

# Appendix C. Internal Representation of Data Types

This appendix contains the detailed internal representations of the PDS standard data types listed in Table 3.2 of the *Data Type Definitions* chapter of this document.

## Chapter Contents

Appendix C. Internal Representation of Data Types .....	C-1
C.1 MSB_INTEGER .....	C-2
C.2 MSB_UNSIGNED_INTEGER .....	C-4
C.3 LSB_INTEGER .....	C-6
C.4 LSB_UNSIGNED_INTEGER .....	C-8
C.5 IEEE_REAL .....	C-10
C.6 IEEE_COMPLEX .....	C-13
C.7 PC_REAL .....	C-14
C.8 PC_COMPLEX .....	C-17
C.9 VAX_REAL, VAXG_REAL .....	C-18
C.10 VAX_COMPLEX, VAXG_COMPLEX .....	C-22
C.11 MSB_BIT_STRING .....	C-23
C.12 LSB_BIT_STRING .....	C-25

## C.1 MSB\_INTEGER

**Aliases:** INTEGER  
 MAC\_INTEGER  
 SUN\_INTEGER

This section describes the signed integers stored in Most Significant Byte first (MSB) order. In this section the following definitions apply:

- b0 – b3* Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2, and b3)
- i-sign* Integer sign bit (bit 7 in the highest-order byte)
- i0 – i3* Arrangement of bytes in the integer, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., lowest value = bit 0, highest value = bit 7) in the following way:

4-byte integers:

- In *i0*, bits 0-7 represent  $2^{**0}$  through  $2^{**7}$
- In *i1*, bits 0-7 represent  $2^{**8}$  through  $2^{**15}$
- In *i2*, bits 0-7 represent  $2^{**16}$  through  $2^{**23}$
- In *i3*, bits 0-6 represent  $2^{**24}$  through  $2^{**30}$

2-byte integers:

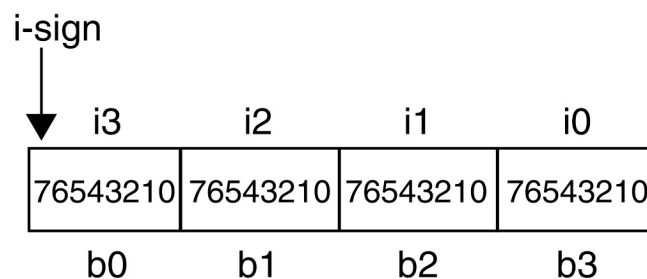
- In *i0*, bits 0-7 represent  $2^{**0}$  through  $2^{**7}$
- In *i1*, bits 0-6 represent  $2^{**8}$  through  $2^{**14}$

1-byte integers:

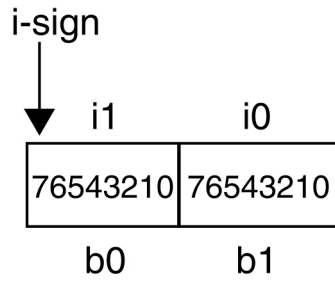
- In *i0*, bits 0-6 represent  $2^{**0}$  through  $2^{**6}$

Negative integers are represented in two's complement.

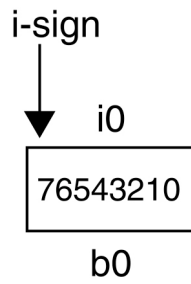
### C.1.1 MSB 4-byte Integer



### C.1.2 MSB 2-byte Integer



### C.1.3 MSB 1-byte Integer



## C.2 MSB\_UNSIGNED\_INTEGER

**Aliases:** UNSIGNED\_INTEGER  
 MAC\_UNSIGNED\_INTEGER  
 SUN\_UNSIGNED\_INTEGER

This section describes unsigned integers stored in Most Significant Byte first (MSB) format. In this section the following definitions apply:

*b0 – b3* Arrangement of bytes as they appear when read from a file (e.g., read *b0* first, then *b1*, *b2* and *b3*)

*i0 – i3* Arrangement of bytes in the integer, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., lowest value = bit 0, highest value = bit 7), in the following way:

4-bytes:

In *i0*, bits 0-7 represent  $2^{**0}$  through  $2^{**7}$

In *i1*, bits 0-7 represent  $2^{**8}$  through  $2^{**15}$

In *i2*, bits 0-7 represent  $2^{**16}$  through  $2^{**23}$

In *i3*, bits 0-7 represent  $2^{**24}$  through  $2^{**31}$

2-bytes:

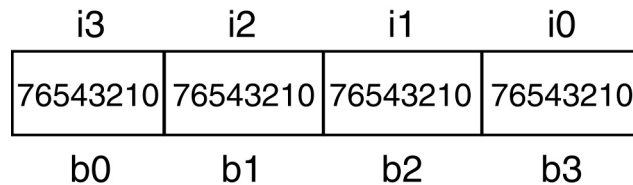
In *i0*, bits 0-7 represent  $2^{**0}$  through  $2^{**7}$

In *i1*, bits 0-7 represent  $2^{**8}$  through  $2^{**15}$

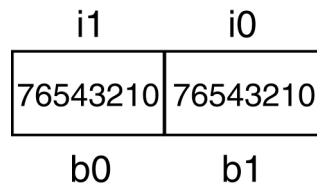
1-byte:

In *i0*, bits 0-7 represent  $2^{**0}$  through  $2^{**7}$

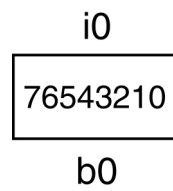
### C.2.1 MSB 4-byte Unsigned Integers



### C.2.2 MSB 2-byte Unsigned Integers



### C.2.3 MSB 1-byte Unsigned Integers



## C.3 LSB\_INTEGER

**Aliases:** PC\_INTEGER  
VAX\_INTEGER

This section describes signed integers stored in Least Significant Byte first (LSB) order. In this section the following definitions apply:

*b0 – b3* Arrangement of bytes as they appear when reading a file (e.g., read byte b0 first, then b1, b2 and b3)

*i-sign* Integer sign bit – bit 7 in the highest order byte

*i0 – i3* Arrangement of bytes in the integer, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., lowest value = bit 0, highest value = bit 7), in the following way:

4-bytes:

- In *i0*, bits 0-7 represent  $2^{**0}$  through  $2^{**7}$
- In *i1*, bits 0-7 represent  $2^{**8}$  through  $2^{**15}$
- In *i2*, bits 0-7 represent  $2^{**16}$  through  $2^{**23}$
- In *i3*, bits 0-6 represent  $2^{**24}$  through  $2^{**30}$

2-bytes:

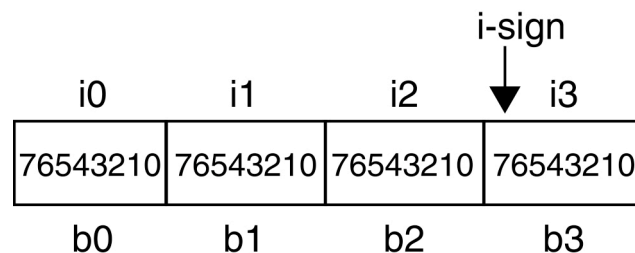
- In *i0*, bits 0-7 represent  $2^{**0}$  through  $2^{**7}$
- In *i1*, bits 0-6 represent  $2^{**8}$  through  $2^{**14}$

1-byte:

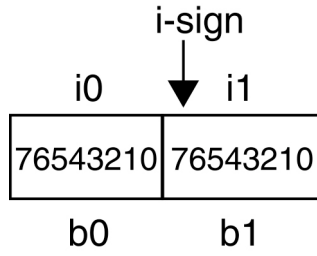
- In *i0*, bits 0-6 represent  $2^{**0}$  through  $2^{**6}$

All negative values are represented in two's complement.

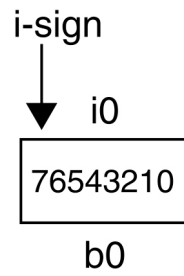
### C.3.1 LSB 4-byte Integers



### C.3.2 LSB 2-byte Integers



### C.3.3 LSB 1-byte Integers



## C.4 LSB\_UNSIGNED\_INTEGER

**Aliases:** PC\_UNSIGNED\_INTEGER  
VAX\_UNSIGNED\_INTEGER

This section describes unsigned integers stored in Least Significant Byte first (LSB) format. In this section the following definitions apply:

*b0 – b3* Arrangement of bytes as they appear when reading a file (e.g., read byte b0 first, then b1, b2 and b3)

*i0 – i3* Arrangement of bytes in the integer, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., lowest value = bit 0, highest value = bit 7), in the following way:

4-bytes:

In *i0*, bits 0-7 represent  $2^{**0}$  through  $2^{**7}$

In *i1*, bits 0-7 represent  $2^{**8}$  through  $2^{**15}$

In *i2*, bits 0-7 represent  $2^{**16}$  through  $2^{**23}$

In *i3*, bits 0-7 represent  $2^{**24}$  through  $2^{**31}$

2-bytes:

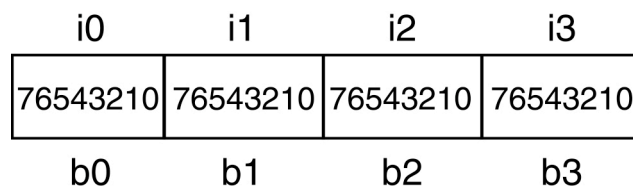
In *i0*, bits 0-7 represent  $2^{**0}$  through  $2^{**7}$

In *i1*, bits 0-7 represent  $2^{**8}$  through  $2^{**15}$

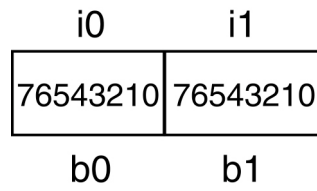
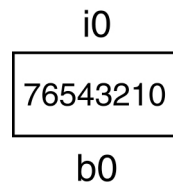
1-byte:

In *i0*, bits 0-7 represent  $2^{**0}$  through  $2^{**7}$

### C.4.1 LSB 4-byte Unsigned Integers





**C.4.2 LSB 2-byte Unsigned Integers****C.4.3 LSB 1-byte Unsigned Integers**

## C.5 IEEE\_REAL

**Aliases:** FLOAT  
 REAL  
 MAC\_REAL  
 SUN\_REAL

This section describes the internal format of IEEE-format floating-point numbers. In this section the following definitions apply:

<i>b0 – b9</i>	Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2, b3, etc.)
<i>m-sign</i>	Mantissa sign bit
<i>int-bit</i>	In 10-byte real format only, the integer part of the mantissa, assumed to be “1” in other formats, is explicitly indicated by this bit
<i>e0 – e1</i>	Arrangement of the portions of the bytes that make up the exponent, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., lowest value = rightmost bit in the exponent part of the byte, highest value = leftmost bit in the exponent part of the byte) in the following way:

10-bytes (temporary):

In e0, bits 0-7 represent  $2^{**0}$  through  $2^{**7}$

In e1, bits 0-6 represent  $2^{**8}$  through  $2^{**14}$

Exponent bias = 16383

8-bytes (double precision):

In e0, bits 4-7 represent  $2^{**0}$  through  $2^{**3}$

In e1, bits 0-6 represent  $2^{**4}$  through  $2^{**10}$

Exponent bias = 1023

4-bytes (single precision):

In e0, bit 7 represent  $2^{**0}$

In e1, bits 0-6 represent  $2^{**1}$  through  $2^{**7}$

Exponent bias = 127

$m0 - m7$  Arrangement of the portions of the bytes that make up the mantissa, from highest order fractions to the lowest order fraction. The order of the bits within each byte progresses from left to right, with each bit representing a fractional power of two, in the following way:

10-bytes (temporary):

- In  $m0$ , bits 6-0 represent  $1/2^{**1}$  through  $1/2^{**7}$
- In  $m1$ , bits 7-0 represent  $1/2^{**8}$  through  $1/2^{**15}$
- In  $m2$ , bits 7-0 represent  $1/2^{**16}$  through  $1/2^{**23}$
- In  $m3$ , bits 7-0 represent  $1/2^{**24}$  through  $1/2^{**31}$
- In  $m4$ , bits 7-0 represent  $1/2^{**32}$  through  $1/2^{**39}$
- In  $m5$ , bits 7-0 represent  $1/2^{**40}$  through  $1/2^{**47}$
- In  $m6$ , bits 7-0 represent  $1/2^{**48}$  through  $1/2^{**55}$
- In  $m7$ , bits 7-0 represent  $1/2^{**56}$  through  $1/2^{**63}$

8-bytes (double precision):

- In  $m0$ , bits 3-0 represent  $1/2^{**1}$  through  $1/2^{**4}$
- In  $m1$ , bits 7-0 represent  $1/2^{**5}$  through  $1/2^{**12}$
- In  $m2$ , bits 7-0 represent  $1/2^{**13}$  through  $1/2^{**20}$
- In  $m3$ , bits 7-0 represent  $1/2^{**21}$  through  $1/2^{**28}$
- In  $m4$ , bits 7-0 represent  $1/2^{**29}$  through  $1/2^{**36}$
- In  $m5$ , bits 7-0 represent  $1/2^{**37}$  through  $1/2^{**44}$
- In  $m6$ , bits 7-0 represent  $1/2^{**45}$  through  $1/2^{**52}$

4-bytes (single precision):

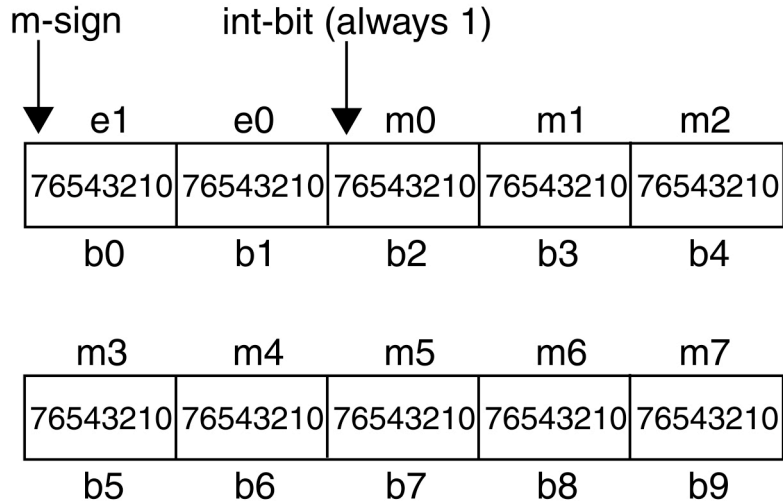
- In  $m0$ , bits 6-0 represent  $1/2^{**1}$  through  $1/2^{**7}$
- In  $m1$ , bits 7-0 represent  $1/2^{**8}$  through  $1/2^{**15}$
- In  $m2$ , bits 7-0 represent  $1/2^{**16}$  through  $1/2^{**23}$

The following representations all follow this format:

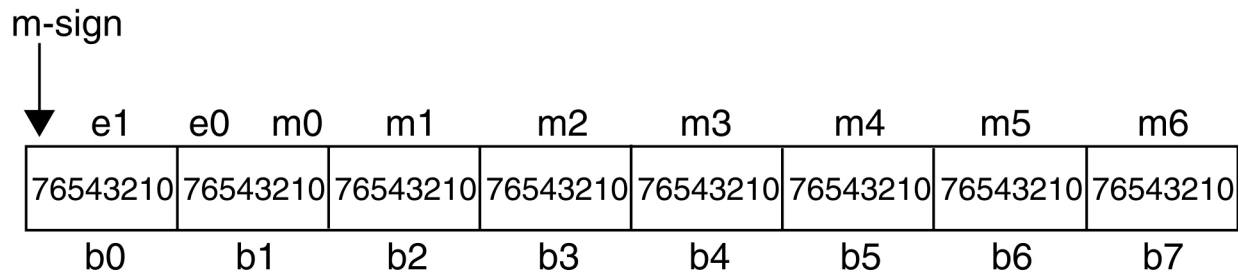
$$1.\textit{mantissa} \times 2^{**}(\textit{exponent} - \textit{bias})$$

Note that the integer part (“1.”) is implicit in all formats except the 10-byte (temporary) real format, as described above. In all cases the exponent is stored as an unsigned, biased integer (that is, the stored exponent value – bias value = true exponent).

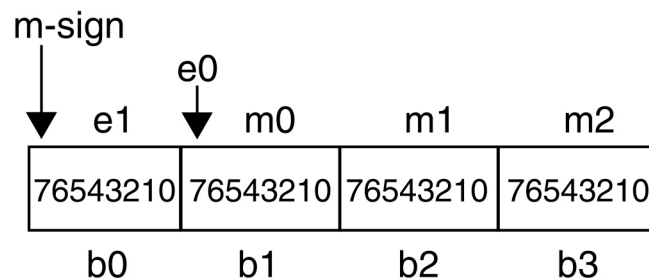
### C.5.1 IEEE 10-byte (Temporary) Real Numbers



### C.5.2 IEEE 8-byte (Double Precision) Real Numbers



### C.5.3 IEEE 4-byte (Single Precision) Real Numbers



## C.6 IEEE\_COMPLEX

**Aliases:** COMPLEX  
MAC\_COMPLEX  
SUN\_COMPLEX

IEEE complex numbers consist of two IEEE\_REAL format numbers of the same precision, contiguous in memory. The first number represents the real part and the second the imaginary part of the complex value.

For more information on using IEEE\_REAL formats, see Section C.5.

## C.7 PC\_REAL

**Aliases:** None

This section describes the internal storage format corresponding to the PC\_REAL data type. In this section the following definitions apply:

<i>b0 – b9</i>	Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2 and b3)
<i>m-sign</i>	Mantissa sign bit
<i>int-bit</i>	In 10-byte real format only, the integer part of the mantissa, assumed to be “1” in other formats, is explicitly indicated by this bit.
<i>e0 – e1</i>	Arrangement of the portions of the bytes that make up the exponent, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., lowest value = rightmost bit in the exponent part of the byte, highest value = leftmost bit in the exponent part of the byte) in the following way:

10-bytes (temporary):

In e0, bits 0-7 represent  $2^{**0}$  through  $2^{**7}$

In e1, bits 0-6 represent  $2^{**8}$  through  $2^{**14}$

Exponent bias = 16383

8-bytes (double precision):

In e0, bits 4-7 represent  $2^{**0}$  through  $2^{**3}$

In e1, bits 0-6 represent  $2^{**4}$  through  $2^{**10}$

Exponent bias = 1023

4-bytes (single precision):

In e0, bit 7 represent  $2^{**0}$

In e1, bits 0-6 represent  $2^{**1}$  through  $2^{**7}$

Exponent bias = 127

<i>m0 – m7</i>	Arrangement of the portions of the bytes that make up the mantissa, from highest order fractions to the lowest order fraction. The order of the bits within each byte progresses from left to right, with each bit representing a fractional power of two, in the following way:
----------------	--

10-bytes (temporary):

In m0, bits 6-0 represent  $1/2^{**1}$  through  $1/2^{**7}$   
 In m1, bits 7-0 represent  $1/2^{**8}$  through  $1/2^{**15}$   
 In m2, bits 7-0 represent  $1/2^{**16}$  through  $1/2^{**23}$   
 In m3, bits 7-0 represent  $1/2^{**24}$  through  $1/2^{**31}$   
 In m4, bits 7-0 represent  $1/2^{**32}$  through  $1/2^{**39}$   
 In m5, bits 7-0 represent  $1/2^{**40}$  through  $1/2^{**47}$   
 In m6, bits 7-0 represent  $1/2^{**48}$  through  $1/2^{**55}$   
 In m7, bits 7-0 represent  $1/2^{**56}$  through  $1/2^{**63}$

8-bytes (double precision):

In m0, bits 3-0 represent  $1/2^{**1}$  through  $1/2^{**4}$   
 In m1, bits 7-0 represent  $1/2^{**5}$  through  $1/2^{**12}$   
 In m2, bits 7-0 represent  $1/2^{**13}$  through  $1/2^{**20}$   
 In m3, bits 7-0 represent  $1/2^{**21}$  through  $1/2^{**28}$   
 In m4, bits 7-0 represent  $1/2^{**29}$  through  $1/2^{**36}$   
 In m5, bits 7-0 represent  $1/2^{**37}$  through  $1/2^{**44}$   
 In m6, bits 7-0 represent  $1/2^{**45}$  through  $1/2^{**52}$

4-bytes (single precision):

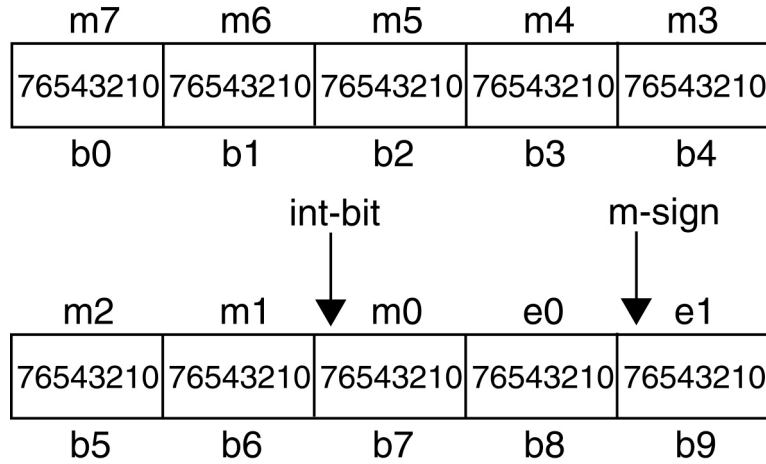
In m0, bits 6-0 represent  $1/2^{**1}$  through  $1/2^{**7}$   
 In m1, bits 7-0 represent  $1/2^{**8}$  through  $1/2^{**15}$   
 In m2, bits 7-0 represent  $1/2^{**16}$  through  $1/2^{**23}$

The following representations all follow this format:

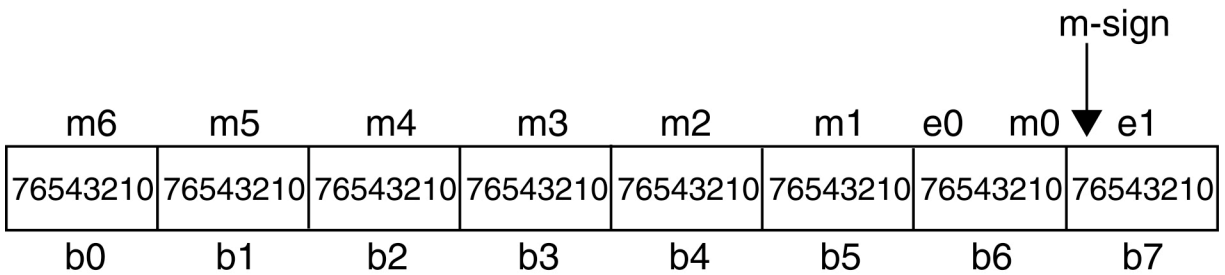
$$1.\textit{mantissa} \times 2^{**}(\textit{exponent} - \textit{bias})$$

Note that the integer part (“1.”) is implicit in all formats except the 10-byte (temporary) real format, as described above. In all cases the exponent is stored as an unsigned, biased integer (that is, the stored exponent value – bias value = true exponent).

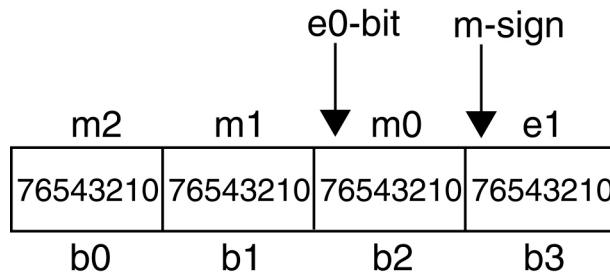
### C.7.1 PC 10-byte (Temporary) Real Numbers



### C.7.2 PC 8-byte (Double Precision) Real Numbers



### C.7.3 PC 4-byte (Single Precision) Real Numbers





## C.8 PC\_COMPLEX

**Aliases:** None

PC complex numbers consist of two PC\_REAL format numbers of the same precision, contiguous in memory. The first number represents the real part and the second the imaginary part of the complex value.

For more information on using PC\_REAL formats, see Section C.7.

## C.9 VAX\_REAL, VAXG\_REAL

**Aliases:** VAX\_DOUBLE for VAX\_REAL only.

No aliases for VAXG\_REAL

This section describes the internal format corresponding to the VAX\_REAL and VAXG\_REAL data types. In this section the following definitions apply:

<i>b0 – b15</i>	Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2 and b3)
<i>m-sign</i>	Mantissa sign bit
<i>e0 – e1</i>	Arrangement of the portions of the bytes that make up the exponent, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., lowest value = rightmost bit in the exponent part of the byte, highest value = leftmost bit in the exponent part of the byte) in the following way:

16-bytes (H-type, quad precision):

In e0, bits 0-7 represent  $2^{**0}$  through  $2^{**7}$

In e1, bits 0-6 represent  $2^{**8}$  through  $2^{**14}$

Exponent bias = 16385

8-bytes (G-type, double precision):

In e0, bits 4-7 represent  $2^{**0}$  through  $2^{**3}$

In e1, bits 0-6 represent  $2^{**4}$  through  $2^{**10}$

Exponent bias = 1025

8-bytes (D-type, double precision):

In e0, bit 7 represents  $2^{**0}$

In e1, bits 0-6 represent  $2^{**1}$  through  $2^{**7}$

Exponent bias = 129

4-bytes (F-type, single precision):

In e0, bit 7 represent  $2^{**0}$

In e1, bits 0-6 represent  $2^{**1}$  through  $2^{**7}$

Exponent bias = 129

*m0 – m13* Arrangement of the portions of the bytes that make up the mantissa, from highest order fractions to the lowest order fraction. The order of the bits within each byte progresses from left to right, with each bit representing a fractional power of two, in the following way:

16-bytes (H-type, quad precision):

- In m0, bits 7-0 represent  $1/2^{**1}$  through  $1/2^{**8}$
- In m1, bits 7-0 represent  $1/2^{**9}$  through  $1/2^{**16}$
- In m2, bits 7-0 represent  $1/2^{**17}$  through  $1/2^{**24}$
- In m3, bits 7-0 represent  $1/2^{**25}$  through  $1/2^{**32}$
- In m4, bits 7-0 represent  $1/2^{**33}$  through  $1/2^{**40}$
- In m5, bits 7-0 represent  $1/2^{**41}$  through  $1/2^{**48}$
- In m6, bits 7-0 represent  $1/2^{**49}$  through  $1/2^{**56}$
- In m7, bits 7-0 represent  $1/2^{**57}$  through  $1/2^{**64}$
- In m8, bits 7-0 represent  $1/2^{**65}$  through  $1/2^{**72}$
- In m9, bits 7-0 represent  $1/2^{**73}$  through  $1/2^{**80}$
- In m10, bits 7-0 represent  $1/2^{**81}$  through  $1/2^{**88}$
- In m11, bits 7-0 represent  $1/2^{**89}$  through  $1/2^{**96}$
- In m12, bits 7-0 represent  $1/2^{**97}$  through  $1/2^{**104}$
- In m13, bits 7-0 represent  $1/2^{**105}$  through  $1/2^{**112}$

8-bytes (G-type, double precision):

- In m0, bits 3-0 represent  $1/2^{**1}$  through  $1/2^{**4}$
- In m1, bits 7-0 represent  $1/2^{**5}$  through  $1/2^{**12}$
- In m2, bits 7-0 represent  $1/2^{**13}$  through  $1/2^{**20}$
- In m3, bits 7-0 represent  $1/2^{**21}$  through  $1/2^{**28}$
- In m4, bits 7-0 represent  $1/2^{**29}$  through  $1/2^{**36}$
- In m5, bits 7-0 represent  $1/2^{**37}$  through  $1/2^{**44}$
- In m6, bits 7-0 represent  $1/2^{**45}$  through  $1/2^{**52}$

8-bytes (D-type, double precision):

- In m0, bits 6-0 represent  $1/2^{**1}$  through  $1/2^{**7}$
- In m1, bits 7-0 represent  $1/2^{**8}$  through  $1/2^{**15}$
- In m2, bits 7-0 represent  $1/2^{**16}$  through  $1/2^{**23}$
- In m3, bits 7-0 represent  $1/2^{**24}$  through  $1/2^{**31}$
- In m4, bits 7-0 represent  $1/2^{**32}$  through  $1/2^{**39}$
- In m5, bits 7-0 represent  $1/2^{**40}$  through  $1/2^{**47}$
- In m6, bits 7-0 represent  $1/2^{**48}$  through  $1/2^{**55}$

4-bytes (F-type, single precision):

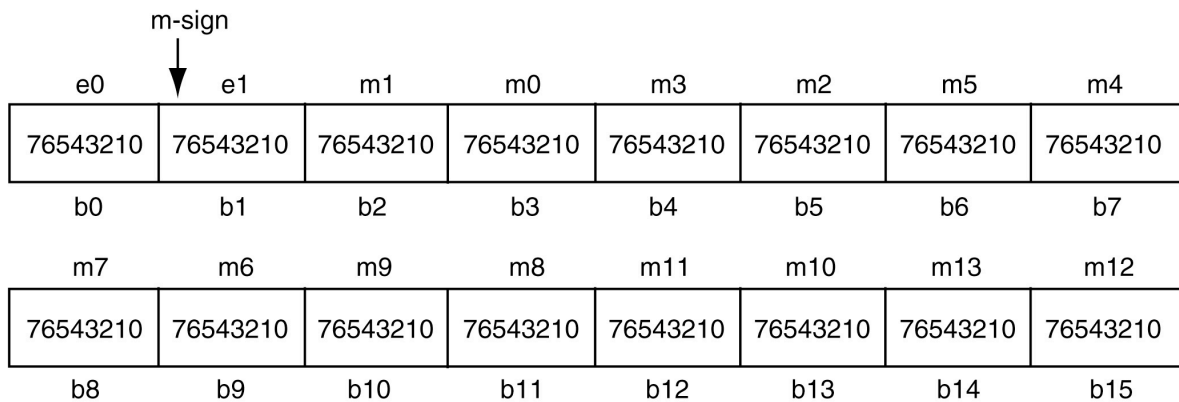
- In m0, bits 6-0 represent  $1/2^{**1}$  through  $1/2^{**7}$
- In m1, bits 7-0 represent  $1/2^{**8}$  through  $1/2^{**15}$
- In m2, bits 7-0 represent  $1/2^{**16}$  through  $1/2^{**23}$

The following representations all follow this format:

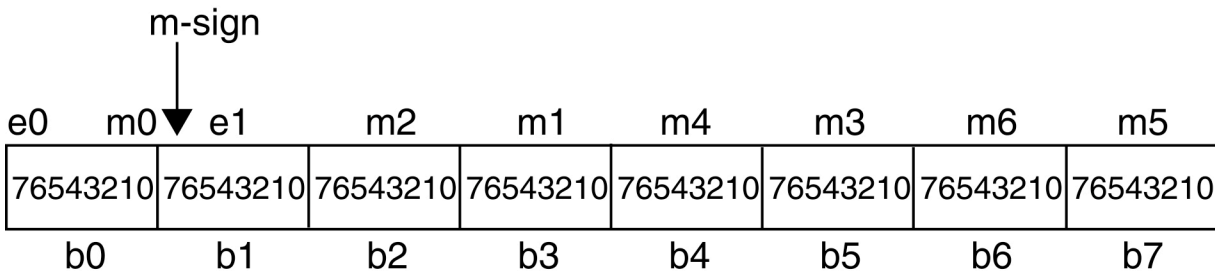
$$1.\textit{mantissa} \times 2^{*(\textit{exponent} - \textit{bias})}$$

Note that the integer part (“1.”) is implicit in all formats except the 10-byte (temporary) real format, as described above. In all cases the exponent is stored as an unsigned, biased integer (that is, the stored exponent value – bias value = true exponent).

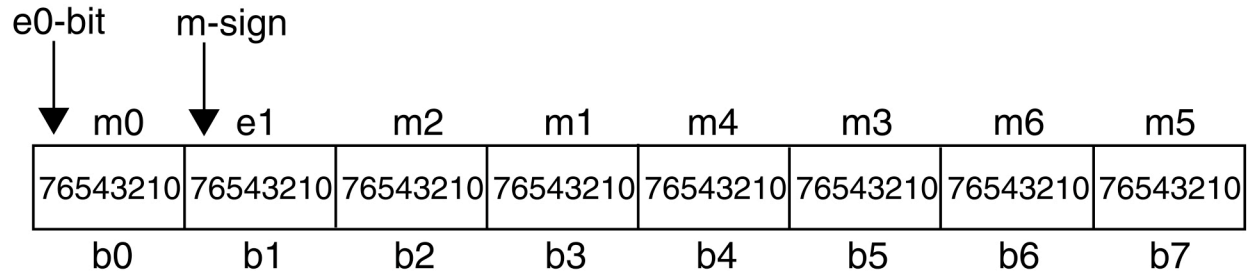
### C.9.1 VAX 16-byte H-type (Quad Precision) Real Numbers



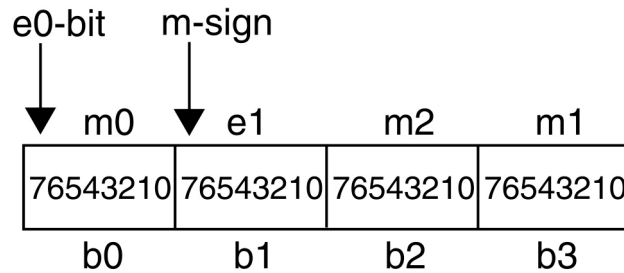
### C.9.2 VAX 8-byte G-type (Double Precision) Real Numbers



### C.9.3 VAX 8-byte D-type (Double Precision) Real Numbers



### C.9.4 VAX 4-byte F-type (Single Precision) Real Numbers



## C.10 VAX\_COMPLEX, VAXG\_COMPLEX

**Aliases:** None

VAX complex numbers consist of two VAX\_REAL (or VAXG\_REAL) format numbers of the same precision, contiguous in memory. The first number represents the real part and the second the imaginary part of the complex value.

For more information on using VAX\_REAL formats, see Section C.9.

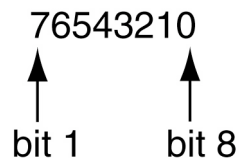
## C.11 MSB\_BIT\_STRING

**Aliases:** None

This section describes the storage format for bit strings stored in Most Significant Byte first (MSB) format. In this section the following definitions apply:

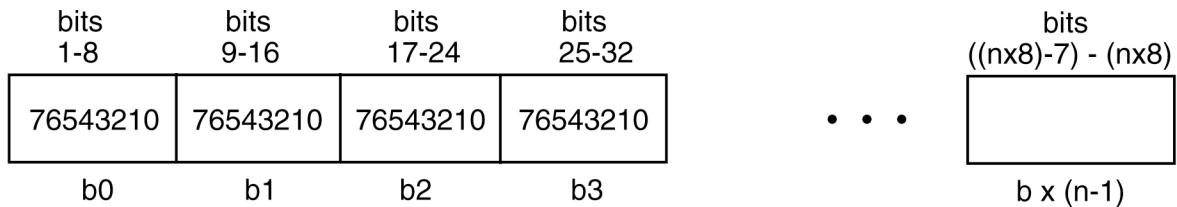
$b0 - b3$  Arrangement of bytes as they appear when read from a file (e.g., read  $b0$  first, then  $b1$ ,  $b2$  and  $b3$ )

The bits within a byte are numbered from left to right, as shown below:

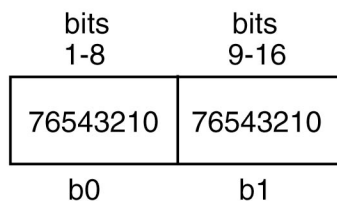


Note that in the case of MSB bit strings, no byte-swapping is required. That is, the physical storage order of the bytes is identical to the logical order.

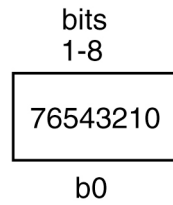
### C.11.1 MSB $n$ -byte Bit Strings



### C.11.2 MSB 2-byte Bit String



### C.11.3 MSB 1-byte Bit String





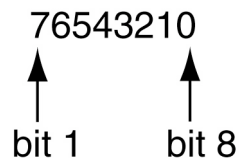
## C.12 LSB\_BIT\_STRING

**Aliases:** VAX\_BIT\_STRING

This section describes the structure of bit strings stored in Least Significant Byte first (LSB) order. In this section, the following definitions apply:

*b0 – b3* Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2 and b3)

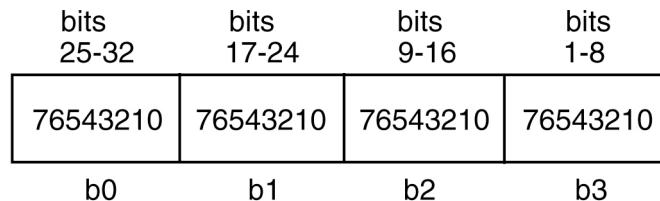
The bits within a byte are numbered from left to right, as shown below:



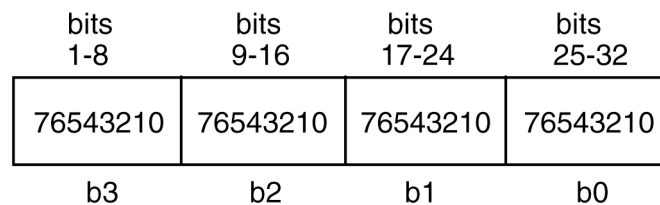
Note that for LSB bit strings byte-swapping is required to convert the storage order of bytes to the logical order.

### C.12.1 LSB 4-byte Bit String

Physical order (as read from the file):

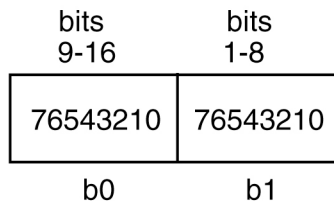


Logical order (after byte-swapping):

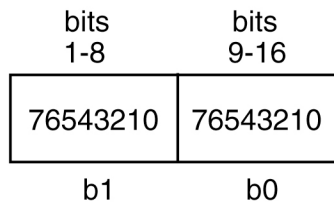


### C.12.2 LSB 2-byte Bit String

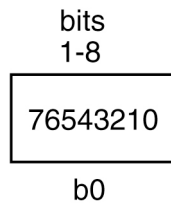
Physical order (as read from the file):



Logical order (after byte-swapping):



### C.12.3 LSB 1-byte Bit String



Note that in this degenerate case no byte-swapping is required.