8.1.11	END PROTECTION REGION	36
8.1.12	BEGIN COMPOUND LINE	36
8.1.13	END COMPOUND LINE	36
8.1.14	BEGIN COMPOUND TEXT PATH	36
8.1.15	END COMPOUND TEXT PATH	36
8.1.16	BEGIN TILE ARRAY	36
8.1.17	END TILE ARRAY	37
8.2	Metafile descriptor elements	38
8.2.1	METAFILE VERSION	38
8.2.2	METAFILE DESCRIPTION	38
8.2.3	VDC TYPE	38
8.2.4	INTEGER PRECISION	38
8.2.5	REAL PRECISION	38
8.2.6	INDEX PRECISION	39
8.2.7	COLOUR PRECISION	39
8.2.8	COLOUR INDEX PRECISION	40
8.2.9	MAXIMUM COLOUR INDEX	40
8.2.10	COLOUR VALUE EXTENT	40
8.2.11	METAFILE ELEMENT LIST	40
8.2.12	METAFILE DEFAULTS REPLACEMENT	41
8.2.13	FONT LIST	41
8.2.14	CHARACTER SET LIST	41
8.2.15	CHARACTER CODING ANNOUNCER	42
8.2.16	NAME PRECISION	42
8.2.17	MAXIMUM VDC EXTENT	42
8.2.18	SEGMENT PRIORITY EXTENT	42
8.2.19	COLOUR MODEL	42
8.2.20	COLOUR CALIBRATION	42
8.2.21	FONT PROPERTIES	44
8.2.22	GLYPH MAPPING	46
8.2.23	SYMBOL LIBRARY LIST	47

ISO/IEC 8632-2:1992 (E)

8.3	Picture descriptor elements	48
8.3.1	SCALING MODE	48
8.3.2	COLOUR SELECTION MODE	48
8.3.3	LINE WIDTH SPECIFICATION MODE	48
8.3.4	MARKER SIZE SPECIFICATION MODE	48
8.3.5	EDGE WIDTH SPECIFICATION MODE	48
8.3.6	VDC EXTENT	49
8.3.7	BACKGROUND COLOUR	49
8.3.8	DEVICE VIEWPORT	49
8.3.9	DEVICE VIEWPORT SPECIFICATION MODE	49
8.3.10	DEVICE VIEWPORT MAPPING	49
8.3.11	LINE REPRESENTATION	50
8.3.12	MARKER REPRESENTATION	50
8.3.13	TEXT REPRESENTATION	50
8.3.14	FILL REPRESENTATION	51
8.3.15	EDGE REPRESENTATION	51
8.3.16	INTERIOR STYLE SPECIFICATION MODE	52
8.3.17	LINE AND EDGE TYPE DEFINITION	52
8.3.18	HATCH STYLE DEFINITION	52
8.3.19	GEOMETRIC PATTERN DEFINITION	53
8.4	Control elements	54
8.4.1	VDC INTEGER PRECISION	54
8.4.2	VDC REAL PRECISION	54
8.4.3	AUXILIARY COLOUR	55
8.4.4	TRANSPARENCY	55
8.4.5	CLIP RECTANGLE	55
8.4.6	CLIP INDICATOR	55
8.4.7	LINE CLIPPING MODE	55
8.4.8	MARKER CLIPPING MODE	56
8.4.9	EDGE CLIPPING MODE	56
8.4.10	NEW REGION	56
8.4.11	SAVE PRIMITIVE CONTEXT	56
8.4.12	RESTORE PRIMITIVE CONTEXT	56
8.4.13	PROTECTION REGION INDICATOR	56
8.4.14	GENERALIZED TEXT PATH MODE	57
8.4.15	MITRE LIMIT	57
8.4.16	TRANSPARENT CELL COLOUR	57
8.5	Graphical primitive elements	58
8.5.1	POLYLINE	58
8.5.2	DISJOINT POLYLINE	58
8.5.3	POLYMARKER	58
8.5.4	TEXT	58
8.5.5	RESTRICTED TEXT	58
8.5.6	APPEND TEXT	59
8.5.7	POLYGON	59
8.5.8	POLYGON SET	59
8.5.9	CELL ARRAY	59
8.5.10	GENERALIZED DRAWING PRIMITIVE	61
8.5.11	RECTANGLE	61
8.5.12	CIRCLE	61
8.5.13	CIRCULAR ARC 3 POINT	61
8.5.14	CIRCULAR ARC 3 POINT CLOSE	61
8.5.15	CIRCULAR ARC CENTRE	62

8.5.16	CIRCULAR ARC CENTRE CLOSE	62
8.5.17	ELLIPSE	62
8.5.18	ELLIPTICAL ARC	62
8.5.19	ELLIPTICAL ARC CLOSE	63
8.5.20	CIRCULAR ARC CENTRE REVERSED	63
8.5.21	CONNECTING EDGE	63
8.5.22	HYPERBOLIC ARC	63
8.5.23	PARABOLIC ARC	63
8.5.24	NON-UNIFORM B-SPLINE	64
8.5.25	NON-UNIFORM RATIONAL B-SPLINE	64
8.5.26	POLYBEZIER	64
8.5.27	POLYSYMBOL	65
8.5.28	BITONAL TILE	65
8.5.29	TILE	66
8.6	Attribute elements	67
8.6.1	LINE BUNDLE INDEX	67
8.6.2	LINE TYPE	67
8.6.3	LINE WIDTH	67
8.6.4	LINE COLOUR	67
8.6.5	MARKER BUNDLE INDEX	67
8.6.6	MARKER TYPE	67
8.6.7	MARKER SIZE	68
8.6.8	MARKER COLOUR	68
8.6.9	TEXT BUNDLE INDEX	68
8.6.10	TEXT FONT INDEX	68
8.6.11	TEXT PRECISION	68
8.6.12	CHARACTER EXPANSION FACTOR	68
8.6.13	CHARACTER SPACING	69
8.6.14	TEXT COLOUR	69
8.6.15	CHARACTER HEIGHT	69
8.6.16	CHARACTER ORIENTATION	69
8.6.17	TEXT PATH	69
8.6.18	TEXT ALIGNMENT	69
8.6.19	CHARACTER SET INDEX	70
8.6.20	ALTERNATE CHARACTER SET INDEX	70
8.6.21	FILL BUNDLE INDEX	70
8.6.22	INTERIOR STYLE	70
8.6.23	FILL COLOUR	71
8.6.24	HATCH INDEX	71
8.6.25	PATTERN INDEX	71
8.6.26	EDGE BUNDLE INDEX	71
8.6.27	EDGE TYPE	71
8.6.28	EDGE WIDTH	72
8.6.29	EDGE COLOUR	72
8.6.30	EDGE VISIBILITY	72
8.6.31	FILL REFERENCE POINT	72
8.6.32	PATTERN TABLE	72
8.6.33	PATTERN SIZE	72
8.6.34	COLOUR TABLE	73
8.6.35	ASPECT SOURCE FLAGS	73
8.6.36	PICK IDENTIFIER	74
8.6.37	LINE CAP	74
8.6.38	LINE JOIN	74

ISO/IEC 8632-2:1992 (E)

8.6.39	LINE TYPE CONTINUATION	74
8.6.40	LINE TYPE INITIAL OFFSET	75
8.6.41	TEXT SCORE TYPE	75
8.6.42	RESTRICTED TEXT TYPE	75
8.6.43	INTERPOLATED INTERIOR	75
8.6.44	EDGE CAP	76
8.6.45	EDGE JOIN	76
8.6.46	EDGE TYPE CONTINUATION	76
8.6.47	EDGE TYPE INITIAL OFFSET	77
8.6.48	SYMBOL LIBRARY INDEX	77
8.6.49	SYMBOL COLOUR	77
8.6.50	SYMBOL SIZE	77
8.6.51	SYMBOL ORIENTATION	77
8.7	Escape elements	78
8.7.1	ESCAPE	78
8.7.2	DOMAIN RING	78
8.8	External elements	79
8.8.1	MESSAGE	79
8.8.2	APPLICATION DATA	79
8.9	Segment elements	80
8.9.1	COPY SEGMENT	80
8.9.2	INHERITANCE FILTER	80
8.9.3	CLIP INHERITANCE	82
8.9.4	SEGMENT TRANSFORMATION	82
8.9.5	SEGMENT HIGHLIGHTING	82
8.9.6	SEGMENT DISPLAY PRIORITY	83
8.9.7	SEGMENT PICK PRIORITY	83
9	Defaults	84
10	Conformance	85
A	Formal grammar	86

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. 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.

International Standard ISO/IEC 8632-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

This second edition cancels and replaces the first edition (ISO 8632-2:1987), which has been technically revised.

ISO/IEC 8632 consists of the following parts, under the general title *Information technology – Computer graphics – Metafile for the storage and transfer of picture description information* :

Part 1: Functional specification

Part 2: Character encoding

Part 3: Binary encoding

Part 4: Clear text encoding

Annex A forms an integral part of this part of ISO/IEC 8632.

Introduction

0.1 Purpose of the character encoding

The Character Encoding of the Computer Graphics Metafile (CGM) provides a representation of the Metafile syntax intended for situations in which it is important to minimize the size of the metafile or transmit the metafile through character-oriented communications services. The encoding uses compact representation of data that is optimized for storage or transfer between computer systems.

If minimizing the processing overhead is more important than data compaction, an encoding such as the Binary Encoding contained in ISO/IEC 8632-3 may be more appropriate. If human readability is the most important criterion, an encoding such as the Clear Text Encoding in ISO/IEC 8632-4 may be more appropriate.

0.2 Objectives

This encoding was designed with the following objectives:

- a) regular syntax: All elements of the metafile should be encoded in a uniform way so that parsing the metafile is simple;
- b) compactness: The encoding should provide a highly compact metafile, suitable for systems with restricted storage capacity or transfer bandwidth;
- c) extensibility: the encoding should allow for future extensions;
- d) transportability: the encoding should be suitable for use with transport mechanisms designed for character-oriented data based on a standard national character set derived from ISO/IEC 646.

0.3 Metafile characteristics

Each CGM command follows a simple regular syntax. Thus, new commands can be added in a future revision of ISO/IEC 8632 such that existing CGM interpreters can recognize (and ignore) the new commands. Also, new operands can be added to existing commands in the future revision of the standard such that existing CGM interpreters can recognize (and ignore) the additional operands.

Each CGM operand follows a simple regular syntax. Operands are variable in length. This permits small values to be represented by the smallest number of bytes.

Metafile characteristics**Introduction**

A certain range of operand values of standard commands have been reserved for private use; the remaining range is either standardized or reserved for future standardization.

0.4 Relationship to other International Standards

The Character Encoding has been developed in collaboration with the ISO subcommittee responsible for character sets and coding, ECMA, and CEPT. The encoding conforms to the rules for code extension specified in ISO 2022 in the category of complete coding system.

The representation of character data in this part of ISO/IEC 8632 follows the rules of ISO/IEC 646 and ISO 2022.

For certain elements, the CGM defines value ranges as being reserved for registration. The values and their meanings will be defined using the established procedures (see ISO/IEC 8632-1, sub-clause 4.12.)

Information technology – Computer graphics – Metafile for the storage and transfer of picture description information –

Part 2 : Character encoding

1 Scope

This part of ISO/IEC 8632 specifies a character encoding of the Computer Graphics Metafile. For each of the elements specified in ISO/IEC 8632-1 an encoding is specified.

This encoding of the Computer Graphics Metafile provides a highly compact representation of the metafile, suitable for applications that require the metafile to be of minimum size and suitable for transmission with character-oriented transmission services.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 8632. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 8632 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

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

ISO 2022:1986, *Information processing – ISO 7-bit and 8-bit coded character sets – Code extension techniques*.

ISO 2375:1985, *Data processing – Procedure for registration of escape sequences*.

ISO 6429:1988, *Information processing – Control functions for 7-bit and 8-bit coded character sets*.

ECMA 96, *Graphics Data Syntax for a multiple Workstation Interface*.

CEPT, *Revision of T/CD 6.1 Videotex Presentation Layer Data Syntax*.

3 Notational conventions

3.1 7-Bit and 8-Bit code tables

The bits of the bit combinations of the 7-bit code are identified by b7, b6, b5, b4, b3, b2, and b1, where b7 is the highest-order, or most-significant, bit and b1 is the lowest-order, or least-significant, bit.

The bit combinations may be interpreted to represent integers in the range 0 to 127 in binary notation by attributing the following weights to the individual bits:

Bit:	b7	b6	b5	b4	b3	b2	b1
Weight:	64	32	16	8	4	2	1

In this part of ISO/IEC 8632, the bit combinations are identified by notation of the form x/y , where x is a number in the range 0 to 7 and y is a number in the range 0 to 15. The correspondence between the notations of the form x/y and the bit combinations consisting of the bits b7 to b1 is as follows:

- x is the number represented by b7, b6, and b5 where these bits are given the weights 4, 2, and 1 respectively;
- y is the number represented by b4, b3, b2, and b1 where these bits are given the weights 8, 4, 2, and 1 respectively.

The notations of the form x/y are the same as those used to identify code table positions, where x is the column number and y is the row number.

A 7-bit code table consists of 128 positions arranged in eight columns and sixteen rows. The columns are numbered 0 to 7 and the rows are numbered 0 to 15. Figure 1 shows a 7-bit code table.

An example illustrates the 7-bit code: "1/11" refers to the bit combination in column 1, row 11 of the code table, binary 0011011.

The bits of the bit combinations of the 8-bit code are identified by b8, b7, b6, b5, b4, b3, b2, and b1, where b8 is the highest-order, or most-significant, bit and b1 is the lowest-order, or least-significant, bit.

The bit combinations may be interpreted to represent integers in the range 0 to 255 in binary notation by attributing the following weights to the individual bits:

Bit:	b8	b7	b6	b5	b4	b3	b2	b1
Weight:	128	64	32	16	8	4	2	1

Using these weights, the bit combinations of the 8-bit code are interpreted to represent numbers in the range 0 to 255.

In this part of ISO/IEC 8632, the bit combinations are identified by notation of the form xx/yy , where xx and yy are numbers in the range 00 to 15. The correspondence between the notations of the form xx/yy and the bit combinations consisting of the bits b8 to b1 is as follows:

- xx is the number represented by b8, b7, b6, and b5 where these bits are given the weights 8, 4, 2, and 1 respectively;
- yy is the number represented by b4, b3, b2, and b1 where these bits are given the weights 8, 4, 2, and 1 respectively.

The notations of the form xx/yy are the same as those used to identify code table positions, where xx is the column number and yy is the row number. An 8-bit code table consists of 256 positions arranged in sixteen columns and sixteen rows. The columns and rows are numbered 00 to 15. Figure 2 shows an 8-bit code

ISO/IEC 8632-2:1992 (E)

7-Bit and 8-Bit code tables

Notational conventions

table.

An example illustrates the 8-bit code: 04/01 represents the 8-bit byte 01000001, whereas 4/1 represents the 7-bit byte 1000001.

3.2 Code extension techniques vocabulary

In describing the characters that may occur within string parameters, certain terms imported from other standards (e.g., ISO 2022) are useful. In the context of the CGM, these terms, and the concepts to which they refer, apply only within the string parameters of the TEXT, APPEND TEXT, and RESTRICTED TEXT metafile elements.

3.2.1 C0 sets

A C0 set is a set of 30 control characters represented in a 7-bit code by 0/0 to 1/15, except 0/14 and 0/15 which shall be unused, and in an 8-bit code by 00/00 to 01/15, except 00/14 and 00/15 which shall be unused. C0 sets occupy columns 0 and 1 of a 7-bit code table or columns 00 and 01 of an 8-bit code table. The meanings of C0 controls within string parameters are described in 6.9.3.

3.2.2 C1 sets

A C1 set is a set of up to 32 control characters represented by bit combinations 08/00 to 09/15 in an 8-bit code. C1 sets occupy columns 08 and 09 of the 8-bit code table. In a 7-bit code the C1 control functions are represented by 2-byte escape sequences. This CGM encoding reserves the bit combinations 9/8 and 9/12 (ESC 5/8 and ESC 5/12 in a 7-bit environment, ESC = 1/11); these shall not be part of the content of string parameters. Other C1 control characters from other standards, such as ISO 6429, may be used within string parameters by agreement between the interchanging parties.

3.2.3 G-sets

The G-sets (G0, G1, G2, G3) are coded character sets of 94 or 96 characters. CHARACTER SET INDEX designates which character set is to be the G0 set. ALTERNATE CHARACTER SET INDEX designates a character set to be used as both the G1 and G2 sets. The G-sets may be "invoked into" (caused to occupy) columns 2 through 7 of a 7-bit code table, or columns 02 through 07 and 10 through 15 of an 8-bit code table. This encoding of the CGM uses the G0 and G1/G2 sets within string parameters. The G3 set may be used within the string parameters of conforming metafiles; this requires selection of the extended 7-bit or extended 8-bit mode in the CHARACTER CODING ANNOUNCER. The CGM does not provide an element to explicitly designate the G3 sets; this may be done within a text string in accordance with ISO 2022, or by other means agreed upon by the interchanging parties.

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	C0 SET		02/0								10/0					
1																
2																
3																
4																
5																
6																
7																
8																
9																
10																
11																
12																
13																
14																
15																
								07/15							15/15	

Figure 2 — The 8-bit code table

4 Entering and leaving the metafile environment

4.1 Implicitly entering the metafile environment

The CGM coding environment may be entered implicitly, by agreement between the interchanging parties. This is suitable only if there is not to be any interchange with services using other coding techniques.

4.2 Designating and invoking the CGM coding environment from ISO 2022

For interchange with services using the code extension techniques of ISO 2022, the CGM coding environment shall be designated and invoked from ISO 2022 environment by the following escape sequence:

ESC 2/5 F

where ESC is the bit combination 1/11, and F refers to a bit combination that will be assigned by the ISO Registration Authority for ISO 2375.

The first bit combination occurring after this escape sequence will then represent the opcode of a CGM metafile element.

After the end of one or more metafiles (i.e., after the END METAFILE element) or between pictures (i.e., after the END PICTURE element), the following escape sequence may be used to return to the ISO 2022 coding environment:

ESC 2/5 4/0

This not only returns to the ISO 2022 coding environment, but also restores the designation and invocation of coded character sets to the state that existed prior to entering the CGM coding environment with the ESC 2/5 F sequence. (The terms "designation" and "invocation" are defined in ISO 2022.)

5 Method of encoding opcodes

Each metafile element is composed of one opcode and parameters as required. The opcodes are coded as a sequence of bit combinations from columns 2 and 3 of the code chart. The encoding technique supplies:

- the basic opcode set;
- extension opcode sets.

5.1 Encoding technique of the basic opcode set

The basic opcode set consists of single-byte and double-byte opcodes. Single-byte opcodes are from column 2 of the code chart. Bits b4 to b1 are used to encode the opcode. The format is as follows:

```

b8          b1
+--+-----+-----+
|X|0 1 0|b b b b|
+--+-----+-----+

```

The "X" bit (bit b8) is the parity bit (or omitted bit) in a 7-bit environment. In an 8-bit environment it is 0. For double-byte opcodes the first byte is from column 3 and the second byte is from column 2 or 3 of the code chart. Bits b4 to b1 of the first byte and bits b5 to b1 of the second byte are used to encode the opcode:

```

b8          b1  b8          b1
+--+-----+-----+ +--+-----+-----+
|X|0 1 1|b b b b| |X|0 1|b b b b b|
+--+-----+-----+ +--+-----+-----+

```

The bit combination 3/15, the EXTEND OPCODE SPACE (EOS) allows extension of the basic opcode space (see 5.2).

The basic opcode set, supplied by this encoding technique consists of 496 opcodes, being:

- 16 single-byte opcodes (from column 2);
- $15 \times 32 = 480$ double-byte opcodes (first byte from column 3 except bit combination 3/15, second byte from column 2 or 3).

5.2 Extension mechanism

The basic opcode set can be extended with an unlimited number of extension opcode sets by means of the EXTEND OPCODE SPACE code (EOS, 3/15).

The N-th extension opcode set consists of opcodes of the basic opcode set, prefixed with n times the code EOS. The three possible formats of an opcode from the N-th extension opcode set are

Opcode format	Extension codes	Basic opcode set codes
1	<EOS>...<EOS> ----- n times	<2/x>

Method of encoding opcodes

Extension mechanism

2	$\langle \text{EOS} \rangle \dots \langle \text{EOS} \rangle$ $\begin{array}{c} \text{-----} \\ \\ \text{n times} \end{array}$	$\langle 3/y \rangle \langle 2/z \rangle$
3	$\langle \text{EOS} \rangle \dots \langle \text{EOS} \rangle$ $\begin{array}{c} \text{-----} \\ \\ \text{n times} \end{array}$	$\langle 3/y \rangle \langle 3/z \rangle$

$\langle \text{EOS} \rangle = 3/15$

$x = 0, 1, \dots, 15$

$y = 0, 1, \dots, 14$

$z = 0, 1, \dots, 15$

$n = 0, 1, \dots$

$n = 0$ selects the basic opcode set.

$n = 1$ selects the first extension opcode set.

$n = N$ selects the N-th extension opcode set.

The number of opcodes supplied by this encoding technique (basic opcode set plus extension opcode sets) is $496 \cdot (n+1)$, where n is the number of extension sets. (Each extension set has 496 opcodes — 16 single-byte opcodes plus 480 double-byte codes.)

5.3 Opcode assignments

Table 1 lists the opcode assignments for the CGM elements. All opcodes are from the basic opcode set. They are organized as follows: single-byte opcodes are assigned to Graphical Primitive elements except for some of the Circular and Elliptical output elements, where double-byte opcodes with combination 3/4 for the first byte are assigned. All other metafile elements have double-byte opcodes with the following bit combinations for the first byte:

3/0	for Delimiter Elements
3/1	for Metafile Descriptor Elements
3/2	for Picture Descriptor Elements
3/3	for Control Elements
3/5 and 3/6	for Attribute Elements
3/7	for Escape and External Elements
3/8	for Segment Control Elements and Segment Attribute Elements

Table 1 — Opcodes for metafile elements

Opcode	7-Bit coding		8-Bit coding	
BEGIN METAFILE opcode	3/0	2/0	03/0	02/0
END METAFILE opcode	3/0	2/1	03/0	02/1
BEGIN PICTURE opcode	3/0	2/2	03/0	02/2
BEGIN PICTURE BODY opcode	3/0	2/3	03/0	02/3
END PICTURE opcode	3/0	2/4	03/0	02/4
BEGIN SEGMENT opcode	3/0	2/5	03/0	02/5
END SEGMENT opcode	3/0	2/6	03/0	02/6
BEGIN FIGURE opcode	3/0	2/7	03/0	02/7
END FIGURE opcode	3/0	2/8	03/0	02/8
BEGIN PROTECTION REGION opcode	3/0	2/12	03/0	02/12
END PROTECTION REGION opcode	3/0	2/13	03/0	02/13
BEGIN COMPOUND LINE opcode	3/0	2/14	03/0	02/14
END COMPOUND LINE opcode	3/0	2/15	03/0	02/15
BEGIN COMPOUND TEXT PATH opcode	3/0	3/0	03/0	03/0
END COMPOUND TEXT PATH opcode	3/0	3/1	03/0	03/1
BEGIN TILE ARRAY opcode	3/0	3/2	03/0	03/2
END TILE ARRAY opcode	3/0	3/3	03/0	03/3
METAFILE VERSION opcode	3/1	2/0	03/1	02/0
METAFILE DESCRIPTION opcode	3/1	2/1	03/1	02/1
VDC TYPE opcode	3/1	2/2	03/1	02/2
INTEGER PRECISION opcode	3/1	2/3	03/1	02/3
REAL PRECISION opcode	3/1	2/4	03/1	02/4
INDEX PRECISION opcode	3/1	2/5	03/1	02/5
COLOUR PRECISION opcode	3/1	2/6	03/1	02/6
COLOUR INDEX PRECISION opcode	3/1	2/7	03/1	02/7
MAXIMUM COLOUR INDEX opcode	3/1	2/8	03/1	02/8
COLOUR VALUE EXTENT opcode	3/1	2/9	03/1	02/9
METAFILE ELEMENT LIST opcode	3/1	2/10	03/1	02/10
BEGIN METAFILE DEFAULTS REPLACEMENT opcode	3/1	2/11	03/1	02/11
END METAFILE DEFAULTS REPLACEMENT opcode	3/1	2/12	03/1	02/12
FONT LIST opcode	3/1	2/13	03/1	02/13
CHARACTER SET LIST opcode	3/1	2/14	03/1	02/14
CHARACTER CODING ANNOUNCER opcode	3/1	2/15	03/1	02/15
NAME PRECISION opcode	3/1	3/0	03/1	03/0
MAXIMUM VDC EXTENT opcode	3/1	3/1	03/1	03/1
SEGMENT PRIORITY EXTENT opcode	3/1	3/2	03/1	03/2
COLOUR MODEL opcode	3/1	3/3	03/1	03/3
COLOUR CALIBRATION opcode	3/1	3/4	03/1	03/4
FONT PROPERTIES opcode	3/1	3/5	03/1	03/5
GLYPH MAPPING opcode	3/1	3/6	03/1	03/6
SYMBOL LIBRARY LIST opcode	3/1	3/7	03/1	03/7

Method of encoding opcodes

Opcode assignments

Table 1 — Opcodes for metafile elements (continued)

Opcode	7-Bit coding		8-Bit coding	
SCALING MODE opcode	3/2	2/0	03/2	02/0
COLOUR SELECTION MODE opcode	3/2	2/1	03/2	02/1
LINE WIDTH SPECIFICATION MODE opcode	3/2	2/2	03/2	02/2
MARKER SIZE SPECIFICATION MODE opcode	3/2	2/3	03/2	02/3
EDGE WIDTH SPECIFICATION MODE opcode	3/2	2/4	03/2	02/4
VDC EXTENT opcode	3/2	2/5	03/2	02/5
BACKGROUND COLOUR opcode	3/2	2/6	03/2	02/6
DEVICE VIEWPORT opcode	3/2	2/7	03/2	02/7
DEVICE VIEWPORT SPECIFICATION MODE opcode	3/2	2/8	03/2	02/8
DEVICE VIEWPORT MAPPING opcode	3/2	2/9	03/2	02/9
LINE REPRESENTATION opcode	3/2	2/10	03/2	02/10
MARKER REPRESENTATION opcode	3/2	2/11	03/2	02/11
TEXT REPRESENTATION opcode	3/2	2/12	03/2	02/12
FILL REPRESENTATION opcode	3/2	2/13	03/2	02/13
EDGE REPRESENTATION opcode	3/2	2/14	03/2	02/14
INTERIOR STYLE SPECIFICATION MODE opcode	3/2	2/15	03/2	02/15
LINE AND EDGE TYPE DEFINITION opcode	3/2	3/0	03/2	03/0
HATCH STYLE DEFINITION opcode	3/2	3/1	03/2	03/1
GEOMETRIC PATTERN DEFINITION opcode	3/2	3/2	03/2	03/2
VDC INTEGER PRECISION opcode	3/3	2/0	03/3	02/0
VDC REAL PRECISION opcode	3/3	2/1	03/3	02/1
AUXILIARY COLOUR opcode	3/3	2/2	03/3	02/2
TRANSPARENCY opcode	3/3	2/3	03/3	02/3
CLIP RECTANGLE opcode	3/3	2/4	03/3	02/4
CLIP INDICATOR opcode	3/3	2/5	03/3	02/5
LINE CLIPPING MODE opcode	3/3	2/6	03/3	02/6
MARKER CLIPPING MODE opcode	3/3	2/7	03/3	02/7
EDGE CLIPPING MODE opcode	3/3	2/8	03/3	02/8
NEW REGION opcode	3/3	2/9	03/3	02/9
SAVE PRIMITIVE CONTEXT opcode	3/3	2/10	03/3	02/10
RESTORE PRIMITIVE CONTEXT opcode	3/3	2/11	03/3	02/11
PROTECTION REGION INDICATOR opcode	3/3	3/0	03/3	03/0
GENERALIZED TEXT PATH MODE opcode	3/3	3/1	03/3	03/1
MITRE LIMIT opcode	3/3	3/2	03/3	03/2
TRANSPARENT CELL COLOUR opcode	3/3	3/3	03/3	03/3
POLYLINE opcode	2/0		02/0	
DISJOINT POLYLINE opcode	2/1		02/1	
POLYMARKER opcode	2/2		02/2	
TEXT opcode	2/3		02/3	
RESTRICTED TEXT opcode	2/4		02/4	
APPEND TEXT opcode	2/5		02/5	
POLYGON opcode	2/6		02/6	
POLYGON SET opcode	2/7		02/7	
CELL ARRAY opcode	2/8		02/8	
GENERALIZED DRAWING PRIMITIVE opcode	2/9		02/9	
RECTANGLE opcode	2/10		02/10	

Table 1 — Opcodes for metafile elements (continued)

Opcode	7-Bit coding		8-Bit coding	
CIRCLE opcode	3/4	2/0	03/4	02/0
CIRCULAR ARC 3 POINT opcode	3/4	2/1	03/4	02/1
CIRCULAR ARC 3 POINT CLOSE opcode	3/4	2/2	03/4	02/2
CIRCULAR ARC CENTRE opcode	3/4	2/3	03/4	02/3
CIRCULAR ARC CENTRE CLOSE opcode	3/4	2/4	03/4	02/4
ELLIPSE opcode	3/4	2/5	03/4	02/5
ELLIPTICAL ARC opcode	3/4	2/6	03/4	02/6
ELLIPTICAL ARC CLOSE opcode	3/4	2/7	03/4	02/7
CIRCULAR ARC CENTRE REVERSED opcode	3/4	2/8	03/4	02/8
CONNECTING EDGE opcode	3/4	2/9	03/4	02/9
HYPERBOLIC ARC opcode	3/4	2/10	03/4	02/10
PARABOLIC ARC opcode	3/4	2/11	03/4	02/11
NON-UNIFORM B-SPLINE opcode	3/4	2/12	03/4	02/12
NON-UNIFORM RATIONAL B-SPLINE opcode	3/4	2/13	03/4	02/13
POLYBEZIER opcode	3/4	2/14	03/4	02/14
POLYSYMBOL opcode	3/4	2/15	03/4	02/15
BITONAL TILE opcode	3/4	3/0	03/4	03/0
TILE opcode	3/4	3/1	03/4	03/1
LINE BUNDLE INDEX opcode	3/5	2/0	03/5	02/0
LINE TYPE opcode	3/5	2/1	03/5	02/1
LINE WIDTH	3/5	2/2	03/5	02/2
LINE COLOUR opcode	3/5	2/3	03/5	02/3
MARKER BUNDLE INDEX opcode	3/5	2/4	03/5	02/4
MARKER TYPE opcode	3/5	2/5	03/5	02/5
MARKER SIZE opcode	3/5	2/6	03/5	02/6
MARKER COLOUR opcode	3/5	2/7	03/5	02/7
TEXT BUNDLE INDEX opcode	3/5	3/0	03/5	03/0
TEXT FONT INDEX opcode	3/5	3/1	03/5	03/1
TEXT PRECISION opcode	3/5	3/2	03/5	03/2
CHARACTER EXPANSION FACTOR opcode	3/5	3/3	03/5	03/3
CHARACTER SPACING opcode	3/5	3/4	03/5	03/4
TEXT COLOUR opcode	3/5	3/5	03/5	03/5
CHARACTER HEIGHT opcode	3/5	3/6	03/5	03/6
CHARACTER ORIENTATION opcode	3/5	3/7	03/5	03/7
TEXT PATH opcode	3/5	3/8	03/5	03/8
TEXT ALIGNMENT opcode	3/5	3/9	03/5	03/9
CHARACTER SET INDEX opcode	3/5	3/10	03/5	03/10
ALTERNATE CHARACTER SET INDEX opcode	3/5	3/11	03/5	03/11
FILL BUNDLE INDEX opcode	3/6	2/0	03/6	02/0
INTERIOR STYLE opcode	3/6	2/1	03/6	02/1
FILL COLOUR opcode	3/6	2/2	03/6	02/2
HATCH INDEX opcode	3/6	2/3	03/6	02/3
PATTERN INDEX opcode	3/6	2/4	03/6	02/4
EDGE BUNDLE INDEX opcode	3/6	2/5	03/6	02/5
EDGE TYPE opcode	3/6	2/6	03/6	02/6
EDGE WIDTH opcode	3/6	2/7	03/6	02/7
EDGE COLOUR opcode	3/6	2/8	03/6	02/8
EDGE VISIBILITY opcode	3/6	2/9	03/6	02/9
FILL REFERENCE POINT opcode	3/6	2/10	03/6	02/10

Table 1 — Opcodes for metafile elements (concluded)

Opcode	7-Bit coding		8-Bit coding	
PATTERN TABLE opcode	3/6	2/11	03/6	02/11
PATTERN SIZE opcode	3/6	2/12	03/6	02/12
COLOUR TABLE opcode	3/6	3/0	03/6	03/0
ASPECT SOURCE FLAGS opcode	3/6	3/1	03/6	03/1
PICK IDENTIFIER opcode	3/6	3/2	03/6	03/2
LINE CAP opcode	3/5	2/8	03/5	02/8
LINE JOIN opcode	3/5	2/9	03/5	02/9
LINE TYPE CONTINUATION opcode	3/5	2/10	03/5	02/10
LINE TYPE INITIAL OFFSET opcode	3/5	2/11	03/5	02/11
TEXT SCORE TYPE opcode	3/5	2/12	03/5	02/12
RESTRICTED TEXT TYPE opcode	3/5	2/13	03/5	02/13
INTERPOLATED INTERIOR opcode	3/5	2/14	03/5	02/14
EDGE CAP opcode	3/5	2/15	03/5	02/15
EDGE JOIN opcode	3/6	2/13	03/6	02/13
EDGE TYPE CONTINUATION opcode	3/6	2/14	03/6	02/14
EDGE TYPE INITIAL OFFSET opcode	3/6	2/15	03/6	02/15
SYMBOL LIBRARY INDEX opcode	3/6	3/3	03/6	03/3
SYMBOL COLOUR opcode	3/6	3/4	03/6	03/4
SYMBOL SIZE opcode	3/6	3/5	03/6	03/5
SYMBOL ORIENTATION opcode	3/6	3/6	03/6	03/6
ESCAPE opcode	3/7	2/0	03/7	02/0
DOMAIN RING opcode	3/7	3/0	03/7	03/0
MESSAGE opcode	3/7	2/1	03/7	02/1
APPLICATION DATA opcode	3/7	2/2	03/7	02/2
COPY SEGMENT opcode	3/8	2/0	03/8	02/0
INHERITANCE FILTER opcode	3/8	2/1	03/8	02/1
CLIP INHERITANCE opcode	3/8	2/2	03/8	02/2
SEGMENT TRANSFORMATION opcode	3/8	2/3	03/8	02/3
SEGMENT HIGHLIGHTING opcode	3/8	2/4	03/8	02/4
SEGMENT DISPLAY PRIORITY opcode	3/8	2/5	03/8	02/5
SEGMENT PICK PRIORITY opcode	3/8	2/6	03/8	02/6

NOTE — There is no no-op element in the character encoding. However, the character substitution mechanism allows generators to achieve this effect.

6 Method of encoding parameters

The parameter part of a CGM element may contain one or more parameters, each parameter consisting of one or more bytes.

All parameters are coded in columns 4 through 7. (However, the coded representation of a 'string' parameter may include bit combinations from other columns of the code table — see the description of string parameters in 6.9). The general format of a parameter byte is

```

      b8          b1
    +--+-----+
    |X|1|b b b b b|
    +--+-----+

```

The "X" (bit b8) is the parity bit (or omitted bit) in a 7-bit environment. In an 8-bit environment it is 0. Bit b7 is the parameter flag.

Except for 'string', 'string fixed', and 'data record' parameters all parameters are encoded using one or both of two formats, Basic format or Bitstream format.

6.1 Basic format

Each Basic format parameter is coded as a sequence of one or more bytes, structured as follows:

```

      b8          b1
    +--+-----+
    |X|1|e|s|b b b b|  first byte
    +--+-----+
      .
      .
      .
      b8          b1
    +--+-----+
    |X|1|e|b b b b b|  last byte
    +--+-----+

```

The "X" (bit b8) is the parity bit (or omitted bit) in a 7-bit environment. In an 8-bit environment it is 0. Bit b7 is the parameter flag.

"e" (b6 of each byte) is the extension flag. For single byte parameters, the extension flag is 0. In multi-byte parameters, the extension flag is 1 in all bytes except the last byte, where it is 0.

Bits b5 through b1 are the data bits of the parameter.

"s" is the sign bit; if equal to 0 then the integer is non-negative and if equal to 1 then the integer is negative. The number zero shall always be coded as "plus zero" 4/0. The "minus zero" coding is reserved for special usage (see 6.6.3).

The Basic format is used to encode

- enumerated types (E);
- colour indices (CI);
- colour components (CCO);
- indices other than colour indices (IX);
- integers (I);

Method of encoding parameters

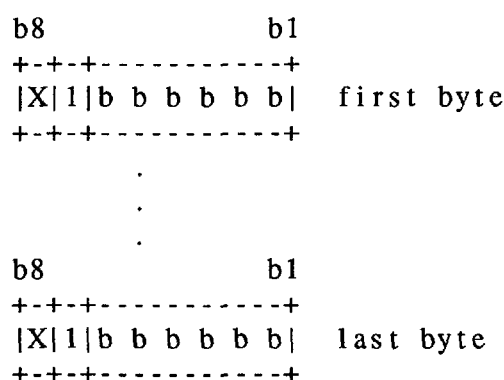
Basic format

- f) real numbers (R);
- g) non-incremental coordinates.
- h) fixed-precision 8-bit unsigned integers — octets (UI8);
- i) fixed-precision 32-bit unsigned integers (UI32).

The most significant part of the parameter is coded in the first byte. The least significant part of the parameter is coded in the last byte.

6.2 Bitstream format

Each Bitstream format parameter is encoded as a sequence of one or more bytes, structured as follows:



The "X" is the parity bit (or omitted bit) in a 7-bit environment. In an 8-bit environment it is 0. Bit b7 is the parameter flag.

Bits b6 through b1 are the data bits of the parameter.

The Bitstream format is used to encode:

- a) incremental mode coordinates (see 6.6.2);
- b) colour direct (see 6.8);
- c) colour index lists (see 6.8).
- d) the compressed bitstream (BS) data type (see 6.15).

Bitstream data are packed in consecutive databits starting from high-numbered bits to lower-numbered bits of the first byte for the most significant part of the bitstream data.

The end of a Bitstream format parameter cannot be derived from the Bitstream format itself (the format is not self-delimiting). Instead,

- for incremental mode coordinates, the end of the data (which identifies the end of the Bitstream format parameter) is identified by the <End of Block> code;
- for colour index lists, the number of bits needed to encode the colour index list (which identifies the end of the Bitstream format parameter) is set by the COLOUR INDEX PRECISION element, or by the *local colour precision* parameter (for those elements which contain such a parameter);
- for colour direct data, the number of bits needed to encode the data (which identifies the end of the Bitstream format operand) is set by the COLOUR PRECISION element, or by the *local colour precision* parameter (for those elements which contain such a parameter).
- for compressed binary colour specifiers of Tile Array elements, the SOS/ST delimiters that are also used for String operands are used to delimit the Bitstream operand.

6.3 Coding integers

Integers are coded as sequences of bytes in the range from 4/0 to 7/15 in the Basic format. If a byte is from columns 4 or 5 of the code table, it is either the last byte in the integer's coded representation or it is a single-byte integer. A multi-byte integer begins with a byte from columns 6 or 7 of the code table.

The structure of integer parameters is as illustrated for Basic format (see 6.1). "bbb..." are bits representing the magnitude of the integer. The most significant part of the parameter is coded in the first byte. The least significant part of the parameter is coded in the last byte.

Any integer can be coded with leading most significant bits which are all zero. For example, 4/3 and 6/0 6/0 4/3 are both valid codings for the integer "+3"; however, efficient metafile generators should avoid such redundant codings.

The size of integer parameters is limited by the current INTEGER PRECISION value.

Integers in the range of -15 to +15 can be coded as single bytes.

(integer: +1) = 4/1	(integer: -1) = 5/1
(integer: +15) = 4/15	(integer: -15) = 5/15

Larger integers require more bytes.

(integer: +16) = 6/0 5/0	(integer: -16) = 7/0 5/0
(integer: +1034) = 6/1 6/0 4/10	(integer: -1034) = 7/1 6/0 4/10

6.4 Coding real numbers

Each real number is coded as an integer mantissa followed by an optional exponent, both coded in the Basic format. The exponent is the power of two by which the integer mantissa is to be multiplied.

The exponent may be implicitly defined as a default exponent, which is then omitted in the Real format, or the exponent may be coded explicitly as the second part of the Real format:

<real format> = <mantissa part> [<exponent part>]

Depending on the "exponent allowed" parameter of the REAL PRECISION element, one of the bits in the first byte of the mantissa tells whether the exponent follows. If the "exponent follows" bit in the mantissa is zero, or if REAL PRECISION has specified that there is to be no "exponent follows" bit, then the exponent is omitted and a default value is assumed which is set by another parameter of the REAL PRECISION element.

Note that the coding of real VDC coordinates is controlled by VDC REAL PRECISION, not by REAL PRECISION, and that the rules for default exponents in real VDC coordinates are slightly different (see 6.5 and 6.6).

The mantissa is an integer which is coded in the Basic format. The first byte takes one of two forms, depending on whether or not REAL PRECISION has specified that an "exponent follows" bit is to be included. The format is as follows (first byte):

Method of encoding parameters

Coding real numbers

```

b8          b1
+--+--+--+--+ (if explicit exponent allowed = 'allowed',
|X|1|e|s|p|b b b| i.e. the exponent-follows bit is present)
+--+--+--+--+

or

b8          b1
+--+--+--+--+ (if explicit exponent allowed = 'forbidden',
|X|1|e|s|b b b b| i.e. there is no exponent-follows bit)
+--+--+--+--+

.
.
b8          b1
+--+--+--+--+
|X|1|e|b b b b b| {last byte}
+--+--+--+--+

```

"e" is the extension flag, see 6.1 for a description.

"s" is the sign bit (bit b5 of the first byte): 0 for a positive mantissa, 1 for a negative mantissa.

Here "p" is the exponent-follows bit. If the current "explicit exponent allowed" value of the REAL PRECISION element is 'allowed', bit b4 of the first byte of a mantissa is used as the "exponent follows" bit: 1 if an explicit exponent follows, 0 if no exponent follows the mantissa (then the default exponent set by REAL PRECISION is assumed). If the current "explicit exponent allowed" value of the REAL PRECISION element is 'forbidden', bit b4 of the first byte is used as a databit and the default exponent set by REAL PRECISION is assumed.

The exponent is coded as an integer in Basic format, see 6.3.

Mantissas or exponents of "minus zero" are not allowed and reserved for future use.

For example, suppose that REAL PRECISION has specified that each real parameter is to be coded with an "exponent follows" bit in its mantissa. In that case, the binary numeral +1.110010110011 is coded as follows:

```

(real: binary +1.11 00101 10011)

= (mantissa: +111 00101 10011, "exponent-follows")
  (exponent: -12)

= |X|1|1|0|1|1 1 1| |X|1|1|0 0 1 0 1| |X|1|0|1 0 0 1 1| {mantissa}
+--+--+--+--+ +--+--+--+--+ +--+--+--+--+
+--+--+--+--+
|X|1|0|1|1 1 0 0| {exponent}
+--+--+--+--+

= 6/15 6/5 5/3 5/12

```

Again, suppose that REAL PRECISION has specified that real parameters are to be coded without exponents and without exponent-follows bits in their mantissas, and that the default exponent is to be -13. In that case, binary +1.110010110011 is coded as follows:

ISO/IEC 8632-2:1992 (E)

Coding real numbers

Method of encoding parameters

$$\begin{aligned}
 & (\text{real: binary } +1.110010110011) \\
 & = (\text{mantissa: } +1110 \ 01011 \ 00110) \\
 & = \begin{array}{ccccccc} + & - & + & - & + & - & + \\ |X| & 1 & 1 & 0 & 1 & 1 & 0 \\ + & - & + & - & + & - & + \end{array} \begin{array}{ccccccc} + & - & + & - & + & - & + \\ |X| & 1 & 1 & 0 & 1 & 0 & 1 \\ + & - & + & - & + & - & + \end{array} \begin{array}{ccccccc} + & - & + & - & + & - & + \\ |X| & 1 & 0 & 0 & 0 & 1 & 1 \\ + & - & + & - & + & - & + \end{array} \{ \text{mantissa} \} \\
 & = 6/14 \quad 6/11 \quad 4/6
 \end{aligned}$$

6.5 Coding VDCs and points

A point is a pair of VDC scalars. A VDC scalar is either an integer or real number according to whether VDC TYPE is integer or real.

When VDC TYPE is integer, the encodings of the VDC and point data types are as described in 6.3, Coding Integers. The size of the VDC and point parameters is limited by the current VDC INTEGER PRECISION value.

When VDC TYPE is real, the encodings of the VDC and point data types are as described in 6.4, Coding Real Numbers. The size of the VDC and point parameters is limited by the current VDC REAL PRECISION value. Whether or not "exponent follows" bits are included in the mantissas of VDCs and points is determined by a parameter of VDC REAL PRECISION. The default value for an omitted exponent in a VDC is determined by the "default exponent" parameter in VDC REAL PRECISION. If the exponent is omitted from a point data type, a default exponent is assumed as follows:

- If the point is not in a point list, or is the first point of a point list, the omitted exponent assumes the default value exponents of VDC parameters, as specified by VDC REAL PRECISION.
- If the point is in a point list, but is not the first point of that point list, an omitted exponent may assume a different value as follows: if the exponent is omitted from the x-component of a real point, it defaults to the value of the exponent in the preceding x-component; similarly, an exponent omitted from a y-component assumes the value of the exponent in the preceding y-coordinate.

6.6 Coding point list parameters

Point lists may be coded with one or both of the following coding structures. These structures are called 'Displacement Mode' and 'Incremental Mode'. Both formats may be used within a single point list.

Individual points and the first point of each point list are always be coded in Displacement Mode. The displacement values delta-x and delta-y are displacements measured from the origin, i.e. absolute positions. For points after the first point in a point list, each displacement is measured from the preceding point of the point list (this is true regardless of whether the preceding point was coded in Displacement Mode or in Incremental Mode).

6.6.1 Displacement mode

In Displacement Mode, each point is coded as a sequence of two VDCs. The first VDC value gives the x-component of the point's displacement from the preceding point, while the second VDC value specifies the y-component of that displacement.

$$\langle P \rangle = \langle \text{VDC: delta } x \rangle \langle \text{VDC: delta } y \rangle$$

Method of encoding parameters

Coding point list parameters

6.6.2 Incremental mode

The Incremental Mode is defined as a Differential Chain Code (DCC).

NOTE — Because of the difficulty of applying the DCC method effectively, the use of DCC coding is discouraged. The DCC method may be removed in a future revision of this part of ISO/IEC 8632.

The data in this mode does not reflect actual coordinates, but defines steps (increments) from one coordinate position to another. These increments are identified by points on a Ring. A Ring is a set of points on a square whose centre is the previously identified point. The first centre point is encoded in Displacement Mode.

A Ring is characterized by its Radius (R) in Basic Grid Units (BGU), its Angular resolution (by a factor p) and its Direction (D). The maximum number of points on a Ring is 8R. The actual number of points on a Ring with a given Angular resolution factor p follows from

$$N = \frac{8R}{2^p}, \quad p=0,1,2,3$$

N is required to be even.

The points on the Ring are numbered, starting at the Direction point D, counter clockwise from 0 to M-1 and clockwise from -1 to -M, with $M=N/2$.

Following are examples of two Rings, both having Radius R=3. The Ring on the left has Angular resolution factor p=0 and the one on the right has Angular resolution factor p=1.

9	.	8	7	6	5	4	.	3	.	4	3	2	.
10	.						.	2	5	.			1
11	.						.	1					
-12	.			.			.	0	-6	.	.		0
-11	.			centre			.	-1			centre		
-10	.						.	-2	-5	.			-1
-9	.	.	.	.	.	.	.	-3	.	.	.	.	
		-8	-7	-6	-5	-4				-4	-3	-2	

The Direction of the Ring is given by the position of the point with number ZERO. The initial position of this point is on the positive x-axis of VDC space, while the Cartesian axes are drawn through the centre point of the Ring. The Direction of the Rings following the initial one is dependent on the direction of the increments. This Direction is determined in the following way:

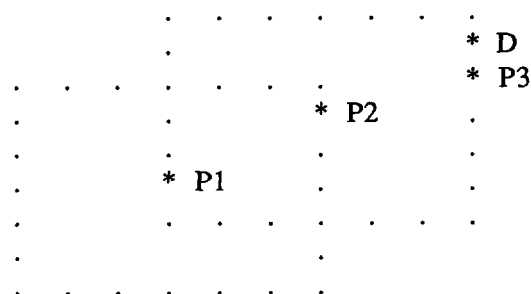
Let P2 be the current centre point and P1 the previous one, i.e., P2 is a point on the Ring with centre in P1. The Direction of the Ring is obtained by drawing a line from P1 through P2; where this direction ray intersects P2's Ring is the ZERO point of that Ring. So the Direction of the Ring is dependent on the direction of the line to be displayed.

ISO/IEC 8632-2:1992 (E)

Coding point list parameters

Method of encoding parameters

In the DCC only the differences between points on the consecutive Rings are coded. The position of point P3, i.e., the increment on the new Ring (centre P2) is described by the difference between the position of point P2 on the previous Ring and the position of the new point P3 on the current Ring, the positions numbered with respect to the previous Ring. As shown in example below, the position of point P3 is defined by the difference: $P3 - P2 = -1$. P3 and P2 being point numbers on the two Rings, numbered as given in the previous figure. The Direction (position of the point with number ZERO) is identified by D.



Change of direction with R=3

The basic Radius of the Ring, as used in Incremental mode, is dependent on the BGU. The BGU is the smallest nonzero value that can be expressed within the precision set by the VDC INTEGER or REAL PRECISION element. For VDC TYPE integer, basic Radius (=BGU) = 1. For VDC TYPE real the size of the BGU is set by a parameter of VDC REAL PRECISION, called smallest-real code: letting "src" stand for smallest-real code, $BGU=2^{src}$. The basic Radius is a multiple of the BGU, i.e. it is a positive integer. For VDC TYPE real the default value for the basic Radius follows from

smallest-real code	basic Radius
> g	1
= g	1
< g	$\frac{2^g}{BGU}$

where g is a nominal smallest-real code. A value of g = -8 is used in this part of ISO/IEC 8632.

The basic Radius and the Angular resolution factor may be changed with the DOMAIN RING element. If the basic Radius is set to a value less than ONE, the value ONE is assumed for the basic Radius.

The encoding used in Incremental Mode makes use of the DCC property by using variable length code-words (Huffman Code). The encoding also allows changing of the Radius and the Angular resolution factor. The Radius can have a value of R, 2R, 4R or 8R, where R is the defined Radius. The Angular resolution factor p can be 0, 1, 2, or 3, the default value is 0.

The Huffman Code table (see table 2) used in the Incremental mode is a fixed length table. To allow the encoding of more points on a Ring two Escape codes are defined. With these Escape codes the points outside the Huffman Code table can be addressed. The end of the Incremental mode data is indicated by an End of Block value in the Huffman Code table.

Method of encoding parameters

Coding point list parameters

Table 2 — Huffman code table for incremental mode

Length	Code-word	Point number
2	00	0
2	10	1
2	01	-1
4	1100	2
4	1101	-2
6	111000	3
6	111001	-3
6	111010	4
6	111011	-4
8	11110000	5
8	11110001	-5
8	11110010	6
8	11110011	-6
8	11110100	7
8	11110101	-7
8	11110110	8
8	11110111	-8
10	1111100000	9
10	1111100001	-9
10	1111100010	10
10	1111100011	-10
10	1111100100	11
10	1111100101	-11
10	1111100110	12
10	1111100111	-12
10	1111101000	13
10	1111101001	-13
10	1111101010	14
10	1111101011	-14
10	1111101100	15
10	1111101101	-15
10	1111101110	16
10	1111101111	-16
10	1111110000	17
10	1111110001	-17
10	1111110010	18
10	1111110011	-18
10	1111110100	19
10	1111110101	-19
10	1111110110	C1
10	1111110111	-20
10	1111111000	C2
10	1111111001	C3
10	1111111010	C4
10	1111111011	C5
10	1111111100	C6
10	1111111101	IM-ESC 1
10	1111111110	IM-ESC 2
10	1111111111	End of Block

The <End of Block> code from the Huffman Code table identifies the end of the Incremental mode data. Remaining bits in the last Incremental mode data byte have no meaning; they are required to be zero.