Chapter 3. DATA_TYPE Values and Data File Storage Formats

Each PDS archived product is described using label objects that provide information about the data types of stored values. The data elements DATA_TYPE, BIT_DATA_TYPE, and SAMPLE_TYPE appear together with related elements defining starting location and length for each field. In PDS data object definitions the byte, bit, and record positions are counted from left to right, or first to last encountered, and always begin with 1.

Data files may be in ASCII or binary format. ASCII format is often more easily transferred between hardware systems or even application programs on the same computer. Notwithstanding, numeric data are often stored in binary files when the ASCII representation would require substantially more storage space. (For example, each 8-bit signed pixel value in a binary image file would require a four-byte field if stored as an ASCII table.)

3.1 Data Elements

Table 3.1 identifies by object the data elements providing type, location, and length information. The elements ITEMS and ITEM_BYTES are used to subdivide a single COLUMN, FIELD, BIT_COLUMN, or HISTOGRAM into a regular vector containing as many elements as specified for the value of ITEMS. In these objects the DATA_TYPE must indicate the type of a single item in the vector. In the past, the data element ITEM_TYPE was used for this purpose, but DATA_TYPE is now the preferred parameter.

3.2 Data Types

Table 3.2 identifies the valid values for the DATA_TYPE, BIT_DATA_TYPE, and SAMPLE_TYPE data elements used in PDS data object definitions. The values for these elements must be one of the standard values listed in the *Planetary Science Data Dictionary* (PSDD). Please note:

- In all cases, these standard values refer to the physical storage format of the data in the data file.
- In some cases, obsolete values from previous versions of the PDS Standards have been retained as aliases for more specific values (the type “INTEGER”, for example, is interpreted as “MSB_INTEGER” when it is encountered). In these cases the more specific value should always be used in new data sets – the obsolete value is retained only for backward compatibility. Obsolete values are indicated in the table.
- Aliases have been supplied for some of the generic data types that indicate the kind of system on which the data originated. For example, “MAC_REAL” is an alias for “IEEE_REAL”, but “VAX_REAL” has no alias, as the VAX binary storage format is unique to VAX systems. In general, the more generic term is preferred, but the system-specific version may be used if needed.

Table 3.1: Type Elements Used in Data Label Objects

Data Object	Data Elements	Notes
COLUMN	DATA_TYPE	
(without ITEMS)	START_BYTE	
	BYTES	
COLUMN	DATA_TYPE	alias for ITEM_TYPE
(with ITEMS)	START_BYTE	
	BYTES *(optional)*	total bytes in COLUMN
	ITEMS	
	ITEM_BYTES	bytes in each ITEM
BIT_COLUMN	BIT_DATA_TYPE	
(without ITEMS)	START_BIT	
	BITS	
BIT_COLUMN	START_BIT	
(with ITEMS)	BITS (*optional*)	total bits in BIT_COLUMN
	ITEMS	
	ITEM_BITS	bits in each ITEM
FIELD	DATA_TYPE	if populated
(no items)	FIELD_NUMBER	
	BYTES	maximum FIELD bytes
FIELD	DATA_TYPE	if populated
(with items)	FIELD_NUMBER	
	BYTES	maximum bytes in FIELD
	ITEMS	
	ITEM_BYTES	maximum item bytes
IMAGE	SAMPLE_TYPE	
	SAMPLE_BITS	
HISTOGRAM	DATA_TYPE	alias for ITEM_TYPE
	BYTES (*optional*)	total bytes in HISTOGRAM
	ITEMS	number of bins in HISTOGRAM
	ITEM_BYTES	bytes in each ITEM

Table 3.2: Standard PDS Data Types

Data Element Usage Codes:

D = DATA_TYPE
B = BIT_DATA_TYPE
S = SAMPLE_TYPE

Usage	Value	Description
D	ASCII_REAL	ASCII character string representing a real number; see Section 5.4 for formatting rules
D	ASCII_INTEGER	ASCII character string representing an integer; see Section 5.4 for formatting rules
D	ASCII_COMPLEX	ASCII character string representing a complex number; see Section 5.4 for formatting rules
Obsolete	BIT_STRING	alias for MSB_BIT_STRING
D, B	BOOLEAN	True/False Indicator: a 1-, 2- or 4-byte integer or 1-32 bit number. All 0 = False; anything else = True.
D	CHARACTER	ASCII character string; see Section 5.4 for formatting rules
Obsolete	COMPLEX	alias for IEEE_COMPLEX
D	DATE	ASCII character string representing a date in PDS standard format; see Section 5.4 for formatting rules
D	EBCDIC_CHARACTER	EBCDIC character string
Obsolete	FLOAT	alias for IEEE_REAL
D	IBM_COMPLEX	IBM 360/370 mainframe complex number (8- or 16-byte)
D, S	IBM_INTEGER	IBM 360/370 mainframe 1-, 2-, and 4-byte signed integers
D, S	IBM_REAL	IBM 360/370 mainframe real number (4- or 8-byte)
D, B, S	IBM_UNSIGNED_INTEGER	IBM 360/370 mainframe 1-, 2-, and 4-byte unsigned integers
D	IEEE_COMPLEX	8-, 16-, and 20-byte complex numbers
D, S	IEEE_REAL	4-, 8- and 10-byte real numbers
Obsolete	INTEGER	alias for MSB_INTEGER
D	LSB_BIT_STRING	1-, 2-, and 4-byte bit strings
D, S	LSB_INTEGER	1-, 2-, and 4-byte signed integers
D, B, S	LSB_UNSIGNED_INTEGER	1-, 2-, and 4-byte unsigned integers
D	MAC_COMPLEX	alias for IEEE_COMPLEX
D, S	MAC_INTEGER	alias for MSB_INTEGER
D, S	MAC_REAL	alias for IEEE_REAL
D, B, S	MAC_UNSIGNED_INTEGER	alias for MSB_UNSIGNED_INTEGER
D	MSB_BIT_STRING	1-, 2-, and 4-byte bit strings
D, S	MSB_INTEGER	1-, 2-, and 4-byte signed integers
D, B, S	MSB_UNSIGNED_INTEGER	1-, 2-, and 4-byte unsigned integers
D, B	N/A	Used only for spare (or unused) fields included in the data file.

D	PC_COMPLEX	8-, 16-, and 20-byte complex numbers in IBM/PC format
D, S	PC_INTEGER	alias for LSB_INTEGER
D, S	PC_REAL	4-, 8-, and 10-byte real numbers in IBM/PC format
D, B, S	PC_UNSIGNED_INTEGER	alias for LSB_UNSIGNED_INTEGER
Obsolete	REAL	alias for IEEE_REAL
D	SUN_COMPLEX	alias for IEEE_COMPLEX
D, S	SUN_INTEGER	alias for MSB_INTEGER
D, S	SUN_REAL	alias for IEEE_REAL
D, B, S	SUN_UNSIGNED_INTEGER	alias for MSB_UNSIGNED_INTEGER
D	TIME	ASCII character string representing a date/time in PDS standard format; see Section 5.4 for formatting rules
Obsolete	UNSIGNED_INTEGER	alias for MSB_UNSIGNED_INTEGER
D	VAX_BIT_STRING	alias for LSB_BIT_STRING
D	VAX_COMPLEX	Vax F-, D-, and H-type (8-, 16- and 32-byte, respectively) complex numbers
D, S	VAX_DOUBLE	alias for VAX_REAL
D, S	VAX_INTEGER	alias for LSB_INTEGER
D, S	VAX_REAL	Vax F-, D-, and H-type (4-, 8- and 16-byte, respectively) real numbers
D, B, S	VAX_UNSIGNED_INTEGER	alias for LSB_UNSIGNED_INTEGER
D	VAXG_COMPLEX	Vax G-type (16-byte) complex numbers
D, S	VAXG_REAL	Vax G-type (8-byte) real numbers

3.3 Binary Integers

There are two widely used formats for integer representations in 16-bit and 32-bit binary fields: most significant byte first (MSB) and least significant byte first (LSB) architectures. The MSB architectures include IBM mainframes, many UNIX systems such as SUN, and Macintosh computers. The LSB architectures include VAX systems and IBM PCs. In the original PDS system the default format was MSB, thus the designation of “INTEGER” and “UNSIGNED_INTEGER” as aliases of “MSB_INTEGER” and “MSB_UNSIGNED_INTEGER”. New data sets should be prepared using the appropriate specific designation from Table 3.2, above.

3.4 Signed vs. Unsigned Integers

The “_INTEGER” data types refer to signed, 2’s complement integers. Use the corresponding “_UNSIGNED_INTEGER” type for unsigned integer and bit string fields.

3.5 Floating Point Formats

The PDS default representation for floating point numbers is the ANSI/IEEE standard. This representation is defined as the IEEE_REAL data type, with aliases identified in Table 3.2. Several additional specific floating-point representations supported by PDS are described in Appendix C.

3.6 Bit String Data

The BIT_STRING data types are used in definitions of table columns holding individual bit field values. A BIT_COLUMN object defines each bit field. BIT_STRING data types can be 1-, 2-, or 4-byte fields, much like a binary integer. Extraction of specific bit fields within a 2- or 4-byte BIT_STRING is dependent on the host architecture (MSB or LSB). In interpreting bit fields (BIT_COLUMNS) within a BIT_STRING, any necessary conversions such as byte swapping from LSB to MSB are done first, then bit field values (START_BIT, BITS) are used to extract the appropriate bits. This procedure ensures that bit fields are not fragmented due to differences in hardware architectures.

3.7 Character Data

Specification of character field format in ASCII and binary files pending.

3.8 Format Specifications

Data format specifications provided in the FORMAT element serve two purposes:

1. In an ASCII TABLE data file or SPREADSHEET file, they provide a format which can be used in scanning the ASCII record for individual fields; and
2. In a binary data file, they provide a format that can be used to display the data values.

A subset of the FORTRAN data format specifiers is used for the values of FORMAT elements. Valid specifiers include:

Aw	Character data value
Iw	Integer value
Fw.d	Floating point value, displayed in decimal format
Ew.d[Ee]	Floating point value, displayed in exponential format

Where:

- *w* is the total number of positions in the output field (including sign, decimal point, and exponentiation character – usually “E” – if any);
- *d* is the number of positions to the right of the decimal point;
- *e* is the number of positions in exponent length field.

3.9 Internal Representations of Data Types

Appendix C contains the detailed internal representations of the PDS standard data types listed in Table 3.2.

The PDS has developed tools designed to use the specifications contained in Appendix C for interpreting data values for display and validation.