

INTERNATIONAL STANDARD

ISO/IEC
8632-3

Second edition
1992-10-01

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

Part 3: Binary encoding

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

Partie 3: Codage binaire



Reference number
ISO/IEC 8632-3:1992(E)



Documents
Document Sharing Hub

CONTENTS

1	Scope	1
2	Normative references	2
3	Notational conventions	3
4	Overall structure	4
4.1	General form of metafile	4
4.2	General form of pictures	4
4.3	General structure of the binary metafile	4
4.4	Structure of the command header	5
5	Primitive data forms	8
5.1	Signed integer	8
5.1.1	Signed integer at 8-bit precision	8
5.1.2	Signed integer at 16-bit precision	8
5.1.3	Signed integer at 24-bit precision	9
5.1.4	Signed integer at 32-bit precision	9
5.2	Unsigned integer	9
5.2.1	Unsigned integers at 8-bit precision	9
5.2.2	Unsigned integers at 16-bit precision	9
5.2.3	Unsigned integers at 24-bit precision	9
5.2.4	Unsigned integers at 32-bit precision	10
5.3	Character	10
5.4	Fixed point real	10
5.4.1	Fixed point real at 32-bit precision	10
5.4.2	Fixed point real at 64-bit precision	10
5.4.3	Value of fixed point reals	11
5.5	Floating point	11
5.5.1	Floating point real at 32-bit precision	12
5.5.2	Floating point real at 64-bit precision	12
6	Representation of abstract parameter types	13

7	Representation of each element	18
7.1	Method of presentation	18
7.2	Delimiter elements	19
7.3	Metafile descriptor elements	21
7.4	Picture descriptor elements	28
7.5	Control elements	33
7.6	Graphical primitive elements	36
7.7	Attribute elements	43
7.8	Escape element	52
7.9	External elements	53
7.10	Segment control and segment attribute elements	54
8	Defaults	58
9	Conformance	59
A	Formal grammar	60
B	Examples	63
B.1	Example 1 : BEGIN METAFILE 'Example 1'	63
B.2	Example 2 : BEGIN PICTURE 'Test'	63
B.3	Example 3 : POLYLINE from 0,2 to 1,3 to 2,1 to 0,2	64
B.4	Example 4 : TEXT 'Hydrogen' at 0,1	64
B.5	Example 5 : Partitioned POLYLINE with 50 points	65
B.6	Example 6 : METAFILE DEFAULT REPLACEMENT linewidth 0.5	66
B.7	Example 7 : Application Data # 655 with 10K octets (chars) of data	66
C	List of binary encoding metafile element codes	67

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 8632-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

This second edition cancels and replaces the first edition (ISO 8632-3:1987), 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. Annexes B and C are for information only.

Introduction

0.1 Purpose of the Binary Encoding

The Binary Encoding of the Computer Graphics Metafile (CGM) provides a representation of the Metafile syntax that can be optimized for speed of generation and interpretation, while still providing a standard means of interchange among computer systems. The encoding uses binary data formats that are much more similar to the data representations used within computer systems than the data formats of the other encodings.

Some of the data formats may exactly match those of some computer systems. In such cases processing is reduced very much relative to the other standardized encodings. On most computer systems processing requirements for the Binary Encoding will be substantially lower than for the other encodings.

In cases where a computer system's architecture does not match the standard formats used in the Binary Encoding, and where absolute minimization of processing requirements is critical, and where interchange among dissimilar systems does not matter, it may be more appropriate to use a private encoding, conforming to the rules specified in clause 7 of ISO/IEC 8632-1.

0.2 Objectives

This encoding has the following features.

- a) Partitioning of parameter lists: metafile elements are coded in the Binary Encoding by one or more partitions (see clause 4); the first (or only) partition of an element contains the opcode (Element Class plus Element Id).
- b) Alignment of elements: every element begins on a word boundary. When the data of an element (whether partitioned or not) does not terminate on an even-octet boundary, then the following element is aligned by padding after the data of the preceding element with zero bits to the next even-octet boundary. A no-op element is available in this encoding. It is skipped and ignored by interpreters. It may be used to align data on machine-dependent record boundaries for speed of processing.
- c) Uniformity of format: all elements have an associated parameter length value. The length is specified as an octet count. As a result, it is possible to scan the metafile, without interpreting it, at high speed.

Introduction**Objectives**

- d) Alignment of coordinate data: at default precisions and by virtue of alignment of elements, coordinate data always start on word boundaries. This minimizes processing by ensuring, on a wide class of computing systems, that single coordinates do not have to be assembled from pieces of multiple computer words.
- e) Efficiency of encoding integer data: other data such as indexes, colour and characters are encoded as one or more octets. The precision of every parameter is determined by the appropriate precision as given in the Metafile Descriptor.
- f) Order of bit data: in each word, or unit within a word, the bit with the highest number is the most significant bit. Likewise, when data words are accessed sequentially, the least significant word follows the most significant.
- g) Extensibility: the arrangement of Element Class and Element Id values has been designed to allow future growth, such as new graphical elements.
- h) Format of real data: real numbers are encoded using either IEEE floating point representation or a metafile fixed-point representation.
- i) Run length encoding: if many adjacent cells have the same colour (or colour index) efficient encoding is possible. For each run a cell count is specified followed by the colour (or colour index).
- j) Packed list encoding: if adjacent colour cells do not have the same colour (or colour index) the metafile provides bit-stream lists in which the values are packed as closely as possible.

0.3 Relationship to other International Standards

The floating point representation of real data in this part of ISO/IEC 8632 is that in ANSI/IEEE 754-1986.

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

Relationship to other International Standards

Introduction

This page intentionally left blank

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

Part 3 : Binary encoding

1 Scope

This part of ISO/IEC 8632 specifies a binary encoding of the Computer Graphics Metafile. For each of the elements specified in ISO/IEC 8632-1, this part specifies an encoding in terms of data types. For each of these data types, an explicit representation in terms of bits, octets and words is specified. For some data types, the exact representation is a function of the precisions being used in the metafile, as recorded in the METAFILE DESCRIPTOR.

This encoding of the Computer Graphics Metafile will, in many circumstances, minimize the effort required to generate and interpret the metafile.

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*.

ANSI/IEEE 754, *Standard for Binary Floating Point Arithmetic*.

3 Notational conventions

"Command Header" is used throughout this part of ISO/IEC 8632 to refer to that portion of a Binary-Encoded element that contains the opcode (element class plus element id) and parameter length information (see clause 4).

Within this part, the terms "octet" and "word" have specific meanings. These meanings may not match those of a particular computer system on which this encoding of the metafile is used.

An octet is an 8-bit entity. All bits are significant. The bits are numbered from 7 (most significant) to 0 (least significant).

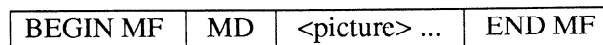
A word is a 16-bit entity. All bits are significant. The bits are numbered from 15 (most significant) to 0 (least significant).

4 Overall structure

4.1 General form of metafile

All elements in the metafile are encoded using a uniform scheme. The elements are represented as variable length data structures, each consisting of opcode information (element class plus element id) designating the particular element, the length of its parameter data and finally the parameter data (if any).

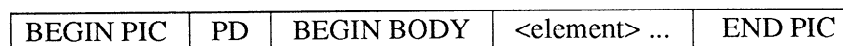
The structure of the metafile is as follows. (For the purposes of this diagram only, MF is used as an abbreviation for METAFILE.)



The BEGIN METAFILE element is followed by the METAFILE DESCRIPTOR (MD). After this the pictures follow, each logically independent of each other. Finally the Metafile is ended with an END METAFILE element.

4.2 General form of pictures

Apart from the BEGIN METAFILE, END METAFILE and Metafile Descriptor elements, the metafile is partitioned into pictures. All pictures are mutually independent. A picture consists of a BEGIN PICTURE element, a PICTURE DESCRIPTOR (PD) element, a BEGIN PICTURE BODY element, an arbitrary number of control, graphical and attribute elements and finally an END PICTURE element. (For the purpose of this diagram only, PIC is used as an abbreviation for PICTURE and BEGIN BODY for BEGIN PICTURE BODY.)



4.3 General structure of the binary metafile

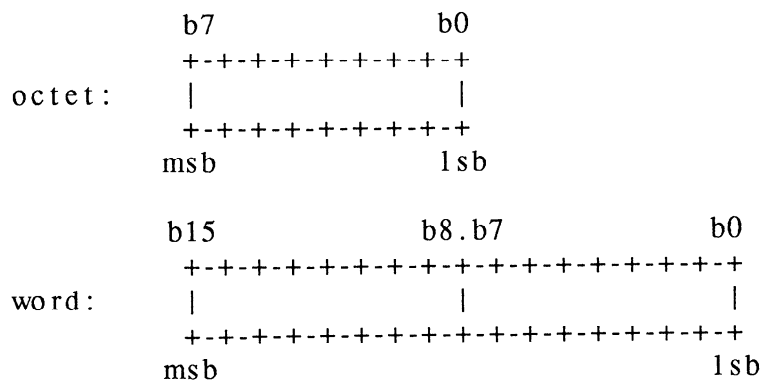
The binary encoding of the metafile is a logical data structure consisting of a sequential collection of bits. For convenience in describing the length and alignment of metafile elements, fields of two different sizes are defined within the structure. These fields are used in the remainder of this part of ISO/IEC 8632 for illustrating the contents and structure of elements and parameters.

For measuring the lengths of elements the metafile is partitioned into octets, which are 8-bit fields.

The structure is also partitioned into 16-bit fields called words (these are logical metafile words). To optimize processing of the binary metafile on a wide collection of computers, metafile elements are constrained to start on word boundaries within the binary data structure (this alignment may necessitate padding an element with bits to a word boundary if the parameter data of the element does not fill to such a boundary).

The octet is the fundamental unit of organization of the binary metafile.

The bits of an octet are numbered 7 to 0, with 7 being the most significant bit. The bits of a word are numbered 15 to 0, with 15 being the most significant bit.

Overall structure**General structure of the binary metafile**

If the consecutive bits of the binary data structure are numbered 1..N, and the consecutive octets are numbered 1..N/8, and the consecutive words are numbered 1..N/16, then the logical correspondence of bits, octets, and words in the binary data structure is as illustrated in the following table:

metafile bit number	octet bit number	word bit number
1	b7/octet1	b15/word1
.	.	.
8	b0/octet1	b8/word1
9	b7/octet2	b7/word1
.	.	.
16	b0/octet2	b0/word1
17	b7/octet3	b15/word2
.	.	.
24	b0/octet3	b8/word2
25	b7/octet4	b7/word2
.	.	.
.	.	.

4.4 Structure of the command header

Throughout this sub-clause, the term "command" is used to denote a binary-encoded element. Metafile elements are represented in the Binary Encoding in one of two forms — short-form commands and long-form commands. There are two differences between them:

- a short-form command always contains a complete element; the long-form command can accommodate partial elements (the data lists of elements can be partitioned);
- a short-form command only accommodates parameter lists up to 30 octets in length; the long-form command accommodates lengths up to 32767 octets per data partition.

The forms differ in the format of the Command Header that precedes the parameter list. The command form for an element (short or long) is established by the first word of the element. For the short-form, the Command Header consists of a single word divided into three fields: element class, element id and parameter list length.

Structure of the command header

Overall structure

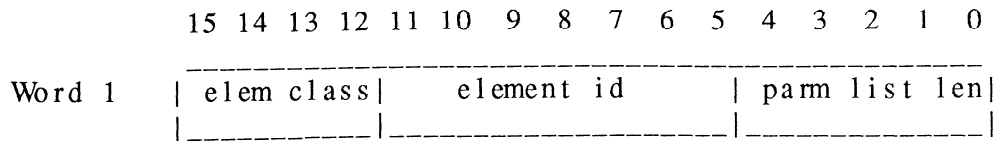


Figure 1 — Format of a short-form Command Header

The fields in the short-form Command Header are as follows:

- bits 15 to 12 element class (value range 0 to 15)
- bits 11 to 5 element id (value range 0 to 127)
- bits 4 to 0 parameter list length: the number of octets of parameter data that follow for this command (value range 0 to 30)

This Command Header is then followed by the parameter list.

The first word of a long-form command is identical in structure to the first word of a short-form command. The presence of the value 11111 binary (decimal 31) in parameter list length field indicates that the command is a long-form command. The Command Header for the long-form command consists of two words. The second word contains the actual parameter list length. The two header words are then followed by the parameter list.

In addition to allowing longer parameter lists, the long-form command allows the parameter list to be partitioned. Bit 15 of the second word indicates whether the given data complete the element or more data follow. For subsequent data partitions of the element, the first word of the long-form Command Header (containing element class and element id) is omitted; only the second word, containing the parameter list length, is given. The parameter list length for each partition specifies the length of that partition, not the length of the complete element. The final partition of an element is indicated by bit 15 of the parameter list length word being zero.

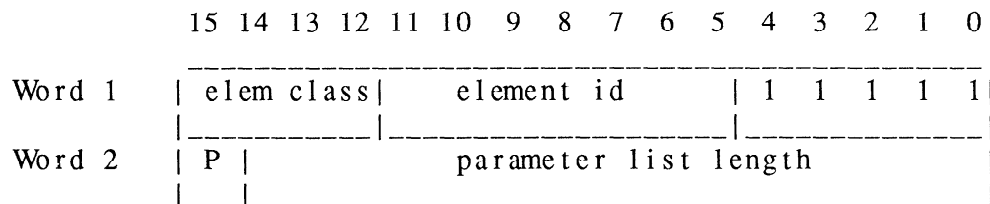


Figure 2 — Format of a long-form Command Header

The fields in the long-form Command Header are as follows:

Word 1

- bits 15 to 12 element class (value range 0 to 15)
- bits 11 to 5 element id (value range 0 to 127)
- bits 4 to 0 binary value 11111 (decimal 31) indicating long-form

Word 2

- bit 15 partition flag
 - 0 for 'last' partition
 - 1 for 'not-last' partition
- bits 14 to 0 parameter list length: the number of octets of parameter data that follow for this command or partition (value range 0 to 32767).

Overall structure**Structure of the command header**

The parameter values follow the parameter list length for either the long-form or short-form commands. The number of values is determined from the parameter list length and the type and precision of the operands. These parameter values have the format illustrated in clause 5 of this part of ISO/IEC 8632. The parameter type for coordinates is indicated in the Metafile Descriptor. For non-coordinate parameters, the parameter type is as specified in clause 5 of ISO/IEC 8632-1. If the parameter type is encoding dependent, its code is specified in the coding tables of clause 7 of this part. Unless otherwise stated, the order of parameters is as listed in clause 5 of ISO/IEC 8632-1.

Every command is constrained to begin on a word boundary. This necessitates padding the command with a single null octet at the end of the command if the command contains an odd number of octets of parameter data. In addition, in elements with parameters whose precisions are shorter than one octet (i.e., those containing a 'local colour precision' parameter) it is necessary to pad the last data-containing octet with null bits if the data do not fill the octet. In all cases, the parameter list length is the count of octets actually containing parameter data — it does not include the padding octet if one is present. It is only at the end of a command that padding is performed, with the single exception of the CELL ARRAY element.

The purpose of this command alignment constraint is to optimize processing on a wide class of computers. At the default metafile precisions, the parameters which are expected to occur in greatest numbers (coordinates, etc) will align on 16-bit boundaries, and Command Headers will align on 16-bit boundaries. Thus, at the default precisions the most frequently parsed entities will lie entirely within machine words in a large number of computer designs. The avoidance of assembling single metafile parameters from pieces of several computer words will approximately halve the amount of processing required to recover element parameters and command header fields from a binary metafile data stream.

This optimization may be compromised or destroyed altogether if the metafile precisions are changed from default. Commands are still constrained to begin on 16-bit boundaries, but the most frequently expected parameters may no longer align on such boundaries as they do at the default precisions.

The short form command header with element class 15, element id 127, and parameter list length 0 is reserved for extension of the number of available element classes in future revisions of this part of ISO/IEC 8632. It should be treated by interpreters as any other element, as far as parsing is concerned. The next "normal" element encountered will have an actual class value different from that encountered in the "element class" field of the command header — it will be adjusted by a bias as will be defined in a future revision of this part of ISO/IEC 8632.

5 Primitive data forms

The Binary Encoding of the CGM uses five primitive data forms to represent the various abstract data types used to describe parameters in ISO/IEC 8632-1.

The primitive data forms and the symbols used to represent them are as follows.

SI	Signed Integer
UI	Unsigned Integer
C	Character
FX	Fixed Point Real
FP	Floating Point Real

Each of these primitive forms (except Character) can be used in a number of precisions. The definitions of the primitive data forms in 5.1 to 5.5 show the allowed precisions for each primitive data form. The definitions are in terms of 'metafile words' which are 16-bit units.

The following terms are used in the following diagrams when displaying the form of numeric values.

msb	most significant bit
lsb	least significant bit
S	sign bit

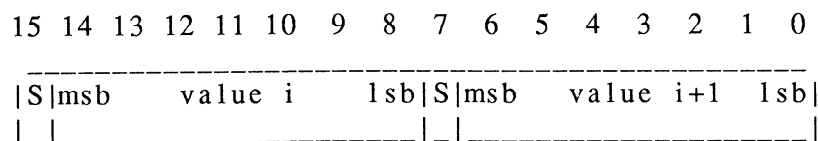
The data types in the following data diagrams are illustrated for the case that the parameter begins on a metafile word boundary. In general, parameters may align on odd or even octet boundaries, because they may be preceded by an odd or even number of octets of other parameter data. Elements containing the local colour precision parameter may have parameters shorter than one octet. It is possible in such cases that the parameters will not align on octet boundaries.

5.1 Signed integer

Signed integers are represented in "two's complement" format. Four precisions may be specified for signed integers: 8-bit, 16-bit, 24-bit and 32-bit. (Integer coordinate data encoded with this primitive data form do not use the 8-bit precision.) In the diagrams of the following subsections, 'value' indicates the value for positive integers and the two's complement of the value for negative integers.

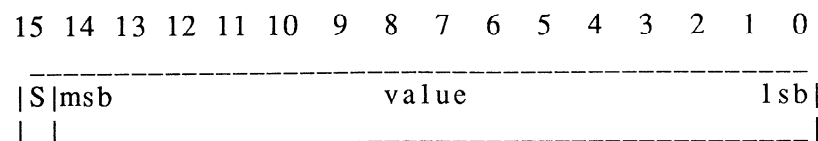
5.1.1 Signed integer at 8-bit precision

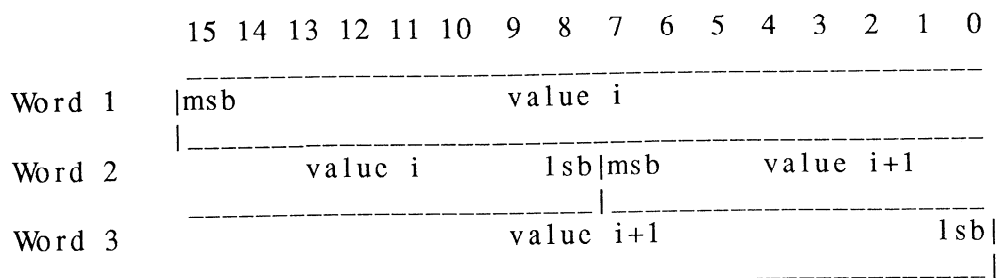
Each value occupies half a metafile word (one octet).



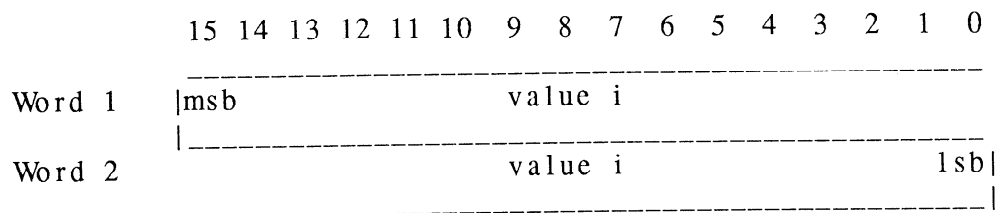
5.1.2 Signed integer at 16-bit precision

Each value occupies one metafile word.

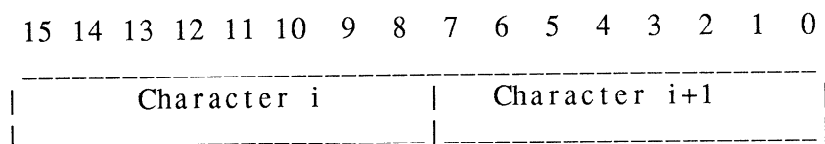


Unsigned integer**Primitive data forms****5.2.4 Unsigned integers at 32-bit precision**

Each value fills two complete metafile words.

**5.3 Character**

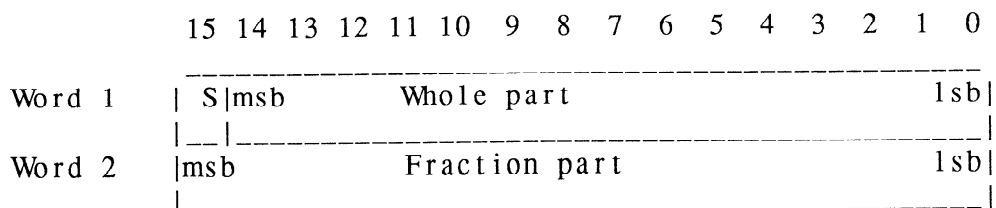
Each character is stored in 1 or more consecutive octets, depending upon the coding of the particular character set. The following illustrates characters which are coded with 1 octet each.

**5.4 Fixed point real**

Fixed point real values are stored as two integers; the first represents the "whole part" and has the same form as a Signed Integer (SI; see 5.1); the second represents the "fractional part" and has the same form as an Unsigned Integer (UI; see 5.2). Two precisions may be specified for Fixed Point Reals: 32-bit or 64-bit.

5.4.1 Fixed point real at 32-bit precision

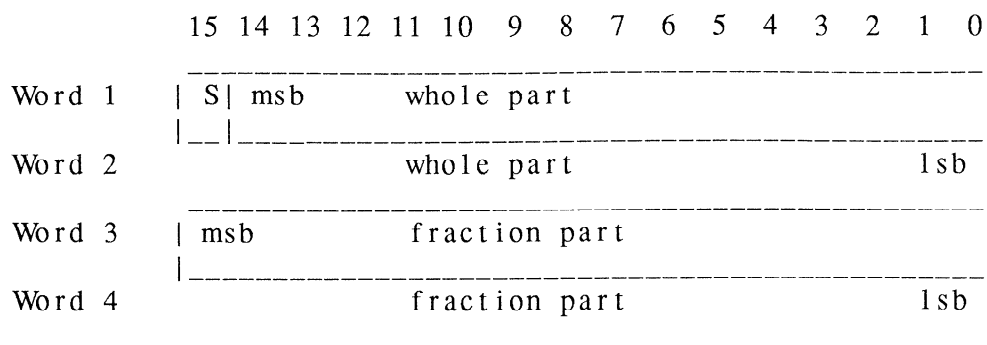
Each Fixed Point Real occupies 2 complete metafile words; the first has the form of a 16-bit Signed Integer and the second the form of a 16-bit Unsigned Integer.

**5.4.2 Fixed point real at 64-bit precision**

Each Fixed Point Real occupies 4 complete metafile words; the first has the form of a 32-bit Signed Integer and the second the form of a 32-bit Unsigned Integer.

Primitive data forms

Fixed point real



5.4.3 Value of fixed point reals

The values of the represented real numbers are given by:

for 32 bits: $\text{real\_value} = SI + \left\lfloor \frac{UI}{2^{16}} \right\rfloor$

for 64 bits: $\text{real\_value} = SI + \left\lfloor \frac{UI}{2^{32}} \right\rfloor$

SI stands for the "whole part" and UI stands for the "fractional part" in these equations. SI, the whole part, is the largest integer less than or equal to the real number being represented.

5.5 Floating point

Floating Point Real values are represented in the floating point format of ANSI/IEEE 754. This format contains three parts:

- a sign bit ('s');
- a biased exponent part ('e');
- a fraction part ('f').

The value is a function of these three values ('s', 'e' and 'f'). If 's' is '0', the value is positive; if 's' is '1', the value is negative. Two precisions may be specified for Floating Point Reals: 32-bit or 64-bit. The magnitude of the value is calculated as follows for 32-bit representation.

- a) If $e = 255$ and $f \neq 0$, then the value is undefined.
- b) If $e = 255$ and $f = 0$, then the value is as large a positive ($s=0$) or negative ($s=1$) value as possible.
- c) If $0 < e < 255$, then the magnitude of the value is $(1.f)(2^{e-127})$.
- d) If $e = 0$ and $f \neq 0$, then the magnitude of the value is $(0.f)(2^{-126})$.
- e) If $e = 0$ and $f = 0$, then the value is 0.

The magnitude of the value is calculated as follows for 64-bit representation.

- a) If $e = 2047$ and $f \neq 0$, then the value is undefined.
- b) If $e = 2047$ and $f = 0$, then the value is as large a positive ($s=0$) or negative ($s=1$) value as possible.
- c) If $0 < e < 2047$, then the magnitude of the value is $(1.f)(2^{e-1023})$.

Floating point

Primitive data forms

- d) If $e = 0$ and $f \neq 0$, then the magnitude of the value is $(0.f)(2^{-1022})$.
- e) If $e = 0$ and $f = 0$, then the value is 0.

5.5.1 Floating point real at 32-bit precision

Each Floating Point Real value occupies 2 metafile words. The size of each field in the value is as follows:

sign	1 bit
exponent	8 bits
fraction	23 bits

1514131211109876543210

Word 1| S|msbExponentlsb|msbFraction

Word 2|Fractionlsb|

5.5.2 Floating point real at 64-bit precision

Each Floating Point Real value occupies 4 metafile words. The size of each field in the value is as follows:

sign	1 bit
exponent	11 bits
fraction	52 bits

1514131211109876543210

Word 1| S|msbExponentlsb|msb

Word 2|Fraction

Word 3|Fraction

Word 4|Fractionlsb|

6 Representation of abstract parameter types

Table 1 shows, for each of the abstract parameter types, how it is represented in the Binary Encoding of the CGM in terms of primitive data forms. The columns of the table are as follows:

- 1) The symbol for the abstract parameter type, as it is specified in clause 5 of ISO/IEC 8632-1.
- 2) The way the parameter type is constructed in terms of the primitive data forms, at the appropriate precisions. The precisions are those defined in clause 5 of ISO/IEC 8632-1.
- 3) The symbol for the number of octets required to represent one instance (occurrence) of the given parameter, at the given precision, and the formula for computing the number.
- 4) The symbol for the range of values which the parameter can assume, followed by the numerical values which the parameter can assume, followed by the numerical values which define the range.

The symbols of columns 3 and 4 are used extensively in the code tables in clause 7. Also used in the code tables are variations on those symbols:

- | | |
|----------------|--|
| +IR, +RR, .. | denote the range of positive integers, range of positive reals, .. |
| -IR, -RR, .. | denote the range of negative integers, range of negative reals, .. |
| ++IR, ++RR, .. | denote the range of non-negative integers, range of non-negative reals, .. |
| mI, mR,.. | denotes 'm' integers, reals, .. |
| I*, R*,.. | denotes an unbounded number of integers, reals, .. |

Combinations are used:

- | | |
|----------------|---|
| 2R, 2I, IX*,.. | indicates a parameter that is represented by 2 reals, then a parameter that is represented by 2 integers and finally a parameter that contains an unlimited number of index values. |
|----------------|---|

Table 1 — Representation of abstract data types

Abstract symbol	Parameter construction from primitive forms	Octets per parameter: symbol and value	Parameter range: symbol and value
CI	UI at colour index precision (cip)	BCI {=cip/8}	CIR {0..($2^{cip}-1$)}
CCO	UI at direct colour precision (dcp)	BCCO {=dcp/8}	CCOR {0..($2^{dcp}-1$)} {see note 2}
CD	(CCO,CCO,CCO) or (CCO,CCO,CCO,CCO)	BCD ={3*BCCO} BCD ={4*BCCO}	CCOR {see notes 1,16}
IX	SI at index precision (ixp)	BIX {=ixp/8}	IXR { -2^{ixp-1} to $2^{ixp-1}-1$ }
E	SI at fixed precision (16-bit) {see note 3}	BE {=2}	{ -2^{15} to $2^{15}-1$ } {see note 18}
I	SI at integer precision (ip)	BI {=ip/8}	IR { -2^{ip-1} to $2^{ip-1}-1$ }
R	FP or FX at real precision (rp)	BR {=sum(rp)/8} {see note 4}	RR {=FPR or FXR, see notes 5,10}
S,SF,D	UI,nC	BS {see note 6}	SR {see notes 6,12}
VDC	SI at VDC integer precision (vip) or FP or FX at VDC real precision (vrp)	BVDC {=vip/8} or BVDC {=sum(vrp)/8} {see note 4}	VDCR { -2^{vip-1} to $2^{vip-1}-1$ } or VDCR {see notes 1,5,7,8}
P	(VDC,VDC)	BP {=2*BVDC}	VDCR {see notes 1,5,7,8}
CO	CI or CD	BCO {=BCI} or BCO {=BCD}	COR {=CIR} {see notes 9,11} or COR {=CCOR}
N	SI at integer precision (np)	BN {=np/8}	NR { -2^{np-1} to $2^{np-1}-1$ }
VC	I or R	BVC {=BI} or BVC {=BR}	VCR {=IR} {see note 13} or VCR {=RR}
VP	(VC,VC)	BVP {=2*BVC}	VCR {see notes 1,13,14}

Representation of abstract parameter types

Table 1 — Representation of abstract data types (concluded)

Abstract symbol	Parameter construction from primitive forms	Octets per parameter: symbol and value	Parameter range: symbol and value
BS	nUI at fixed precision (16-bit) {see note 15}	BBS {=2n}	BSR {see note 15}
UI8	UI at fixed precision (8-bit)	BUI8 {=1}	UI8R {0..255}
UI32	UI at fixed precision (32-bit)	BUI32 {=4}	UI32R {0..2 ³² -1}
SDR	{see note 17}	BSDR	n/a
SS	VDC or R	BSS {=BVDC} or BSS {=BR}	SSR {=VDCR} {see note 19} or SSR {=RR}

NOTES

- 1) For parameters that are composed of multiple identical components (e.g., DIRECT COLOUR, CD, and POINT, P) the range value represents the range of a single component.
- 2) For colour models RGB and CMYK a direct colour component is abstractly a real in the range [0,1]. For colour models CIELAB, CIELUV, and RGB-related it is abstractly a real in the respective colour spaces with possibly different ranges for the direct colour components. The COLOUR VALUE EXTENT element provides for the mapping between a direct colour component represented as UI and the corresponding real value.
- 3) Abstract parameter type Enumeration, E, is encoded identically to abstract type Index, IX, at 16-bit precision.
- 4) The REAL PRECISION element contains an indicator (fixed or floating point) and two precision components. The symbol "sum(rp)" in the table indicates the sum of the number of bits specified in the two components. The same considerations apply to the VDC REAL PRECISION element and the symbol "sum(vrp)" in the tables. The VDC REAL PRECISION control element may cause 'vrp' to be updated in the body of metafile.
- 5) FPR and VDCR (when VDC are floating point reals) are computed following the ANSI/IEEE 754 floating point standard (see clause 5 on the Floating Point Data Form).
- 6) The range for parameter types S and SF is not applicable. The range for character data is not applicable. A string is encoded as a count (unsigned integer) followed by characters. The count is a count of octets in the string, not whole character codes (the two are equal for single byte codes, but not for multi-byte codes).

The encoding of the count is similar to the encoding of length information for metafile commands themselves. If the first octet is in the range 0..254, then it represents the character count for the complete string. If the first octet is 255, then the next 16 bits contain the character count and a continuation flag. The first bit is used as a continuation flag (allowing strings longer than 32767 characters) and the next 15 bits represent the count, 0..32767, for the partial string. If the first bit is 0, then this partial string completes the string parameter. If 1, then this partial string will be followed by another.

If the number of whole character codes in a string is n , and the number of octets per character code is constant within the string and equal to m , and if the string is not continued (as a long-form string may be), then the number of octets in the string parameter is either $n \cdot m + 1$ or $n \cdot m + 3$, depending upon whether the string is short-form or long-form, respectively. If the number of octets per character code is not constant and/or the string is a continued long-form string, then the number of octets in the string is not so easily expressed, but is the total of the octets used in the "data" part of the string and the number of octets used for length information.

Representation of abstract parameter types

- 7) The abstract parameter type VDC, a single VDC value, is either a real or an integer, depending on the declaration of the metafile descriptor function VDC TYPE. Subsequent tables use a single set of symbols, VDC, BVDC and VDCR, recognizing that they are computed differently depending on VDC TYPE.
- 8) The abstract parameter type VDC is a single value; a point, P, is an ordered pair of VDC.
- 9) The parameter type symbol CO corresponds to the data type CO of ISO/IEC 8632-1. It is either direct colour (CD) or indexed colour (CI), depending on the value specified in the COLOUR SELECTION MODE element. The associated octets per parameter and range symbols, BCO and COR, are thus either BCI and CIR or BCD and CDR respectively depending upon COLOUR SELECTION MODE.
- 10) To eliminate the need to support IEEE floating point in applications that do not need the dynamic range for parameters of type R and VDC, a fixed point real format is provided for scalars (such as line width, character spacing) and VDC. Fixed point reals consist of a (SI,UI) pair.

Fixed point reals (FX) apply to VDC, and to all metafile parameters of type R except for:

- a) the *metric scale factor* parameter of the SCALING MODE element;
- b) the *metric scale factor* parameter of the DEVICE VIEWPORT SPECIFICATION MODE element;

In this encoding, these parameters are always encoded as floating point.

- 11) CELL ARRAY colour can optionally specify 1, 2, 4, 8, 16, 24 or 32 bit precisions for cell colours, as well as using the default CI or CD precision.

The way in which the colour values in CELL ARRAY is represented is an extension of the representation of single colour values. The CELL ARRAY element has a 'cell representation flag' which may take one of two values:

- | | |
|---|---------------------------|
| 0 | run length representation |
| 1 | packed representation |

For PACKED mode, each row of the cell array is represented by an array of colour values without compression. Each row starts on a word boundary. No row length information is stored since all rows are the same length.

The colour data thus occupies $2n_y(1 + [(p \cdot n_x - 1)/16])$ octets, where n_x is the number of cells per row, n_y is the number of rows, p is the number of bits per colour, and $[..]$ denotes "the greatest integer in .."

For RUN LENGTH encoding, the data for each row begins on a word boundary and consists of run-length-lists for runs of constant colour value. Each 'run-length-list' consists of a count of a number of consecutive cells and the representation of that colour. In terms of the abstract terms above, the colour list is of format $\langle I, CO \rangle^*$ and its length is $\langle BI, BCO \rangle^*$. With the exception of the first run of a row, the integer count of each run immediately follows the colour specifier of the preceding run with no intervening padding.

- 12) Abstract parameter type Data Record, D, is encoded in this part similarly to string data. However, the constraints on character code values and the character set switching mechanisms (both those related to CHARACTER SET INDEX, and the purely ISO 2022 switching methods) do not necessarily apply to data records, as they do to the structurally similar S and SF parameters.

How the data are encoded, the meaning of the data bytes in the record, and the effect (if any) of character set switching mechanisms are part of the definitions of the individual Escape, GDP, and External elements to which the data record belongs.

The coding technique of the SDR data type (see *ref* 2086) is one valid form for a Data Record parameter. This form is recommended for GDP, Escape, and External element proposals submitted for Graphical Registration.

The coding tables in clause 7 will use the symbol D for the parameter type, and will use the S-related symbols for other information about the parameter.

- 13) The abstract parameter type VC, a single VC value, is either a real or an integer, depending on the declaration of the picture descriptor element DEVICE VIEWPORT SPECIFICATION MODE. When DEVICE VIEWPORT SPECIFICATION MODE is 'fraction of display surface', the value is real. When DEVICE

Representation of abstract parameter types

VIEWPORT SPECIFICATION MODE is 'millimetres with scale factor' or 'physical device coordinates', the value is integer. Subsequent tables use a single set of values, VC, BVC and VCR, recognising that they are computed differently depending on DEVICE VIEWPORT SPECIFICATION MODE.

- 14) The abstract parameter type VC is a single value; a viewport point, VP, is an ordered pair of VC.
- 15) The bitstream (BS) data type is encoded as a stream of binary digits (bits) packed in 16-bit unsigned integers. The BS data type is used in part 1 of this Standard for the compressed colour specifier lists of Tile Array elements. A bitstream type parameter shall be encoded in the Binary Encoding of this part with the smallest number of whole 16-bit words which will hold the bits of the parameter data. If the parameter data bits do not exactly fill an integral number of 16-bit words, the remaining bits in the last word shall be 0. The range for parameter type BS is not applicable.
- 16) The abstract parameter type CD is a 3-tuple or 4-tuple of CCO depending on COLOUR MODEL.
- 17) The structured data record (SDR) of part 1 of this International Standard is composed entirely of other standardized datatypes (including SDR itself) in a structure which is self-defining. SDR is encoded by encoding each of the component operands according to the normal encoding rules for its corresponding data type. The string of octets comprising the encoded operands is then treated as an operand of type S — it is preceded by a string count, short form or long form, and can be continued if long form (see note 6 above).
- 18) Ranges for enumerated and index parameters indicated by {n..m} in tables 3-10 refer to standardized values. Index ranges are subject to extension by registration.
- 19) The parameter type symbol SS corresponds to parameter type SS of ISO/IEC 8632-1. It is not a basic data type. It is a shorthand for data which can be VDC or Real, depending upon an associated specification mode. The associations of these modes with the various element parameters are defined in 5.1 of part 1.