

SEMI P39-0308

OASIS® – OPEN ARTWORK SYSTEM INTERCHANGE STANDARD

This standard was technically approved by the global Micropatterning Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on January 18, 2008. It was available at www.semi.org in February 2008 and on CD-ROM in March 2008. Originally published March 2004; previously published November 2005.

1 Purpose

1.1 The purpose of this specification is to define an interchange and encapsulation format for hierarchical integrated circuit mask layout information.

1.2 *Background* — In the fall of 2001, SEMI's Data Path Task Force formed a working group to define a successor to the venerable GDSII Stream format, which had served the I.C. industry as a *de facto* standard for layout interchange for more than two decades. The old format, limited by 16-bit and 32-bit internal integer fields, by its inefficient representation of cell-native geometric figures, and by high structural overhead, was becoming difficult to use for leading-edge designs, and file sizes were becoming unwieldy, in some cases growing to many tens of gigabytes. The successor format was chartered with several overall goals:

- Achieve at least an order-of-magnitude file size improvement compared to GDSII Stream.
- Remove all 16-bit and 32-bit integer width restrictions—make the new format fully 64-bit capable.
- Efficiently represent cells with large payloads of flat native geometric figures.
- Provide a richer information palette to facilitate interchange of layout-related information between design and manufacturing.

1.2.1 In the months leading up to the formation of the SEMI Data Path Task Force, International Sematech sponsored a series of meetings focusing on Mask EDA issues. Many of the Task Force participants were also involved in these Sematech meetings, and carried forward much useful information from those sessions into the definition of this specification.

2 Scope

2.1 This format is designed primarily to encapsulate hierarchical mask layout for interchange between systems such as EDA software, mask writing tools, and mask inspection/repair tools.

2.2 This format is designed to be both hardware- and software-independent.

NOTICE: This standard does not purport to address safety issues, if any, associated with its use. It is the responsibility of the users of this standard to establish appropriate safety and health practices and determine the applicability of regulatory or other limitations prior to use.

3 Limitations

3.1 Use of extension records such as XNAME, XELEMENT, and XGEOMETRY may impair interoperability between tools. It is recommended that these extensions be used primarily for prototyping, and that interoperability be maintained through the formal inclusion of extensions to this specification.

4 Referenced Standards and Documents

4.1 *IEEE Standard*¹

IEEE 754-1985 — IEEE Standard for Binary Floating-Point Arithmetic

¹ Institute of Electrical and Electronics Engineers, IEEE Operations Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, New Jersey 08855-1331, USA. Telephone: 732.981.0060; Fax: 732.981.1721; <http://www.ieee.org>

4.2 ISO Standards²

ISO 646-IRV — “US-ASCII” Character Set

ISO 3309 — Information technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures – Frame structure

4.3 IETF Standards³

RFC 1951 — DEFLATE Compressed Data Format Specification Version 1.3

NOTICE: Unless otherwise indicated, all documents cited shall be the latest published versions.

5 Terminology

5.1 Abbreviations and Acronyms

5.1.1 *BNF* — Backus-Naur Form

5.1.2 *EDA* — Electronic Design Automation

5.1.3 *OASIS®* — Open Artwork System Interchange Standard⁴

5.2 Definitions

5.2.1 Most definitions of terminology specific to OASIS are found within the text of the paragraphs that contain them.

5.2.2 *cell* — a named object in a layout hierarchy, containing native geometric information, annotation information, and/or placements of other cells.

5.2.3 *geometry* — a two-dimensional geometric figure such as a polygon, rectangle, trapezoid, path, circle, etc. with inherent attributes of *layer* and *datatype*.

5.2.4 *placement* — a specification by reference that a copy of a cell is to be placed within the coordinate space of another cell at a particular location, orientation, and scale. Cell placement is the fundamental mechanism which makes hierarchy within the OASIS file possible.

5.2.5 *property* — an annotation element consisting of a name plus an optional list of values, supplying descriptive information about the characteristics of the file or one of its components.

5.2.6 *record* — the principal data division in an OASIS file.

5.2.7 *text element* — an annotation element consisting of an (x,y) coordinate point and an associated string.

5.3 Symbols

5.3.1 “->” — indicates a mapping of an argument to its contents or its meaning.

6 OASIS Basics

6.1 An OASIS file is a sequence of **bytes** divided into **records**. The length of a record is discernible from its structure and is not explicit (in contrast to GDSII Stream, where all record lengths are explicit).

6.2 An OASIS file has the following overall syntax (using the modified BNF notation described in § 36). Individual record types appear in bold uppercase and are described in more detail in following sections.

² International Organization for Standardization, ISO Central Secretariat, 1 rue de Varembe, Case postale 56, CH-1211 Geneva 20, Switzerland. Telephone: 41.22.749.01.11; Fax: 41.22.733.34.30; <http://www.iso.ch>

³ Internet Engineering Task Force; <http://www.ietf.org>

⁴ OASIS® is a registered trademark of Thomas Grebinski, and is licensed for use by SEMI.

```

<oasis-file> -> <magic-bytes> START { CBLOCK | PAD | PROPERTY | <cell> | <name> } * END
<name> -> { CELLNAME | TEXTSTRING | LAYERNAME | PROPNAME | PROPSTRING | XNAME }
<cell> -> { CELL { CBLOCK | PAD | PROPERTY | XYRELATIVE | XYABSOLUTE | <element> } * }
<element> -> { <geometry> | PLACEMENT | TEXT | XELEMENT }
<geometry> -> { RECTANGLE | POLYGON | PATH | TRAPEZOID | CTRAPEZOID | CIRCLE | XGEOMETRY }

```

6.3 An OASIS file may represent a complete layout hierarchy, a portion of a layout hierarchy, or multiple layout hierarchies. These interpretations are not intrinsic to the format and are governed by application semantics only. Each OASIS file must be syntactically complete—it must begin with <magic-bytes> and contain at least a **START** and **END** record.

6.4 The <magic-bytes> element is a sequence of 13 ASCII characters: “%SEMI-OASIS<CR><NL>” where <CR><NL> represents the ASCII hexadecimal sequence 0D 0A. It is provided as a recognition signature to make OASIS files easily identifiable to the UNIX *file* utility. (The intent of the carriage return and newline is to help detect corruption by FTP programs operating in non-binary mode.)

6.5 **EXCEPTION HANDLING** — OASIS processors should treat any deviation from the syntax presented in this document as a fatal error. OASIS readers are not required to implement syntax-check preprocessing in order to be considered compliant with this specification. The sequence in which exceptions are detected and reported is entirely application-dependent. In addition, for access requests which do not require the interpretation of the entire file (such as retrieval of a single cell or a subset of the cells within the file), this specification does not require OASIS readers to exhaustively check the validity of the entire file.

7 Data Constructs

7.1 Bytes

7.1.1 A byte is a fixed-length 8-bit value. Bit patterns for bytes are shown with the least significant bit (bit 0) on the right.

7.2 Integers

7.2.1 An **unsigned-integer** is an N-byte ($N > 0$) integer value. The low-order byte appears first in the OASIS format. Integer byte length is variable and integers are represented as *byte-continuations* where the most significant bit of each byte except the last in the chain is a 1; the remaining seven bits in each byte are concatenated to form the actual integer value itself. There are no restrictions on integer byte length (and hence, magnitude).

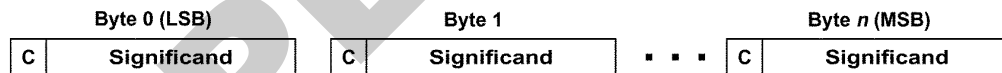


Figure 1
Unsigned-Integer Representation

Table 1 Unsigned-Integer Examples

Unsigned-Integer Value	Binary Representation
0	00000000
127	01111111
128	10000000 00000001
16,383	11111111 01111111
16,384	10000000 10000000 00000001

7.2.2 A **signed-integer** follows the same byte-continuation scheme as an *unsigned-integer*, and is stored in signed-magnitude form, with the *significand* left-shifted one bit and the sign bit stored in the least significant bit of the low-order (first) byte. A sign bit of 0 indicates a positive number, and a sign bit of 1 indicates a negative number. Both representations of zero (+0 and −0) should be treated as numerically equivalent for the purposes of comparison.

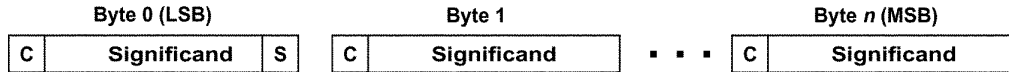


Figure 2
Signed-Integer Representation

Table 2 Signed Integer Examples

<i>Signed Integer Value</i>	<i>Binary Representation</i>
0	00000000
+1	00000010
-1	00000011
+63	01111110
-64	10000001 00000001
+8,191	11111110 01111111
-8,192	10000001 10000000 00000001

7.2.3 EXCEPTION HANDLING — OASIS processors which only support integer data in a restricted space (e.g., 32-bit space) should treat any magnitude outside of this space as a fatal error.

7.3 Reals

7.3.1 A **real** number may be stored in one of several rational forms, or as a single-precision 4-byte (*ieee-4*) or double-precision 8-byte (*ieee-8*) floating point value. The rational forms are usually more compact than the floating-point forms, and have the advantage of being able to precisely represent many values which can only be approximated by the binary floating point representation. The type of representation is stored in an *unsigned-integer* which precedes the significant portion of the real:

Table 3 Real Number Types

<i>Format</i>	<i>Meaning</i>
'0' unsigned-integer	Positive whole number
'1' unsigned-integer	Negative whole number
'2' unsigned-integer	Positive reciprocal
'3' unsigned-integer	Negative reciprocal
'4' unsigned-integer unsigned-integer	Positive ratio
'5' unsigned-integer unsigned-integer	Negative ratio
'6' IEEE-4-byte-float	Single-precision floating point
'7' IEEE-8-byte-float	Double-precision floating point

7.3.2 In types 0 and 1, the *real* is a whole number—its fractional portion is zero. In types 2 and 3, the *unsigned-integer* represents the denominator of a reciprocal, with an implicit numerator of 1. Types 4 and 5 are ratios, with the numerator listed first, followed by the denominator. Types 6 and 7 are binary floating point numbers in IEEE 754-1985 format, with the least significant byte of the *fraction* (byte 0) stored first.

7.3.3 EXCEPTION HANDLING — For types 2–5, a denominator of 0 should be treated as a fatal error. A type outside the range of 0–7 should be treated as a fatal error.

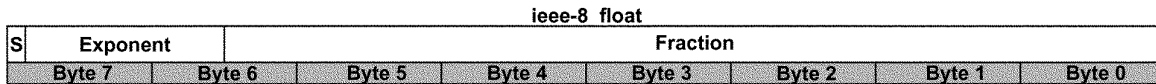
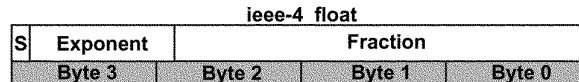


Figure 3
IEEE Floating Point Formats

Table 4 Real Number Examples

Value	Rational Form	IEEE-4 Form
0.0	00000000 00000000	00000110 00000000 00000000 00000000
1.0	00000000 00000001	00000110 00000000 00000000 10000000 00111111
-0.5	00000011 00000010	00000110 00000000 00000000 00000000 10111111
0.3125	00000100 00000101 00010000	00000110 00000000 00000000 10100000 00111110
1/3	00000010 00000011	00000110 10101011 10101010 10101010 00111110
-2/13	00000101 00000010 00001101	00000110 11011001 10001001 00011101 10111110

7.4 Strings

7.4.1 A **string** is a sequence of zero or more bytes (“characters”) preceded by an *unsigned-integer* representing the number of characters in the string:

string -> length byte*

Strings in OASIS are further sub-typed by semantic. A **b-string** (“binary string”) is a string which may contain any combination of 8-bit character codes in any sequence. An **a-string** (“ASCII string”) may contain only *printable* ASCII character codes (hexadecimal 21-7E) plus the SP (space) character (hexadecimal 20), in any sequence. An **n-string** (“name string”) may contain only *printable* ASCII character codes (hexadecimal 21-7E), and must have a length greater than zero.

7.4.2 The set of *printable* ASCII characters consists of hexadecimal character codes 21-7E. In ascending order of character code, we have:

!"#\$%&'()*+,-./0123456789:;<=>?@	[21-40]
ABCDEFGHIJKLMN O PQRSTU VWXYZ[\]^_`	[41-60]
abcdefghijklmnopqrstuvwxyz{ }~	[61-7E]

This excludes space (SP), tabs (HT, VT), and all other control characters.

7.4.3 **EXCEPTION HANDLING** — OASIS processors should treat illegal characters in *a-strings* or *n-strings* as fatal errors. Zero-length *n-strings* should also be treated as fatal errors.

7.5 Deltas

7.5.1 A **delta** represents geometric data (co-ordinates, vectors, planar offsets, etc.).

7.5.2 A **1-delta** is stored as a *signed-integer* and represents a horizontal or vertical displacement. The sign bit, bit 0, encodes direction: 0 for east or north, 1 for west or south. The remaining bits are the magnitude. Horizontal or vertical alignment is implied by context.

7.5.3 A **2-delta** is stored as an *unsigned-integer* and represents a horizontal or vertical displacement. Bits 0-1 encode direction: 0 for east, 1 for north, 2 for west, and 3 for south. The remaining bits are the magnitude.

7.5.4 A **3-delta** is stored as an *unsigned-integer* and represents a horizontal, vertical, or 45° diagonal displacement. Bits 0-2 encode direction: 0 for east, 1 for north, 2 for west, 3 for south, 4 for northeast, 5 for northwest, 6 for

southwest, and 7 for southeast. The remaining bits are the magnitude (for horizontal and vertical deltas) or the magnitude of the projection onto the x- or y-axis (for 45° deltas).

7.5.5 A **g-delta** has two alternative forms and is stored either as a single *unsigned-integer* or as a pair of *unsigned-integers*. The first form is indicated when bit 0 is zero, and represents a horizontal, vertical, or 45° diagonal displacement, with bits 1–3 encoding direction, and the remaining bits storing the magnitude, in the same fashion as a 3-delta. The second form represents a general (x,y) displacement and is a pair of *unsigned-integers*. Bit 0 of the first integer is 1. Bit 1 of the first integer is the x-direction (0 for east, 1 for west). The remaining bits of the first integer represent the magnitude in the x-direction. Bit 0 of the second integer is the y-direction (0 for north, 1 for south). The remaining bits of the second integer represent the magnitude in the y-direction. Both forms may appear in a list of *g-deltas*.

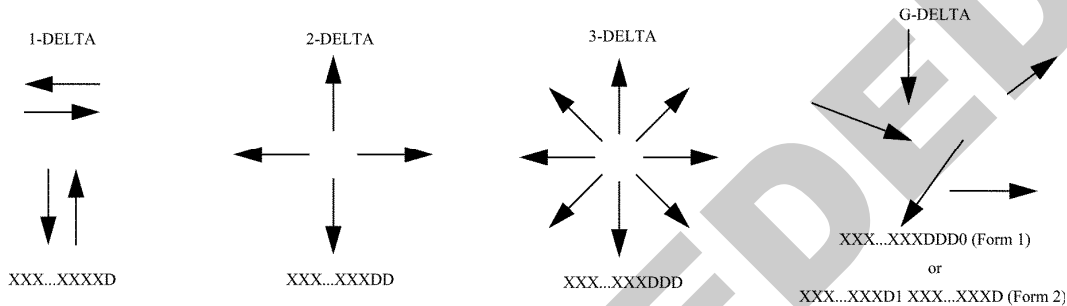


Figure 4
Delta Types

Table 5 Delta Examples

Bit Pattern	Type	Meaning
11111001 00100011	1-delta	$\Delta = -2300$
11111000 00100011	1-delta	$\Delta = +2300$
10011000 00101010	2-delta	$\Delta x = +1350$
10011011 00101010	2-delta	$\Delta y = -1350$
11001101 00000001	3-delta	$\Delta x = -25, \Delta y = +25$
11010111 00000111	3-delta	$\Delta x = +122, \Delta y = -122$
11101001 00000011 01111010	<i>g-delta</i> ₂	$\Delta x = +122, \Delta y = +61$
11101100 00000101	<i>g-delta</i> ₁	$\Delta x = -46, \Delta y = -46$
10111011 00000001 10110111 00001111	<i>g-delta</i> ₂	$\Delta x = -46, \Delta y = -987$

7.6 Repetitions

7.6.1 A **repetition** represents an “array” of cell placements, geometries, or text elements. The repetition is part of the **PLACEMENT**, **<geometry>**, or **TEXT** record itself. A repetition consists of an *unsigned-integer* which encodes the type, followed by any related repetition parameters:

Table 6 Repetition Types

TYPE	Format
0	re-use the previous repetition definition
1	x-dimension y-dimension x-space y-space
2	x-dimension x-space
3	y-dimension y-space
4	x-dimension x-space₁ ... x-space_{N-1}
5	x-dimension grid x-space₁ ... x-space_{N-1}
6	y-dimension y-space₁ ... y-space_{M-1}
7	y-dimension grid y-space₁ ... y-space_{M-1}
8	n-dimension m-dimension n-displacement m-displacement
9	dimension displacement
10	dimension displacement₁ ... displacement_{p-1}
11	dimension grid displacement₁ ... displacement_{p-1}

7.6.1.1 **x-dimension**, **y-dimension**, **x-space**, **y-space**, **dimension**, **n-dimension**, **m-dimension**, and **grid** are all *unsigned-integers*. **displacement**, **n-displacement**, and **m-displacement** are *g-deltas*.

7.6.2 **TYPE 0** indicates that the previous repetition description, stored in modal variable *repetition*, is to be re-used (see § 10). No additional values are stored with this type.

7.6.3 **TYPE 1** is an N-column ($N > 1$) by M-row ($M > 1$) matrix with uniform horizontal and vertical spacing between the elements. **x-dimension** is $N - 2$ and **y-dimension** is $M - 2$. The (*x-offset*, *y-offset*) (cumulative spacing in the (horizontal,vertical) direction) of element (*i,j*) of the repetition ($i = 0, \dots, N-1$ and $j = 0, \dots, M-1$) is ($i \times \mathbf{x-space}$, $j \times \mathbf{y-space}$).

7.6.4 **TYPE 2** is an N-column ($N > 1$) by 1-row vector with uniform horizontal spacing between the elements. **x-dimension** is $N - 2$. The (*x-offset*, *y-offset*) (cumulative spacing in the (horizontal,vertical) direction) of element *i* of the repetition ($i = 0, \dots, N-1$) is ($i \times \mathbf{x-space}$, 0).

7.6.5 **TYPE 3** is a 1-column by M-row ($M > 1$) vector with uniform vertical spacing between the elements. **y-dimension** is $M - 2$. The (*x-offset*, *y-offset*) (cumulative spacing in the (horizontal,vertical) direction) of element *j* of the repetition ($j = 0, \dots, M-1$) is (0, $j \times \mathbf{y-space}$).

7.6.6 **TYPE 4** is an N-column ($N > 1$) by 1-row vector with (potentially) non-uniform horizontal spacing between the elements. **x-dimension** is $N - 2$. The (*x-offset*, *y-offset*) (cumulative spacing in the (horizontal,vertical) direction) of element *i* of the repetition ($i = 0, \dots, N-1$) is ($\mathbf{x-space}_0 + \dots + \mathbf{x-space}_i$, 0), with $\mathbf{x-space}_0 = 0$.

7.6.7 **TYPE 5** is identical to **TYPE 4**, except that all offset values must be multiplied by **grid** during expansion of the repetition.

7.6.8 **TYPE 6** is a 1-column by M-row ($M > 1$) vector with (potentially) non-uniform vertical spacing between the elements. **y-dimension** is $M - 2$. The (*x-offset*, *y-offset*) (cumulative spacing in the (horizontal,vertical) direction) of element *j* of the repetition ($j = 0, \dots, M-1$) is (0, $\mathbf{y-space}_0 + \dots + \mathbf{y-space}_j$), with $\mathbf{y-space}_0 = 0$.

7.6.9 **TYPE 7** is identical to **TYPE 6**, except that all offset values must be multiplied by **grid** during expansion of the repetition.

7.6.10 **TYPE 8** is an N ($N > 1$) by M ($M > 1$) repetition with uniform and (potentially) diagonal displacements between the elements. **n-dimension** is $N - 2$ and **m-dimension** is $M - 2$. Defining **n-displacement** in terms of its components **nx-space** and **ny-space** (and similarly for **m-displacement**), the (*x-offset*, *y-offset*) (cumulative spacing in the (horizontal,vertical) direction) of element (*i,j*) of the repetition ($i = 0, \dots, N-1$ and $j = 0, \dots, M-1$) is ($i \times \mathbf{nx-space} + j \times \mathbf{mx-space}$, $i \times \mathbf{ny-space} + j \times \mathbf{my-space}$).

7.6.11 **TYPE 9** is a P-element ($P > 1$) repetition with uniform and (potentially) diagonal displacements between the elements. **dimension** is $P - 2$. Defining **displacement** in terms of its components **x-space** and **y-space**, the (*x-offset*,

y-offset) (cumulative spacing in the (horizontal,vertical) direction) of element k of the repetition ($k = 0, \dots, P-1$) is ($k \times \mathbf{x-space}$, $k \times \mathbf{y-space}$).

7.6.12 **TYPE 10** is a P -element ($P > 1$) repetition with (potentially) non-uniform and arbitrary two-dimensional displacements between the elements. **dimension** is $P - 2$. Defining **displacement_k** in terms of its components **x-space_k** and **y-space_k**, the (**x-offset**, **y-offset**) (cumulative spacing in the (horizontal,vertical) direction) of element k of the repetition ($k = 0, \dots, P-1$) is (**x-space₀** + ... + **x-space_k**, **y-space₀** + ... + **y-space_k**) with **x-space₀** = **y-space₀** = 0).

7.6.13 **TYPE 11** is identical to **TYPE 10**, except that all offset values must be multiplied by **grid** during expansion of the repetition.

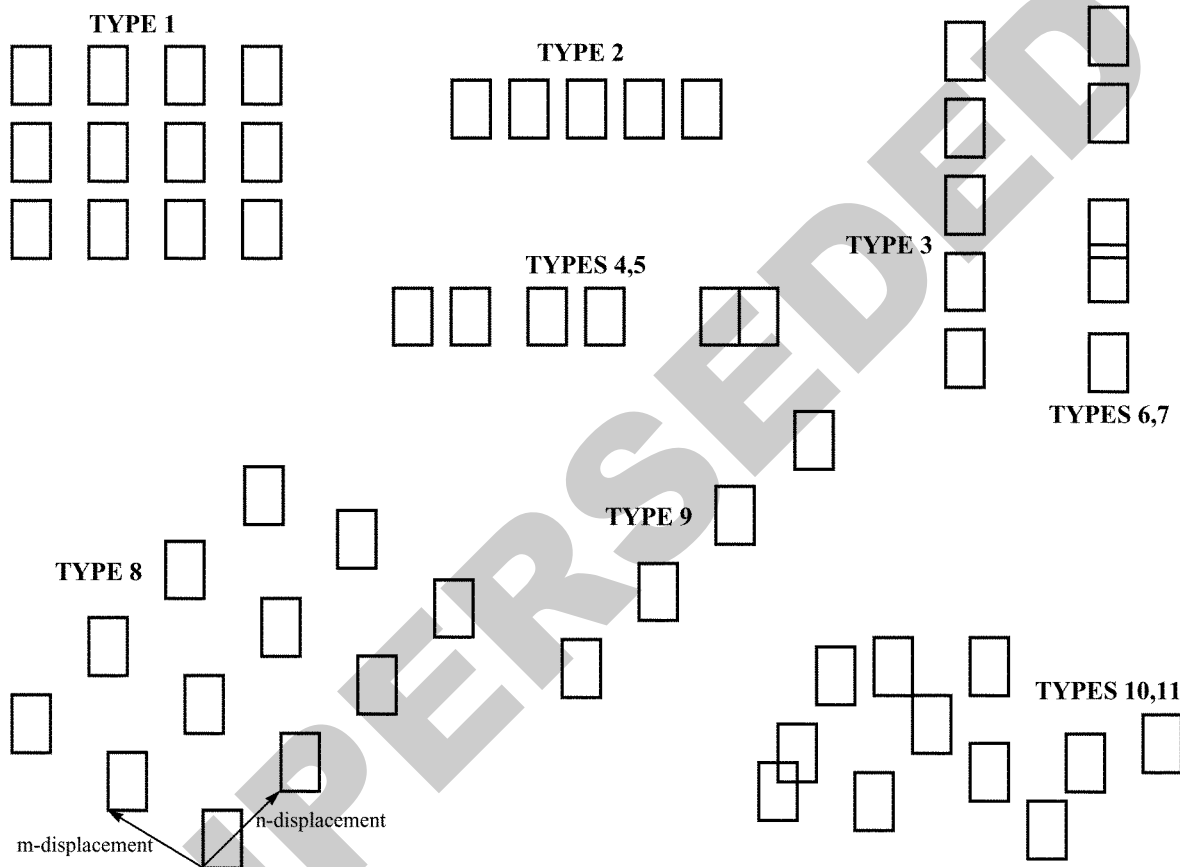


Figure 5
Repetition Types

7.6.14 **EXCEPTION HANDLING** — A repetition type outside the range of 0 to 11 should be treated as a fatal error. A repetition type of 0 may not be the first repetition type used within a cell.

7.7 Point Lists

7.7.1 A **point-list** represents a list of geometric coordinates for polygons or paths, and consists of an **unsigned-integer** denoting its type, followed by a list of **deltas**, in one of several formats. The initial vertex at (**x,y**) is supplied by the **POLYGON** or **PATH** record and is not part of the **point-list**; **vertex-count** (an **unsigned-integer**) is the number of points or deltas, excluding the initial vertex and any implicit vertices.

Table 7 Point List Types

Type	Format	Description
0	vertex-count [1-delta [... 1-delta]]	Implicit <i>manhattan</i> delta point-list (horizontal-first)
1	vertex-count [1-delta [... 1-delta]]	Implicit <i>manhattan</i> delta point-list (vertical-first)
2	vertex-count [2-delta [... 2-delta]]	Explicit <i>manhattan</i> delta point-list
3	vertex-count [3-delta [... 3-delta]]	Explicit <i>octangular</i> delta point-list
4	vertex-count [g-delta [... g-delta]]	Explicit all-angle delta point-list
5	vertex-count [g-delta [... g-delta]]	Explicit all-angle double-delta point-list

7.7.2 A *point-list* of type 0 consists of a list of *1-deltas*, representing alternating horizontal and vertical relative displacements, with the first displacement implicitly horizontal. When describing a polygon *point-list* in this form, the *final two* displacements are omitted, since they can be unambiguously implied from the current point, the last edge, and the starting point. When describing a polygon, **vertex-count** must be an even number greater than or equal to 2.

7.7.3 A *point-list* of type 1 consists of a list of *1-deltas*, representing alternating vertical and horizontal relative displacements, with the first displacement implicitly vertical. When describing a polygon *point-list* in this form, the *final two* displacements are omitted, since they can be unambiguously implied from the current point, the last edge, and the starting point. When describing a polygon, **vertex-count** must be an even number greater than or equal to 2.

7.7.4 A *point-list* of type 2 consists of a list of *2-deltas*, representing a series of *manhattan* relative displacements. When describing a polygon *point-list* in this form, the final displacement is omitted, since the polygon is assumed to be implicitly closed, but this final implicit displacement must be a *manhattan* displacement, with either $\Delta x = 0$ or $\Delta y = 0$.

7.7.5 A *point-list* of type 3 consists of a list of *3-deltas*, representing a series of *octangular* relative displacements. When describing a polygon *point-list* in this form, the final displacement is omitted, since the polygon is assumed to be implicitly closed, but this final implicit displacement must be an *octangular* displacement at an angle that is an integral multiple of 45° .

7.7.6 A *point-list* of type 4 consists of a list of *g-deltas*, representing a series of any-angle relative displacements. When describing a polygon *point-list* in this form, the final displacement is omitted, since the polygon is assumed to be implicitly closed.

7.7.7 A *point-list* of type 5 consists of a list of *g-deltas*, representing a series of adjustments to a relative displacement vector, with the initial vector set to $(\Delta x = 0, \Delta y = 0)$. To calculate the coordinates of each successive point, the x and y components of each successive *g-delta* are added to the relative displacement vector, which in turn describes the relative displacement from the current point to the next point. When describing a polygon *point-list* in this form, the final displacement is omitted, since the polygon is assumed to be implicitly closed. This form of *point-list* is intended to allow more compact representation of polygons and paths which are approximations of large-field curvilinear figures on a fine grid, where the curvature is not extreme.

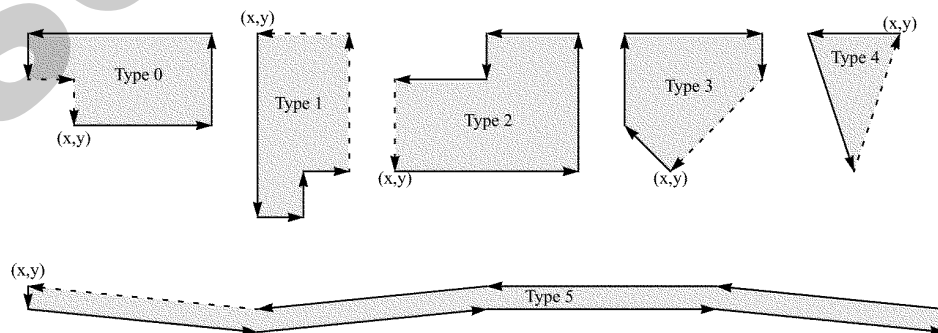


Figure 6
Point List Describing Polygons

Table 8 Polygon Point List for Figure 6

Type	Bit Pattern
0	00000000 00000100 00001100 00001000 00010001 00000101
1	00000001 00000100 00010001 00000100 00000100 00000100
2	00000010 00000101 00100000 00011001 00010010 00001011 00010010
3	00000011 00000100 00010101 00100001 00110000 00010011
4	00000100 00000010 01000100 00001001 00001101
5	00000101 00001001 00000001 00000011 00101001 00000000 00000001 00000100 00000001 00000011 00000001 00000011 00101011 00000100 00101011 00000000 00000001 00000011 00000001 00000011

7.7.8 EXCEPTION HANDLING — A *point-list* type outside the range of 0 to 5 should be treated as a fatal error. For *point-list* types 0–1, successive coincident points and/or adjacent colinear edges are not permitted. A *non-manhattan* implicit closing vector for a polygon using *point-list* type 2, or a *non-octangular* implicit closing vector for a polygon using *point-list* type 3 should be treated as a fatal error. For polygons using *point-list* types 0–1, a vertex count which is odd or less than 2 should be treated as a fatal error.

7.8 Property Values

7.8.1 A **property-value** stores one element of a property value list. It consists of an *unsigned-integer* which encodes its type, followed by either the value itself or a reference number. Types 0–7 are *reals* which conform to the scheme described in Table 3.

Table 9 Property Value Types

Type	Format
0–7	real (see Table 3)
8	unsigned-integer
9	signed-integer
10	a-string
11	b-string
12	n-string
13	propstring-reference-number (implied a-string)
14	propstring-reference-number (implied b-string)
15	propstring-reference-number (implied n-string)

7.8.2 EXCEPTION HANDLING — A *property-value* type outside the range of 0 to 15 should be treated as a fatal error. Use of a **propstring-reference-number** for which there is no corresponding **PROPSTRING** record within the same OASIS file should be treated as a fatal error.

8 CELL Referencing

8.1 As in GDSII Stream, cells in OASIS are identified by *name*. The **CELL** record not only introduces a cell definition but also defines its name. **PLACEMENT** records refer by name to the cell being placed. As in GDSII Stream, there are no “anonymous” cells in OASIS.

9 Layers, Datatypes, and Texttypes

9.1 As in GDSII Stream, every <geometry> has associated with it a *layer number* and a *datatype number* and every text element has associated with it a *textlayer number* and a *texttype number*.

10 Modal Variables

10.1 For compaction purposes, selected data elements in many OASIS records may be implicitly specified through the use of *modal variables* or stored state. At the beginning of the file, and whenever a **CELL** or **<name>** record is encountered, all modal variables with the exception of *placement-x*, *placement-y*, *geometry-x*, *geometry-y*, *text-x*, and *text-y*, are set to a state of *undefined*; the exceptions just mentioned are set to 0. As various elements appear in the cell's description, modal variables related to those elements are set from the elements' definitions. These modal variables can then be used implicitly by successive elements. A modal variable may hold a single value such as *geometry-w*, or a multi-variable structure such as a *repetition*.

Table 10 Modal Variables

<i>Modal Variables</i>	<i>Related Records</i>
repetition	PLACEMENT, TEXT, POLYGON, PATH, RECTANGLE, TRAPEZOID, CTRAPEZOID, CIRCLE, XGEOMETRY
placement-x, placement-y, placement-cell	PLACEMENT
layer, datatype	POLYGON, PATH, RECTANGLE, TRAPEZOID, CTRAPEZOID, CIRCLE, XGEOMETRY
textlayer, texttype, text-x, text-y, text-string	TEXT
geometry-x, geometry-y	POLYGON, PATH, RECTANGLE, TRAPEZOID, CTRAPEZOID, CIRCLE, XGEOMETRY
xy-mode	PLACEMENT, TEXT, POLYGON, PATH, RECTANGLE, TRAPEZOID, CTRAPEZOID, CIRCLE, XGEOMETRY, XYABSOLUTE, XYRELATIVE
geometry-w, geometry-h	RECTANGLE, TRAPEZOID, CTRAPEZOID
polygon-point-list	POLYGON
path-halfwidth, path-point-list, path-start-extension, path-end-extension	PATH
ctrapezoid-type	CTRAPEZOID
circle-radius	CIRCLE
last-property-name, last-value-list	PROPERTY

10.2 Modal variable *xy-mode* governs the interpretation of the **x** and **y** fields for those related record types indicated in Table 10. Two interpretation modes are provided: *absolute* and *relative*. See § 21 for a discussion of how these two modes work.

10.3 **EXCEPTION HANDLING** — An OASIS record which implicitly references a modal variable which is in the *undefined* state should be treated as a fatal error.

11 Records

11.1 The basic unit of information in an OASIS file is a **record**. A record consists of a single *unsigned-integer* which encodes the **record-ID**, followed by the remainder of the record's descriptive data. In this specification, *record-ID* values are displayed as decimal numbers enclosed in apostrophes.

11.2 The **CBLOCK** record is a special case since it encapsulates a series of ordinary records in byte-compressed form. When a **CBLOCK** record is encountered while reading an OASIS file, it is first necessary to decompress its data, which will produce one or more ordinary records, which can in turn be decoded. For more information on **CBLOCK** records refer to § 35.

11.3 Most records have an implicit length—the record must be parsed and decoded in order to determine its length. The **XNAME**, **XELEMENT**, and **XGEOMETRY** records are exceptions to this. They encapsulate all of their user-defined data in a single variable-length *b-string*, so they can be used for prototyping new record types, hiding embedded proprietary data, supporting local non-interoperable extensions, etc. without rendering an OASIS file illegible to older readers, which can simply note the string length and skip over the record.

11.4 **EXCEPTION HANDLING** — OASIS processors should treat the nesting of a **CBLOCK** record within another **CBLOCK** record as a fatal error.

12 PAD Record

12.1 A **PAD** record provides a simple way to reserve space within an OASIS file. It has the following format:

‘0’

12.2 **PAD** records may be inserted between any other two records.

12.3 **EXCEPTION HANDLING** — The presence of a **PAD** record before the **START** record or after the **END** record should be treated as a fatal error.

13 START Record

13.1 A **START** record identifies the beginning of an OASIS file, and immediately follows the <magic-bytes> sequence described in ¶ 6.4. It has the following format:

‘1’ version-string unit offset-flag [table-offsets]

13.2 The **version-string** is an *a-string* whose value is “1.0” for this version of the OASIS specification. Version “1.0” corresponds to the OASIS format as described in this document.

13.3 The **unit** declaration is a positive *real* number which specifies the global precision of the OASIS file’s coordinate system in grid steps per micron.

13.4 **offset-flag** (an *unsigned-integer*) is 0 when the **table-offsets** structure is stored in the **START** record; **offset-flag** is 1 when the **table-offsets** structure is instead stored in the **END** record. The option of storing **table-offsets** in the **END** record is provided to make it possible to write an OASIS file sequentially, with no seek-and-update access required, while still providing cell-level random-access capability for subsequent readers of that OASIS file.

13.5 The **table-offsets** structure consists of 6 pairs of *unsigned-integers*. Each pair consists of a *flag* field, and a corresponding *byte-offset* field, in the following order:

Table 11 Table Offset Order

<i>Flag</i>	<i>Byte-Offset</i>
cellname-flag	cellname-offset
textstring-flag	textstring-offset
propname-flag	propname-offset
propstring-flag	propstring-offset
layername-flag	layername-offset
xname-flag	xname-offset

13.6 Each of the *flag* fields is either 1, indicating *strict* mode, or 0, indicating *non-strict* mode, for its respective table. The corresponding *byte-offset* field indicates the position of the first record of its respective table relative to the first byte (byte 0) of the OASIS file. A *byte-offset* of 0 indicates the absence of that particular table.

13.7 In *non-strict* mode, records of the corresponding type may occur anywhere in the file, even if some of them have been gathered into a table pointed to by the corresponding *byte-offset*.

13.8 In *strict* mode, all records of the corresponding type (plus any associated **PROPERTY** records) have been gathered into a single contiguous table pointed to by the corresponding *byte-offset*. **PAD** records are also permitted in *strict* mode tables. In addition, *strict* mode guarantees that all references to the corresponding class of objects (names, strings, or cells) are made exclusively by *reference-number*.

13.9 When a given *strict* mode table has been encapsulated within one or more **CBLOCK** records, the corresponding *byte-offset* should point to the first byte of the first **CBLOCK** record containing that table, and the