# Data File Standard for Flow Cytometry

# Version FCS 3.1
# Normative Reference


- Josef Spidlen, Terry Fox Laboratory, BC Cancer Agency, Vancouver, BC, Canada
- Wayne Moore, Genetics Department, Stanford University School of Medicine, Stanford, CA, USA
- David Parks, Stanford Shared FACS Facility, Stanford University, Stanford, CA, USA
- Michael Goldberg, Becton Dickinson Biosciences, San Jose, CA, USA
- Chris Bray, Verity Software House, Topsham, ME, USA
- Pierre Bierre, Cytek Development, Fremont, CA, USA
- Peter Gorombey, Soft Flow Informatika, Burnsville, MN, USA
- Bill Hyun, Laboratory for Cell Analysis, Helen Diller Family Comprehensive Cancer Center, University of California, San Francisco, CA, USA
- Mark Hubbard, iCyt, Champaign, IL, USA
- Simon Lange, Partec GmBH, Görlitz, Germany
- Ray Lefebvre, Guava Technologies, Hayward, CA, USA
- Robert Leif, Newport Instruments, San Diego, CA, USA
- David Novo, De Novo Software, Los Angeles, CA, USA
- Leo Ostruszka, Accuri Cytometers, Ann Arbor, MI, USA
- Adam Treister, Treestar, Ltd., Ashland, OR, USA
- James Wood, Department of Cancer Biology, Wake Forest University School of Medicine, Winston-Salem, NC, USA
- Robert F. Murphy, Carnegie Mellon University, Pittsburgh, PA, USA
- Mario Roederer, National Institutes of Health, Bethesda, MD, USA
- Damir Sudar, Lawrence Berkeley Laboratory, Berkeley, CA, USA
- Robert Zigon, Beckman Coulter, Chaska, MN, USA
- Ryan R. Brinkman, Terry Fox Laboratory, BC Cancer Agency, Vancouver, BC, Canada


International Society for Advancement of Cytometry (ISAC)
Data Standards Task Force (DSTF)

**Document Status**

In 1984, the first Flow Cytometry Standard format for data files was adopted as FCS 1.0. This standard was modified in 1990 as FCS 2.0 and again in 1997 as FCS 3.0. This document is the normative specification of FCS 3.1

FCS 3.1 has undergone an extensive revision process including several months of the ISAC wide-membership commentary period and final approval by the ISAC Data Standards Task Force. At this point, ISAC is satisfied that the FCS 3.1 specification meets well the business needs of all involved parties and it is being released as an ISAC Recommendation.

**Patent Disclaimer**

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. ISAC shall not be responsible for identifying patents or patent applications for which a license may be required to implement an ISAC standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

**Acknowledgment**

This work was supported by the Michael Smith Foundation for Health Research and by NIH grant 1R01EB005034 from the National Institute of Biomedical Imaging and Bioengineering to RRB. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institute of Biomedical Imaging and Bioengineering or the National Institutes of Health.

# Table of Contents

# 1 General

## 1.1 Scope

This is version 3.1 of the Flow Cytometry Data File Standard (FCS 3.1). Previous versions of this specification can be found in references (1), (2), and (3). Its purpose is to provide detailed specifications for the structure of the data sets produced as a result of acquiring data on a cytometer and writing the data to a file.

## 1.2 Conformance

To be conformant with FCS 3.1, a data file must conform to the file structure as described in this document and must contain all required keyword-value pairs in the primary TEXT segment of the file. A conformant file must not contain other segments not described in the data set HEADER segment. To be conformant with FCS 3.1 an analysis program must be able to correctly read and interpret all of the data contained in any minimum FCS 3.1 conformant file (a minimum FCS 3.1 conformant file is one with only the required keyword-value pairs in the TEXT segment of the file and no information in the ANALYSIS segment).

# 2 Terminology and General Requirements

## 2.1 Conventions

### 2.1.1

The ASCII character code is used for all keywords throughout an FCS 3.1 file. Only printable characters from the range 32-126 (20-7E hex) encoded as a single byte with the high order bit set to 0 (no parity bit) may be part of a keyword.

### 2.1.2

The UTF-8 (4) character code is used for all keyword values throughout an FCS 3.1 file. UTF-8 stays for the Universal Character Set (UCS) as defined in ISO 10646 at implementation level 3 in UTF-8 encoding defined in ISO 10646-1:2000.

### 2.1.3

Numerical values are base 10 unless otherwise specified.

## 2.2 Definitions

### 2.2.1

An FCS 3.1 data file consists of one or more data sets. However, the usage of multiple data sets within a single data file is deprecated unless the multiple data sets are derived one from another.

### 2.2.2

A data set is defined as the collection of information produced by a cytometer as it carries out its measurements on some number of particles.

## 2.2.3

The collection of information in a data set is divided into at least four segments including a HEADER segment, a primary TEXT segment, a DATA segment, and an ANALYSIS segment. The ANALYSIS segment may be empty and any number of implementor-defined segments may follow the first four segments.

## 2.2.4

The HEADER segment identifies the data set as FCS 3.1 and contains ASCII byte offsets from the start of the data set to the beginning and end of each of the other segments. The first byte of a segment is understood as the beginning of a segment. The last byte of a segment is understood as the end of a segment.

## 2.2.5

A keyword is the label of a data field. A keyword-value pair is the label of the data field with its associated value. Keywords are unique in data sets, i.e., there are no multiple instances of the same keyword in the data set.

## 2.2.6

The TEXT segments contain a series of ASCII keyword-value pairs that describe the format of the DATA segment and most of the experimental operating conditions. The primary TEXT segment contains all required keyword-value pairs. The supplemental TEXT segment may contain optional keyword-value pairs only.

## 2.2.7

The DATA segment contains either a list of the events or histograms of the data. Note that histograms are deprecated in FCS 3.1.

## 2.2.8

An event is an ordered list of the cytometric measurements for one particle. The length of an event is the number of parameters involved in the measurement.

## 2.2.9

A parameter is a (typically numeric) characteristic of the signal produced by one of the detectors of the cytometer or some other characteristic related to an event. The height of the forward scattering signal (FSC-H) is typically one of the measurement parameters. Parameters may also include Time and/or calculated parameters (e.g., FL2-A/FL2-H ratio, event classification, etc.). A parameter value is a digital representation of a parameter.

## 2.2.10

Each data set in a data file contains all the information needed to read and interpret the data set.

## 2.2.11

All space within a file that is not contained in: 1) the HEADER, 2) a segment specified in the HEADER or, 3) the CRC, must be filled with a space character (ASCII 32). This includes unused space between the HEADER and the first segment, between the end of one segment and the beginning of the next segment and between the end of the CRC (after the last segment) and the end of the file (if any).

### 2.2.12

List mode data storage means that events are stored one after the other in a list. A list mode data set may be considered as a linear array of vectors, each vector corresponding to an event and vector components corresponding to types of measurements, such as forward scatter, side scatter, etc.

### 2.2.13

All byte offsets are referenced to the beginning of the data set. The first data set in a file begins at byte zero of the file.

### 2.2.14

The implementor is the entity that creates the software to read and write FCS conformant data files.

### 2.2.15

The delimiter is the first character of the primary TEXT segment and is subsequently placed in the primary TEXT, supplemental TEXT and ANALYSIS segments to separate keywords from keyword values. The delimiter may be any ASCII character from the range 1-126 (01-7E hex) encoded as a single byte with the high order bit set to 0 (no parity bit). Within this document, we use "/" (ASCII 47) as delimiter character in provided examples.

## 2.3 General Concepts

An FCS 3.1 file is composed of one or more data sets, each containing at a minimum HEADER, TEXT and DATA segments. The HEADER, TEXT, and ANALYSIS segments contain ASCII-encoded text readable by a text editor (keyword values may contain UTF-8 encoded characters). The DATA segment contains flow cytometry data stored in list mode (preferred) or as histograms (deprecated).

Please note that the usage of multiple data sets within a single data file is deprecated unless the multiple data sets are derived one from another. Technically, it is still possible to store multiple data sets within an FCS file but implementors are being discouraged to do so unless one data set is derived from the other one within the data file.

# 3 File Segments

## 3.1 HEADER Segment

### 3.1.1

The primary purpose of the HEADER segment is to describe the location of the other segments in the data set. The HEADER segment begins at byte offset zero from the beginning of the data set. The first six bytes in the HEADER segment comprise the version identifier (FCS3.1). Note that there is no space character between the FCS and the 3.1 in the identifier. The next 4 bytes (6 - 9) are occupied by space characters (ASCII 32). Following the identifier are at least three pairs of ASCII-encoded integers indicating the byte offsets for the start and end (=last byte of) of the primary TEXT segment, the DATA segment, and the ANALYSIS segment, respectively. The byte offsets are referenced to the beginning of the data set. Under FCS 3.1 these offsets remain limited to 8 bytes. Each ASCII encoded integer offset is right justified in its 8 byte space. The first byte offset (bytes 10 - 17) is that to the start of the primary TEXT segment. The next byte offset (bytes 18 - 25) is that for the last byte of the primary TEXT segment. The next offset (bytes 26 - 33) is that for the start of the DATA segment. The byte offset for the last byte of the DATA segment occupies bytes 34 - 41. That for the start of the ANALYSIS segment occupies bytes 42 -

49. The byte offset for the last byte of the ANALYSIS segment is in bytes 50 - 57. If there is no ANALYSIS segment these last two byte offsets can be set to zero (right justified) or left blank (filled with space characters). Offsets to the start and end (last byte of) of user-defined OTHER segments of the data set follow the ANALYSIS segment offsets. The user-defined segments will not be interpretable by others unless appropriate information is passed on by the data set originator.

FCS 3.1 maintains support introduced in FCS 3.0 for data sets larger than 99,999,999 bytes. When any portion of a segment falls outside the 99,999,999 byte limit, '0's are substituted in the HEADER for that segments begin and end byte offset. The byte offsets for begin DATA, end DATA, begin ANALYSIS, end ANALYSIS (begin and end supplemental TEXT if appropriate) will then only be found as keyword-value pairs in the primary TEXT segment. Note, when a segment is contained completely within the first 99,999,999 bytes of a data set, the byte offsets for that segment will be duplicated in the TEXT segment as keyword values. Note also, if the ANALYSIS offsets in the HEADER are zero, the $BEGINANALYSIS and $ENDANALYSIS keywords must be checked to determine if an ANALYSIS segment is present.

**Table 1 – Contents of HEADER fields and the byte offsets to the beginning and end of each field. Each offset is right justified in its field.**

| Contents | Start byte position | End byte position |
|---|---|---|
| FCS3.1 | 00 | 05 |
| ASCII(32) - space characters | 06 | 09 |
| ASCII-encoded offset to first byte of TEXT segment | 10 | 17 |
| ASCII-encoded offset to last byte of TEXT segment | 18 | 25 |
| ASCII-encoded offset to first byte of DATA segment | 26 | 33 |
| ASCII-encoded offset to last byte of DATA segment | 34 | 41 |
| ASCII-encoded offset to first byte of ANALYSIS segment | 42 | 49 |
| ASCII-encoded offset to last byte of ANALYSIS segment | 50 | 57 |
| ASCII-encoded offset to user defined OTHER segments | 58 | beginning of next segment |

**Example:**

One example HEADER segment is as follows:

```
FCS3.1*********256****1545****1792**202455*******0*******0
```

The '*' character is used to represent a space character here. The TEXT segment starts at byte 256 from the location of the 'F' in FCS3.1 and ends at byte offset 1545; the length of the TEXT segment is 1290 bytes (i.e., 1545 + 1 - 256). The DATA segment starts at byte offset 1792 and ends at 202455; the length of the DATA segment is 200664 (i.e., 202455 + 1 - 1792). There is no ANALYSIS segment, so the start and end offsets are shown as zeros. They could be left blank. Note that the HEADER segment is a continuous byte stream with no return or line feed characters. The bytes between the end of the HEADER segment and the start of the next segment must be filled with the space character. In this example, the segments are in the order HEADER, TEXT, DATA, and ANALYSIS. The FCS standard requires only that the HEADER

segment be at the start of the data set and the primary TEXT segment be located entirely within the first 99,999,999 bytes.

**Example:**

A second example of a legal HEADER segment is:

```
FCS3.1*********256****1545*******0*******0*******0*******0
```

The '0's in the begin DATA and end DATA positions indicates that the DATA segment exceeds the 99,999,999 byte limit. Therefore, the byte offsets to begin Data and end Data, are located only in the $BEGINDATA, $ENDDATA keyword values in the TEXT segment. The begin ANALYSIS and end ANALYSIS byte offsets are also located in the $BEGINANALYSIS and $ENDANALYSIS keyword values in TEXT segment, if an ANALYSIS segment exist.

**Example:**

A third example of a legal HEADER segment is:

```
FCS3.1******202451**203140****1792**202450*******0*******0
```

This HEADER is different from the other examples in that it describes a data set in which the primary TEXT segment follows the DATA segment.

## 3.2 TEXT Segment

### 3.2.1

The TEXT segments (primary and supplemental) contain a series of ASCII encoded keyword-value pairs that describe various aspects of the data set. For example, $TOT/5000/ is a keyword-value pair indicating that the total number of events in the file is 5000. $TOT is the keyword and 5000 is the value (the value may be UTF-8 encoded). The '$' character flags this keyword as a standard FCS keyword.

### 3.2.2

A data set must contain a primary TEXT segment which contains all required keyword-value pairs and any number of optional keyword-value pairs. The primary TEXT segment must be contained entirely in the first 99,999,999 bytes of data set.

### 3.2.3

A data set may contain an optional supplemental TEXT segment that can contain only optional keyword-value pairs and may be placed anywhere in a data set after the HEADER segment.

### 3.2.4

The byte offset to the beginning and end of the supplemental TEXT segment is found in the $BEGINSTEXT and $ENDSTEXT keyword-value pairs which must be located in the primary TEXT segment.

### 3.2.5

The first character in the primary TEXT segment is the ASCII delimiter character. This character must also be used as the delimiter in the ANALYSIS and supplemental TEXT segments.

### 3.2.6

The delimiter is placed at the start and end of a keyword value.

### 3.2.7

The delimiter may not be the first character in a keyword or keyword value. If the delimiter appears in a keyword or keyword value, it must be immediately followed by a second delimiter. For example, "$SYS/RSX-11//M/" shows a value of RSX-11/M for the keyword $SYS. Since null (zero length) keywords or keyword values are not permitted, two consecutive delimiters can never occur between a value and a keyword.

### 3.2.8

All keywords are encoded in ASCII - printable values from the range 32-126 (20-7E hex) encoded as a single byte with the high order bit set to 0. All keyword values are encoded in UTF-8. UTF-8 is backward compatible with ASCII since all characters 00 through 7F (hex) inclusive are encoded the same way in UTF-8 and ASCII. The $UNICODE keyword is discontinued as a valid FCS keyword.

### 3.2.9

Keywords and keyword values must have lengths greater than zero.

### 3.2.10

Keywords are case insensitive, they may be written in a file in lower case, upper case, or a mixture of the two. However, an FCS file reader must ignore keyword case. A keyword value may be in lower case, upper case or a mixture of the two. Keyword values are case sensitive.

### 3.2.11

There are no default values for any keywords.

### 3.2.12

FCS-defined keywords must begin with the '$' character. Only FCS-defined keywords may begin with the '$' character.

### 3.2.13

FCS-defined keywords may not be redefined by the implementor.

### 3.2.14

There are required and optional FCS keyword-value pairs. The required keyword-value pairs represent the minimum set needed to successfully read and write an FCS data set. Conformant FCS file reading programs must recognize required FCS keywords.

### 3.2.15

The TEXT segments must not contain return (ASCII 13), line feed (ASCII 10) or other unprintable characters (ASCII 0-31 and 127) unless they are within a keyword value or are used as the delimiter character.

### 3.2.16

The parameter description keywords (e.g. $PnR, $PnB, etc) are numbered consecutively in the order in which the parameters are written to the file, beginning with number 1. The required and optional FCS keywords are listed below with one line descriptions. The keywords and their values are described in alphabetical order following the lists. Required keywords are so indicated.

## 3.2.17

Values of numerical keywords (e.g., values of $BEGINANALYSIS, $BEGINDATA, $BEGINSTEXT, $ENDANALYSIS, $ENDDATA, $ENDSTEXT, $NEXTDATA, $PAR, $PnB, $PnR, $TOT) shall not be padded with any non-numerical characters (including spaces and other white characters). However, leading zeros '0' (ASCII 48) may be used. For example, you may use $BEGINDATA/0000012345/ and you may not use $BEGINDATA/ 12345/.

## 3.2.18

The required FCS primary TEXT segment keywords are as follows:

| | |
|---|---|
| **$BEGINANALYSIS** | Byte-offset to the beginning of the ANALYSIS segment. |
| **$BEGINDATA** | Byte-offset to the beginning of the DATA segment. |
| **$BEGINSTEXT** | Byte-offset to the beginning of a supplemental TEXT segment. |
| **$BYTEORD** | Byte order for data acquisition computer. |
| **$DATATYPE** | Type of data in DATA segment (ASCII, integer, floating point). |
| **$ENDANALYSIS** | Byte-offset to the last byte of the ANALYSIS segment. |
| **$ENDDATA** | Byte-offset to the last byte of the DATA segment. |
| **$ENDSTEXT** | Byte-offset to the last byte of a supplemental TEXT segment. |
| **$MODE** | Data mode (list mode - preferred, histogram - deprecated). |
| **$NEXTDATA** | Byte offset to next data set in the file. |
| **$PAR** | Number of parameters in an event. |
| **$PnB** | Number of bits reserved for parameter number n. |
| **$PnE** | Amplification type for parameter n. |
| **$PnN** | Short name for parameter n. |
| **$PnR** | Range for parameter number n. |
| **$TOT** | Total number of events in the data set. |

## 3.2.19

The optional FCS TEXT segment keywords are as follows:

| | |
|---|---|
| **$ABRT** | Events lost due to data acquisition electronic coincidence. |
| **$BTIM** | Clock time at beginning of data acquisition. |
| **$CELLS** | Description of objects measured. |
| **$COM** | Comment. |
| **$CSMODE** | Cell subset mode, number of subsets to which an object may belong. |
| **$CSVBITS** | Number of bits used to encode a cell subset identifier. |
| **$CSVnFLAG** | The bit set as a flag for subset n. |
| **$CYT** | Type of flow cytometer. |
| **$CYTSN** | Flow cytometer serial number. |
| **$DATE** | Date of data set acquisition. |
| **$ETIM** | Clock time at end of data acquisition. |

| | |
|---|---|
| **$EXP** | Name of investigator initiating the experiment. |
| **$FIL** | Name of the data file containing the data set. |
| **$GATE** | Number of gating parameters. |
| **$GATING** | Specifies region combinations used for gating. |
| **$GnE** | Amplification type for gating parameter number n (deprecated). |
| **$GnF** | Optical filter used for gating parameter number n (deprecated). |
| **$GnN** | Name of gating parameter number n (deprecated). |
| **$GnP** | Percent of emitted light collected by gating parameter n (deprecated). |
| **$GnR** | Range of gating parameter n (deprecated). |
| **$GnS** | Name used for gating parameter n (deprecated). |
| **$GnT** | Detector type for gating parameter n (deprecated). |
| **$GnV** | Detector voltage for gating parameter n (deprecated). |
| **$INST** | Institution at which data was acquired. |
| **$LAST_MODIFIED** | Timestamp of the last modification of the data set. |
| **$LAST_MODIFIER** | Name of the person performing last modification of a data set. |
| **$LOST** | Number of events lost due to computer busy. |
| **$OP** | Name of flow cytometry operator. |
| **$ORIGINALITY** | Information whether the FCS data set has been modified (any part of it) or is original as acquired by the instrument. |
| **$PKn** | Peak channel number of univariate histogram for parameter n (deprecated). |
| **$PKNn** | Count in peak channel of univariate histogram for parameter n (deprecated). |
| **$PLATEID** | Plate identifier. |
| **$PLATENAME** | Plate name. |
| **$PnCALIBRATION** | Conversion of parameter values to any well defined units, e.g., MESF. |
| **$PnD** | Suggested visualization scale for parameter n. |
| **$PnF** | Name of optical filter for parameter n. |
| **$PnG** | Amplifier gain used for acquisition of parameter n. |
| **$PnL** | Excitation wavelength(s) for parameter n. |
| **$PnO** | Excitation power for parameter n. |
| **$PnP** | Percent of emitted light collected by parameter n. |
| **$PnS** | Name used for parameter n. |
| **$PnT** | Detector type for parameter n. |
| **$PnV** | Detector voltage for parameter n. |
| **$PROJ** | Name of the experiment project. |
| **$RnI** | Gating region for parameter number n. |
| **$RnW** | Window settings for gating region n. |

| | |
|---|---|
| **$SMNO** | Specimen (e.g., tube) label. |
| **$SPILLOVER** | Fluorescence spillover matrix. |
| **$SRC** | Source of the specimen (patient name, cell types) |
| **$SYS** | Type of computer and its operating system. |
| **$TIMESTEP** | Time step for time parameter. |
| **$TR** | Trigger parameter and its threshold. |
| **$VOL** | Volume of sample run during data acquisition. |
| **$WELLID** | Well identifier. |

## 3.2.20 Alphabetical listing and detailed description of keywords

For all the keywords below 'n', 'n1', 'n2', etc represent ASCII-encoded integer values.

The character 'f','f1','f2', etc. represents an ASCII-encoded floating point number. Lexical representation of an ASCII-encoded floating point number consists of a finite-length sequence of decimal digits (ASCII 48-57) separated by a dot (".", ASCII 46) as a decimal indicator. An optional leading sign is allowed, either "+" (ASCII 43) or "-" (ASCII 45) is allowed. If the sign is omitted, "+" is assumed. Leading and trailing zeroes are optional. If the fractional part is zero, the period and following zero(es) can be omitted. Standard scientific notation is supported. This means that the described number (called "coefficient" in this case) may be followed by the character "E" (ASCII 69) or "e" (ASCII 101) and an "exponent". An "exponent" is a finite-length sequence of decimal digits (ASCII 48-57) that may optionally be led by a sign character, either "+" (ASCII 43) or "-" (ASCII 45). If the sign is omitted, "+" is assumed. The final resulting value is calculated as "coefficient" x 10^"exponent" ("coefficient" times 10 to the power of "exponent").

The word 'string' represents an UTF-8 encoded TEXT string that can be of any length greater than zero.

The character 'c' represents a single UTF-8-encoded character (may consist of more than one byte).

The '/' character (ASCII 47) is used here as the delimiter for illustrative purposes.

**$ABRT/n/ $ABRT/1265/**

Number of events lost due to data acquisition electronic coincidence effects. The number of aborted events here was 1265.

**$BEGINANALYSIS/n/ $BEGINANALYSIS/123456789/ [REQUIRED]**

This field contains the byte-offset from the beginning of the data set to the beginning of the optional ANALYSIS segment. If there is no ANALYSIS segment, a '0' should be placed in this keyword value. If the ANALYSIS segment is completely contained within the first 99,999,999 bytes of the data set, this value duplicates the offset contained in the HEADER segment. If the ANALYSIS segment exceeds the first 99,999,999 bytes, the offset information in the HEADER segment is set to 0 and the relevant offset information is stored in the keyword value only.

In this example, the ANALYSIS segment begins at byte 123,456,789.

**$BEGINDATA/n/ $BEGINDATA/123456789/ [REQUIRED]**

This field contains the byte-offset from the beginning of the data set to the beginning of the DATA segment. If the DATA segment is completely contained within the first 99,999,999 bytes of the data set, this value duplicates the offset contained in the HEADER segment. If the DATA segment exceeds the first 99,999,999 bytes, the offset information in the HEADER segment is set to 0 and the relevant offset information is stored in the keyword value only.

In this example, the DATA segment begins at byte 123,456,789

**$BEGINSTEXT/n/ $BEGINSTEXT/123456789/ [REQUIRED]**

This field contains the byte-offset from the beginning of the data set to the beginning of the supplemental TEXT segment. If there is no supplemental TEXT segment, the value should be set to '0'. If the supplemental TEXT segment is completely contained within the first 99,999,999 bytes of the data set, this value duplicates the offset contained in the HEADER segment. If the supplemental TEXT segment exceeds the first 99,999,999 bytes, the offset information in the HEADER segment is set to 0 and the relevant offset information is stored in the keyword value only.

In this example, the supplemental TEXT segment begins at byte 123,456,789.

**$BTIM/hh:mm:ss[.cc]/ $BTIM/14:22:10.47/**

Clock time at the beginning of data acquisition. The format of the value is 24-hour clock hours:minutes:seconds.number of fractional seconds in units of 1/100 of a second. The fractional seconds [.cc] is optional. Data acquisition began at 14 hours, 22 minutes, 10 seconds, and 47/100 of a second.

**$BEGINSTEXT/n/ $BEGINSTEXT/123456789/ [REQUIRED]**

This field contains the byte-offset from the beginning of the data set to the beginning of the supplemental TEXT segment. If there is no supplemental TEXT segment, the value should be set to '0'. If the supplemental TEXT segment is completely contained within the first 99,999,999 bytes of the data set, this value duplicates the offset contained in the HEADER segment. If the supplemental TEXT segment exceeds the first 99,999,999 bytes, the offset information in the HEADER segment is set to 0 and the relevant offset information is stored in the keyword value only.

In this example, the supplemental TEXT segment begins at byte 123,456,789.

**$BTIM/hh:mm:ss[.cc]/ $BTIM/14:22:10.47/**

Clock time at the beginning of data acquisition. The format of the value is 24-hour clock hours:minutes:seconds.number of fractional seconds in units of 1/100 of a second. The fractional seconds [.cc] is optional. Data acquisition began at 14 hours, 22 minutes, 10 seconds, and 47/100 of a second.

**$BYTEORD/n1,n2,n3,n4/ $BYTEORD/4,3,2,1/ [REQUIRED]**

This keyword specifies the endianness of the data, i.e., the byte order used to binary store numeric data values in the data set. This value of the keyword corresponds to the order from numerically least significant [1] to numerically most significant [4] in which four binary data bytes are written to compose a 32-bit word in the data acquisition computer. The numbers are separated by commas (ASCII 44). Only two distinct values are allowed:

- $BYTEORD/1,2,3,4/ (little endian, i.e., least significant byte written first, e.g., x86 based personal computers)
- $BYTEORD/4,3,2,1/ (big endian, i.e., least significant byte is written last, e.g., PowerPC including older Apple Macintosh computers prior to switch to Intel-based architecture)

One of these values shall be used to specify the endianness even if the size of data values exceeds 32 bits ($DATATYPE/D/)

**$CELLS/string/ $CELLS/Normal human peripheral blood/**

Type of cells or other objects measured. This specimen is normal human peripheral blood.

**$COM/string/ $COM/Incubation time was 47 minutes./**

This keyword is used to attach a comment to the data set. It should not to be used as a substitute for other standard keywords. This example shows the use of $COM to add a brief note to the data set, a note that otherwise might appear only in a laboratory notebook.

**$CSMODE/n/ $CSMODE/3/**

Cell subset mode, i.e., the number "n" of subsets to which a object may belong. The simplest case is that the cell subset parameter encodes a single value per object as would be indicated by n = 1. If the value of n is greater than 1 it indicates that the value of the cell subset parameter may encode n subset identifiers. In these cases, the $CSVBITS and $CSVnFLAG keyword values will specify how the cell subset values are encoded. It should be noted that regardless of the value for this keyword, a cell subset value of zero indicates that the object is undefined by the analysis scheme that was used. This keyword is used to support clinical use cases where simple gating/classification reproducibility is crucial.

**$CSVBITS/n/ $CSVBITS/4/**

The number of bits used to encode a cell subset value. When the $CSMODE keyword value is greater than 1, the number of bits used to encode a cell subset identifier must be specified by the $CSVBITS keyword value. In the cited example, 4 bits, i.e., values of 0-15, are used to encode cell subset identifiers. See the discussion of the ANALYSIS segment in section 3.4.

**$CSVnFLAG $CSV1FLAG/4096/**

The value used as a "flag" to indicate that the "n" identifier field encodes a value. In the cited example, if bit 13 is set in the value of the cell subset parameter (parameter value AND 8192 is TRUE), one should read the second field of bits to decode the value. It is not necessary to set "flags", but if one wishes to use zero to encode the first subset for any field, one must set a "flag" to indicate that the zero in that field refers to a subset. See the discussion of the ANALYSIS segment in section 3.4.

**$CYT/string/ $CYT/FACScan/**

The name of the flow cytometer used for the data set. Here a FACScan was used.

**$CYTSN/string/ $CYTSN/400E370/**

The serial number of the flow cytometer used for the data set. Here the serial number is 400E370.

**$DATATYPE/c/ $DATATYPE/I/ [REQUIRED]**

This keyword describes the type of data written in the DATA segment of the data set. The four allowed values are 'I', 'F', 'D', or 'A'. The DATA segment is a continuous bit stream with no delimiters. 'I' stands for unsigned binary integer, F stands for single precision IEEE floating point, 'D' stands for double precision IEEE floating point, and 'A' stands for ASCII. The additional keywords $PnB (bits per parameter) and $PnR (range per parameter) are needed to completely describe an event in the DATA segment.

$DATATYPE/I/ means that the events are written as unsigned binary integers. For each parameter in an event, both the maximum length in bits allocated for storage of the parameter and the actual integer range used by the parameter within that allocation are needed. The number of bits per parameter is specified by $PnB. For example, $P1B/16/ specifies that 16 bits are allocated for parameter 1. $P1R/1024/ specifies that parameter 1 values range from 0 to 1023. This allows the data word length to be specified, facilitating compatibility between machines with different data word lengths and enabling bit compression of the data.

$DATATYPE/F/ means that the data are written as single precision floating point values in the IEEE standard format (5). Note that the $PnB keywords should be set to a value of 32 for each parameter in an event. For example, $P1B/32/. In this case, the value of the $P1R keyword represents the original resolution to acquire the data; however, it may not correspond to the range where data is stored. Note that $PnE/0,0/ shall be used for all parameters if $DATATYPE/F/ is used.

$DATATYPE/D/ means that the data are written as double precision floating point values in the IEEE standard format (5). The $PnB keyword should be set to a value of 64 for each parameter in an event. For example, $P3B/64/ says that parameter 3 is allocated 64 bits of storage space. In this case, the value of the $P1R keyword represents the original resolution to acquire the data; however, it may not correspond to the range where data is stored. Note that $PnE/0,0/ shall be used for all parameters if $DATATYPE/D/ is used.

$DATATYPE/A/ is deprecated in FCS 3.1. While still allowed, the users are discouraged from using $DATATYPE/A/ as it will likely not be supported by future revisions of the FCS standard.

$DATATYPE/A/ means that the data are written as ASCII-encoded integer values. In this case, the keyword $PnB specifies the number of bytes allocated per value (one byte per character). This represents fixed format ASCII data. $P1B/4/ indicates that the maximum value for parameter 1 would be 9999. Data are stored in a continuous byte stream, with no delimiters. If the value of the $PnB keyword is the * character, e.g., $P1B/*/, the data are free format and number of characters per parameter value may vary. In this case, all values are separated by one of the following delimiters: "space", "tab", "comma", "carriage return", or "line feed" characters. Note that multiple, consecutive delimiters are treated as a single delimiter. Since there are significant differences between the ways in which consecutive delimiters are treated by different programming languages, care should be taken when using this format. Zero values must be explicitly specified by the "0" (ASCII 48) character. Thus, the string "1,3,, ,3" (note the space between the third and fourth commas) would only specify three values.

**$DATE/dd-mmm-yyyy/ $DATE/01-OCT-1994/**

This keyword specifies the date on which the data set was created. If the beginning and end of data acquisition occur on different dates, the value of the $DATE keyword should correspond to the beginning of acquisition. The format is day-month-year with the number of characters specified by dd-mmm-yyyy. This data set was created on 01 October 1994. Note that the all the character positions should be filled including leading zeros. Accepted abbreviations for the months are: JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC.

**$ENDANALYSIS/n/ $ENDANALYSIS/123456789/ [REQUIRED]**

This field contains the byte-offset from the beginning of the data set to the end of the ANALYSIS segment. If there is no ANALYSIS segment, a '0' should be placed in this keyword value. If the ANALYSIS segment is completely contained within the first 99,999,999 bytes of the data set, this value duplicates the offset contained in the HEADER segment. If the ANALYSIS segment exceeds the first 99,999,999 bytes, the offset information in the HEADER segment is set to 0 and the relevant offset information is stored in the keyword value only.

In this example, the ANALYSIS segment ends at byte 123,456,789

**$ENDDATA/n/ $ENDDATA/123456789/ [REQUIRED]**

This field contains the byte-offset from the beginning of the data set to the end of the DATA segment. If the DATA segment is completely contained in the first 99,999,999 bytes of the data set, this value duplicates the offset contained in the HEADER segment. If the DATA segment exceeds the first 99,999,999 bytes, the offset information in the HEADER segment is set to 0 and the relevant offset information is stored in the keyword value only.

In this example, the DATA segment ends at byte 123,456,789.

**$ENDSTEXT/n/ $ENDSTEXT/123456789/ [REQUIRED]**

This field contains the byte-offset from the beginning of the data set to the end of the supplemental TEXT segment. If there is no supplemental TEXT segment, the value should be set to '0'. If the supplemental TEXT segment is completely contained in the first 99,999,999 bytes of the data set, this value duplicates the offset contained in the HEADER segment. If the supplemental TEXT segment exceeds the first 99,999,999 bytes, the offset information in the HEADER segment is set to 0 and the relevant offset information is stored in the keyword value only. In this example, the supplemental TEXT segment ends at byte 123,456,789.

**$ETIM/hh:mm:ss[.cc]/ $ETIM/14:22:10.47/**

Clock time at the end of data acquisition. The format of the value is 24-hour clock hours:minutes:seconds.number of fractional seconds in units of 1/100 of a second. Data acquisition ended at 14 hours, 22 minutes, 10 seconds, and 47/100 of a second. The fractional seconds keyword value is optional as indicated by the square brackets.

**$EXP/string/ $EXP/A. Smith/**

The name of the person initiating the experiment. This experiment was under the direction of A. Smith.

**$FIL/string/ $FIL/071494.001/**

The name of the file this dataset was originally saved in. The name may or may not include the path. In this example, the data have been saved in a file named 071494.001.

**$GATE/n/ $GATE/2/**

This keyword specifies the number of parameters used for gating. It is analogous to the $PAR keyword, which specifies the total number of parameters for each event in the data set. In this example, there are two gating parameters. The current practice in many flow cytometry laboratories is that the gating parameters are collected as part of the data set. This fact is reflected in the redefinition of the $RnI keyword described below.

**$GATING/string/ $GATING/R1/ $GATING/R1 AND (R2.0R.R3)/**

This keyword specifies the conditions under which the data in the data set have been acquired. The conditions are set through Boolean operations among regions defined below using the $RnI and $RnW keywords. Allowed Boolean operators are AND, OR(inclusive), and NOT. The operands are the regions (Rn). Operators are separated from operands or other operators by spaces or periods. Operator precedence is from left to right unless overridden with parentheses. In the first example, data were collected using gating region R1. Events with parameter values falling outside R1 were excluded from the data set. In the second example, an event is included in the data set only if the appropriate parameter value is inside R1 and is inside R2 or R3 or both.

**$GnE/f1,f2/ $G3E/4.0,0.01/ [DEPRECATED]**

This keyword specifies whether linear or logarithmic amplifiers were used for gating parameter number n. When the amplification is logarithmic the value of f1 specifies the number of logarithmic decades and f2 represents the linear value that would have been obtained for a signal with a log value of 0. In the example above, the data for parameter 3 were collected using a four-decade logarithmic amplifier and the 0 channel represents the linear value, 0.01. When linear amplification is used or when amplification is undefined such as with some calculated parameters, f1 and f2 are set to 0.

The use of gating parameters ($GnE, $GnF, $GnN, $GnP, $GnR, $GnS, $GnT, and $GnV keywords) is deprecated in FCS 3.1. While these keywords are still technically allowed,

implementors are being discouraged to use these since they may not be supported in the next version of FCS.

**$GnF/string/ $G2F/520LP/ [DEPRECATED]**

This keyword specifies the optical filter that was used for the light reaching the detector for gating parameter n. This example shows that the optical filter used for the second gating parameter was a type 520 nm long pass.

The use of gating parameters ($GnE, $GnF, $GnN, $GnP, $GnR, $GnS, $GnT, and $GnV keywords) is deprecated in FCS 3.1. While these keywords are still technically allowed, implementors are being discouraged to use these since they may not be supported in the next version of FCS.

**$GnN/string/ $G1N/FL2/ [DEPRECATED]**

This keyword specifies a short name for gating parameter number n. Here "FL2" is the name for gating parameter 1.

The use of gating parameters ($GnE, $GnF, $GnN, $GnP, $GnR, $GnS, $GnT, and $GnV keywords) is deprecated in FCS 3.1. While these keywords are still technically allowed, implementors are being discouraged to use these since they may not be supported in the next version of FCS.

**$GnP/n1/ $G3P/27/ [DEPRECATED]**

The amount of light collected by the detector for gating parameter number n1 expressed as a percentage of the light emitted by a fluorescent object. In the example, 27% of the emitted light was captured by the detector for gating parameter number 3.

The use of gating parameters ($GnE, $GnF, $GnN, $GnP, $GnR, $GnS, $GnT, and $GnV keywords) is deprecated in FCS 3.1. While these keywords are still technically allowed, implementors are being discouraged to use these since they may not be supported in the next version of FCS.

**$GnR/n1/ $G2R/1024/ [DEPRECATED]**

This keyword specifies the range, n1, of gating parameter n. In this example, the events for gating parameter 2 range from 0 to 1023.

The use of gating parameters ($GnE, $GnF, $GnN, $GnP, $GnR, $GnS, $GnT, and $GnV keywords) is deprecated in FCS 3.1. While these keywords are still technically allowed, implementors are being discouraged to use these since they may not be supported in the next version of FCS.

**$GnS/string/ $G1S/FITC-CD45/**

This keyword specifies a longer name for gating parameter n than is allowed by $GnN. Here, FITC-labeled CD45 is the name for gating parameter 1.

The use of gating parameters ($GnE, $GnF, $GnN, $GnP, $GnR, $GnS, $GnT, and $GnV keywords) is deprecated in FCS 3.1. While these keywords are still technically allowed, implementors are being discouraged to use these since they may not be supported in the next version of FCS.

**$GnT/string/ $G2T/PMT9524/ [DEPRECATED]**

This keyword specifies the detector type for gating parameter n. Here, gating parameter 2 uses a photomultiplier tube (PMT) of type 9524.

The use of gating parameters ($GnE, $GnF, $GnN, $GnP, $GnR, $GnS, $GnT, and $GnV keywords) is deprecated in FCS 3.1. While these keywords are still technically allowed, implementors are being discouraged to use these since they may not be supported in the next version of FCS.

**$GnV/n1/ $G2V/645/ [DEPRECATED]**

This keyword specifies the detector voltage, n1, in Volts for gating parameter n. In this example, the detector for gating parameter 2 is set to 645 Volts.

The use of gating parameters ($GnE, $GnF, $GnN, $GnP, $GnR, $GnS, $GnT, and $GnV keywords) is deprecated in FCS 3.1. While these keywords are still technically allowed, implementors are being discouraged to use these since they may not be supported in the next version of FCS.

**$INST/string/ $INST/Laboratory of FCM, RPCI/**

The institution or laboratory in which the data were collected. In this example, the data were collected in the Laboratory of Flow Cytometry at Roswell Park Cancer Institute.

**$LAST_MODIFIED/dd-mmm-yyyy hh:mm:ss[.cc]/**
**$LAST_MODIFIED/25-SEP-2008 15:22:10.47/**

Timestamp of the last modification of the data set. The $LAST_MODIFIED optional keyword may only be present if the value of the $ORIGINALITY keyword is not set to "Original". Any altering of the data file is consider modification; please see also the $ORIGINALITY and the $LAST_MODIFIER keywords. Note that performing any modifications to FCS data sets generally represents bad practices and the presence of the $LAST_MODIFIED keyword is not meant to encourage these. The original file should always be kept and any modifications should be saved in a copy of the original file.

The format of the timestamp is day-month-year followed by a space followed by 24-hour clock hours:minutes:seconds.number of fractional seconds in units of 1/100 of a second. Note that all the character positions should be filled including leading zeros. Accepted abbreviations for the months are: JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC. The fractional seconds [.cc] is optional. In this example, the data set was last modified on September 25th, 2008 at 15 hours, 22 minutes, 10 seconds, and 47/100 of a second.

**$LAST_MODIFIER/string/ $LAST_MODIFIER/Jack Evil/**

The name of the person performing the last modification of the data set. The $LAST_MODIFIER optional keyword may only be present if the value of the $ORIGINALITY keyword is not set to "Original". Any altering of the data file is consider modification; please see also the $ORIGINALITY and the $LAST_MODIFIED keywords. Note that performing any modifications to FCS data sets generally represents bad practices and the presence of the $LAST_MODIFIER keyword is not meant to encourage these. The original file should always be kept and any modifications should be saved in a copy of the original file.

In this example, Jack Evil is the name of the person performing the last modification of the data set.

**$LOST/n/ $LOST/457/**

This keyword specifies the number of events lost during data acquisition because the computer was busy with other tasks. Here, 457 events were so lost.

**$MODE/c/ $MODE/L/ [REQUIRED]**

This keyword specifies the mode in which the data were acquired. Allowed values for the character c are 'L', 'C' (deprecated), or 'U' (deprecated). These options are described as follows:

L - List mode. For each event, the value of each parameter is stored in the order in which the parameters are described. The number of bits reserved for parameter 1 is described using the $P1B keyword. There can be only one set of list mode data per data set. The $DATATYPE keyword describes the data format. This is the most versatile mode for the storage of flow cytometry data because mode C and mode U data can be created from mode L data.

C (deprecated) - One correlated multivariate histogram is stored in the data set as a multidimensional array. There can be only one such histogram per data set. In storing multiparameter correlated data, the index for the first parameter is incremented first, then the second, etc. For bivariate data, the first data value corresponds to index 1 for parameter 1 and index 1 for parameter 2, the second data value corresponds to index 2 for parameter 1 and index 1 for parameter 2, etc.

The use of correlated multivariate histograms is deprecated in FCS 3.1. While still allowed, implementors are encouraged to avoid this option since it may be discontinued in next versions of FCS.

U (deprecated) - Uncorrelated univariate histograms. There can be more than one univariate histogram per data set. The histogram frequencies for parameter 1 are stored first followed by those for parameter 2, etc. If the univariate histograms have been gated, they must all have been acquired with the same gates so that the total number of events in each histogram is the same.

The use of uncorrelated univariate histograms is deprecated in FCS 3.1. While still allowed, implementors are encouraged to avoid this option since it may be discontinued in next versions of FCS.

**$NEXTDATA/n/ $NEXTDATA/202512/ [REQUIRED]**

An FCS 3.1 data file consists of one or more data sets. When there is more than one data set in an FCS file, this keyword gives the byte offset from the beginning of a data set to the first byte in the HEADER of the next data set in the FCS file. If n is zero (0), this is the final or only data set in the file. This example shows that the next data set begins at byte 202512 from the beginning of the present data set. Each data set stands alone and must contain a full complement of keywords.

Please note that the usage of multiple data sets within a single data file is deprecated unless the multiple data sets are derived one from another. Technically, it is still possible to store multiple data sets within an FCS file but implementors are being discouraged to do so unless one data set is derived from the other one within the data file.

**$OP/string/ $OP/Dave/**

The name of the operator of the flow cytometer. Here Dave was the operator of this instrument.

**$ORIGINALITY/string/**
**$ORIGINALITY/Original/ $ORIGINALITY/NonDataModified/**
**$ORIGINALITY/Appended/ $ORIGINALITY/DataModified/**

Information whether the FCS data set has been modified or is original as acquired by the instrument. If this keyword is present, the value shall be one of "Original", "NonDataModified", "Appended", or "DataModified" indicating the following:

$ORIGINALITY/Original/ indicates that the full data set is as acquired or originally created by an instrument or software. No part of the dataset has been modified.

$ORIGINALITY/NonDataModified/ indicates that there have been changes made to the original data set (as acquired or originally created by an instrument or software), but these changes have not modified anything in the DATA segment. For example, a keyword/value pair may have been altered or added in the TEXT segment or an ANALYSIS segment may have been added in the dataset. See also the $LAST_MODIFIER and the $LAST_MODIFIED keywords.

$ORIGINALITY/Appended/ indicates that there have been changes made to the original data set (as acquired or originally created by an instrument or software), but these changes have kept all original data in the DATA segment and have added new data to the DATA segment. Typically, this would mean adding new (e.g., computed) parameters (e.g., FL2-A/FL2-H ratio, classification results, etc.) or adding new events. In these cases, modifications in the TEXT segment are allowed (and essential since $PAR, etc. need to be updated).

$ORIGINALITY/DataModified/ indicates that there have been changes made to the original data set (as acquired or originally created by an instrument or software) including changes to the original data in the DATA segment. See also the $LAST_MODIFIER and the $LAST_MODIFIED keywords.

Please note that good practice generally demands that original data files be retained and that any modifications to FCS data sets should be carried out with great caution to ensure that the resulting files are identified as derivative. Please note that, for future standards specifications, ISAC is considering mechanisms to verify the integrity of original data sets.

**$PAR/n/ $PAR/5/ [REQUIRED]**

This keyword specifies the total number of parameters stored in each event in the data set. In this example, data for five parameters are stored for each event.

**$PKn/n1/ $PK2/374/ [DEPRECATED]**

For a univariate histogram of parameter n, this keyword specifies the channel number, n1, containing the highest frequency of events. In this example, the peak in the univariate histogram for parameter 2 is located in channel 374. The $PKNn keyword specifies the count in that channel.

In FCS 3.1, the $PKn and $PKNn keywords are deprecated along with the deprecation of histograms.

**$PKNn/n1/ $PKN2/12803/ [DEPRECATED]**

For a univariate histogram of parameter n, this keyword specifies the number of events, n1, in the channel number (histogram bin) containing the maximum event frequency. In this example, the univariate histogram for parameter 2 has a maximum event frequency of 12803. The $PKn keyword above specifies that this peak count occurs at channel 374.

In FCS 3.1, the $PKn and $PKNn keywords are deprecated along with the deprecation of histograms.

**$PLATEID/string/ $PLATEID/27e029f-3d35-4bda-bda4-72cffeae8b2a/**

This keyword specifies the unique identifier associated with the plate from which a well was the source of the data set. In the example, 27e029f-3d35-4bda-bda4-72cffeae8b2a is the identifier of the plate.

**$PLATENAME/string/ $PLATENAME/Plate1 96 Well - U bottom/**

This keyword specifies the name of the plate from which a well was the source of the data set. In the example, Plate1 96 Well - U bottom is the name of the plate.

**$PnB/n1/ $P3B/16/ [REQUIRED]**

For $DATATYPE/I/ (binary integers), $DATATYPE/F/ (floating point numbers), and $DATATYPE/D/ (double precision floating point numbers), this keyword specifies the number of bits allocated, n1, for storage of parameter n. For $DATATYPE/A/ (ASCII-encoded integers), $PnB specifies the number of characters, n, per measured value for parameter n.

For $DATATYPE/F/, the value of $PnB shall always be 32 (bits per value).

For $DATATYPE/D/, the value of $PnB shall always be 64 (bits per value).

For $DATATYPE/A/, an integer value may be specified or $P1B/*/ may be used for variable number of characters per value; see $DATATYPE for further details.

For $DATATYPE/I/, these keywords enable tight bit packing of events. For example, the data storage could be specified by $PnB/10/$PnR/1024/ for each of the n parameters in an event. Then fewer bits would be wasted in storing each event. However, packing these data for storage and unpacking them later for analysis is very time-consuming. In practice, most flow cytometers use $PnB/16/$PnR/1024/ for 10-bit data. A flow cytometer with 8-bit ADCs would use $PnB/8/$PnR/256/ where n represents integers from one to the number of parameters measured.

In this example, the data value for parameter 3 would be stored as two bytes (16 bits). This keyword is used in conjunction with $PnR to determine how the data are actually stored. A flow cytometer with 10-bit analog-to-digital converters (ADCs) would have $PnR/1024/. A 10-bit number would be stored in the 16-bit space allocated by $PnB/16/ leaving 6 empty bits per parameter. Implementors should use a bit mask when reading these list mode parameter values to insure that erroneous values are not read from the unused bits. If $PnR is not a power of two, the next higher power of 2 should be used to create the bit mask and any information in the bits above the next higher power of 2 should be ignored.

**$PnCALIBRATION/f,string/ $P1CALIBRATION/1.234,MESF/**

The $PnCALIBRATION keyword is used to specify the conversion of arbitrary signal units, recorded as parameter values (uncompensated or compensated) to some well defined unit, for example, mean equivalent soluble fluorochrome (MESF) or antibody molecules. The keyword value includes a single positive floating point value and a string value separated with a comma ',' (ASCII 44 or 0x2C) with the following semantics:

- f (a single positive floating point value): represents the number of calibrated units corresponding to a unit signal value of parameter n;
- string (a UTF-8 character string): represents the name of the units corresponding to calibration value. For example if the signal on parameter n has the scale value X then the calibrated value is X * f units.

This transformation is applied to the scale data value, i.e., after any antilog ($PnE) or gain ($PnG) scaling has been performed. In the example, parameter 1 has 1.234 MESF per scale unit. See also Appendix A: Calibration, compensation and related calculations.

**$PnD/string,f1,f2/ $P3D/Linear,0,1024/ $P2D/Logarithmic,4,0.1/**

$PnD is an optional keyword that recommends visualization scale for parameter n. If this keyword is present, the value of the string part shall be one of "Linear" or "Logarithmic". It is not mandatory that any software uses the suggested visualization scale, e.g., the end user may still be able to select another scale to visualize the parameter. If $PnD is missing, analytical software may still be able guess the best scale based on other hints, e.g., $PnE, $PnN, $PnS.

The string value encodes the type of display transformation. Two types of display transforms are recognized by the standard. These are "Linear" and "Logarithmic" and the value should map to the transformation that was applied to the data during acquisition. Note that this is different from $PnE since some software exports all data as linear so that the original transformation is not maintained without use of the $PnD keyword.

Each of the transformations has a different parameter list (floating point numbers f1, f2) that specifies how to construct the transformation.

- $PnD/Linear,f1,f2/ $P3D/Linear,0,1024/

  Data should be displayed as linear scale. Note that the f1 and f2 parameter values are in "scale" units, not "channel" units, see below for details.

f1: Lower bound - the scale value corresponding to the left edge of the display

f2: Upper bound - the scale value corresponding to the right edge of the display

**Example:** $P3D/Linear,0,1024/ specifies a linear display ranging from 0 to 1024 (scale value).

**Example:** $P5D/Linear,100,200/ specifies that the parameter values to be displayed run from 100 to 200.

Note: All parameter values are in "scale" units, not "channel" units. Consider the following cases:

$P3B/8/ $P3R/256/ $P3G/4/ $P3E/0,0/ $P3D/Linear,0,32/

This is a linear parameter with channel values going from 0 to 255. Taking account the gain, the scale values go from 0 to 64. The $P3D specifies a linear display from 0 to 32 scale units, which only encompasses the bottom half of the collected data range.

$P4B/16/ $P4R/1024/ $P4E/4,1/ $P4D/Linear,0,1000/

The display keyword specifies that the data should be shown in linear scaling, with only the bottom 10th of the scale values shown. This will restrict the display to channel values between 0 and 768 (the bottom 3 decades), with channels being distributed exponentially in the linear display.

- $PnD/Logarithmic,f1,f2/ $P2D/Logarithmic,4,0.1/

  Data should be displayed with logarithmic scaling. Note that the f1 and f2 parameter values are in "scale" units, not "channel" units, see below for details.

  f1: Decades - The number of decades to display.

  f2: Offset - The scale value corresponding to the left edge of the display

  **Example:** $P2D/Logarithmic,4,0.1/ specifies a linear display ranging from 0.1 to 1000 (scale value), which is 4 decades of display width.

  **Example:** $P1D/Logarithmic,5,0.01/ specifies a linear display ranging from 0.01 to 1000 (scale value), which is 5 decades of display width.

  Note: All parameter values are in "scale" units, not "channel" units. Consider the following case:

  $P4B/16/ $P4R/1024/ $P4E/4,1/ $P4D/Logarithmic,3,1/

  The display keyword specifies that the data should be shown in logarithmic scaling, with only the bottom 3 decades shown. This will restrict the display to channel values between 0 and 768 (1024*3/4).

**$PnE/f1,f2/ $P3E/4.0,0.01/ [REQUIRED]**

This keyword specifies whether parameter number n is stored on linear or logarithmic scale and includes details about the logarithmic amplification if logarithmic scale is used.

When linear scale is used, $PnE/0,0/ shall be entered. If the floating point data type is used (either $DATATYPE/F/ or $DATATYPE/D/), then all parameters shall be stored as linear with $PnE/0,0/.

When logarithmic scale is used, the value of f1 specifies the number of logarithmic decades and f2 represents the linear value that would have been obtained for a signal with a log value of 0. In the example above, the data for parameter 3 were collected using a four-decade logarithmic amplifier and the 0 channel represents the linear value, 0.01.

Note that both the values f1 and f2 shall either be zero or positive numbers. Especially, $PnE/f1,0/ with f1 > 0 is not a valid entry and shall not be written. If this entry is found in an FCS file, it is recommended to handle it as $PnE/f1,1/.

Explanation: Entries such as $PnE/4,0/ have never been correct. Unfortunately, the lack of clear explanation made these widely used. For $PnE/f1,f2/, f1 specifies the number of logarithmic decades and f2 represents the minimum on the log scale, which cannot be 0 since the logarithm of zero is not defined. For example, $PnE/4,1/ means 4 decades log reaching from 1 to 10000; $PnE/5,0.01/ means 5 decades log reaching from 0.01 to 1000.

Converting from channel values on logarithmic scale to linear scale values:

For $PnR/r/, r>0, $PnE/f1,f2/, f1>0, f2>0: n is a logarithmic parameter with channel values going from 0 to r-1, and scale values going from f2 to f2*10^f1. A channel value xc can be converted to a scale value xs as xs = 10^(f1 * xc /(r)) * f2.

Examples of converting from channel to scale values:

$P1R/1024/, $P1E/4,1/: This is a logarithmic parameter with channel values going from 0 to 1023, and scale values going from 1 to approximately 10000. A channel value xc can be converted to a scale value xs as xs = 10^(4 * xc / 1024) * 1.

$P2R/256/, $P2E/4.5,0.1/: This is a logarithmic parameter with channel values going from 0 to 255, and scale values going from 0.1 to approximately 10^3.5 (~3162). A channel value xc can be converted to a scale value xs as xs = 10^(4.5 * xc / 256) * 0.1.

**$PnF/string/ $P2F/520LP/**

This keyword specifies the optical filter that was used for the light reaching the detector for parameter n. This example shows that the optical filter used for the second parameter was a type 520 nm long pass.

**$PnG/f/ $P2G/10.0/**

This keyword specifies the gain that was used to amplify the signal for parameter n. This example shows that parameter 2 was amplified 10.0-fold before digitization. The gain shall not be used in combination with logarithmic amplification, i.e.,$PnG/f/, f not equal to 1, shall not be used together with $PnE different from $PnE/0,0/.

Converting linearly amplified data from channel values to scale values. For $P1G/g/, g>0, $PnE/0,0/: A channel value xc can be converted to a scale value xs as xs = xc / g.

Example of converting from channel to scale values:

$P1R/1024/, $P1E/0,0/, $P1G/8/: This is a parameter with channel values going from 0 to 1023, and scale values from 0 to approximately 128. In this case, a channel value xc can be converted to a scale value xs as xs = xc / 8.

**$PnL/n1[,n2,...]/ $P1L/488/ P3L/560,625/**

This keyword specifies the excitation wavelength(s) in nm for parameter n. In the first example, the wavelength was 488 nm for parameter number 1. Typically, a single wavelength is related to each of the parameters; however, some of the modern instruments allow for multiple lasers with different wavelengths related to a single parameter. In the second example, two co-axial lasers with different wavelengths (560 nm and 625 nm, respectively) have been used in a way that the result of their excitation activity was detected using the detector related to parameter number 3.

**$PnN/string/ $P3N/FL1-H/ [REQUIRED]**

This keyword is used to specify the short name of parameter n. Here parameter 3 has a short name of FL1-H. Parameter names must be unique, i.e., each name (value of an $PnN keyword) may occur only once in the data set. The character ',' (comma, ASCII 44) is not allowed to be part

of the value of this keyword (comma in the short name would introduce conflict with some other keywords, e.g., $TR, $SPILLOVER).

The value "TIME" shall be used in order to indicate TIME measurement with steps indicated in $TIMESTEP.

**$PnO/n1/ $P2O/200/**

This keyword specifies the excitation power, n1, in milliwatts for the light source associated with the measurements for parameter n. Here 200 mW was used to produce the signal associated with parameter 2.

**$PnP/n1/ $P4P/50/**

The amount of light collected by the detector for parameter number n expressed as a percentage of the light emitted by a fluorescent object. In the example, 50% of the emitted light was captured by the detector for parameter number 4.

**$PnR/n1/ $P2R/1024/ $P2R/262144/ [REQUIRED]**

For $DATATYPE/I/ this keyword specifies the maximum range, n1, of parameter n. For $MODE/L/ (list mode data), this corresponds to the ADC range, e.g., 1024. In that case, the data values can range from 0 to 1023. For univariate histogram data ($MODE/C/ or $MODE/U/), n1 is the number of channels in the histogram for parameter n. For $DATATYPE/I/, the value of $PnR also indirectly specifies the bit mask that should be used when reading values. See the description of $PnB and section 3.3 for more information.

For $DATATYPE/F/ and $DATATYPE/D/ this keyword specifies the maximum range, n1, of parameter n used to acquire the data. $DATATYPE/F/ with $PnR/262144/ is commonly used by some instruments. The actual value stored in the data set may exceed this range on both sides of the interval. Specifically, there may be negative values as well as values greater than n1, e.g., as a consequence of compensation. No bit mask shall be applied when reading $DATATYPE/F/ or $DATATYPE/D/ data.

**$PnS/string/ $PnS/CD45 FITC Fluorescence - Area/**

This keyword specifies a long name (full name) to be used as an axis label in a plot of parameter n. Here CD45 FITC Fluorescence - Area is the label. $PnS is the long name equivalent of $PnN. Unlike $PnN, $PnS may include commas and is not required to be unique among the parameters.

**$PnT/string/ $P2T/PMT9524/**

This keyword specifies the detector type for parameter n. Here, parameter 2 uses a photomultiplier tube (PMT) of type 9524.

**$PnV/f1/ $P2V/445.5/**

This keyword specifies the detector voltage, f1, in Volts for parameter n. In this example, the detector for parameter 2 is set to 445.5 Volts.

**$PROJ/string/ $PROJ/AML patient study/**

This keyword provides the name of the project. Here it is an AML patient study.

**$Rnl/string1,[string2]/ $R3I/P2,P4/ $R2I/G3/**

This keyword associates a gating region number, n, with one or two parameters, here shown as string1 and string2. The two strings are of the form "Pn" or "Gn". "Pn" stands for collected parameter n, while "Gn" stands for gating parameter n. In the first example, gating region 3 is associated with a bivariate dot plot or bivariate histogram for parameters 2 and 4. The $RnW

keyword described below specifies the shape of the gating region. In the second example, gating region 2 is associated with gating parameter 3. See the discussion for the $GATE keyword.

**$RnW/f1,f2[;f3,f4;...]/ $R1W/345.5,366.8/**

This keyword specifies the window settings for gating region n. This window setting is useful only if the $RnI keyword is also specified. If the keyword $RnI has only a single value, then f1 and f2 specify the inclusive lower and upper bounds for the window in a univariate histogram. For example, $R2I/3/$R2W/345.5,366.8/ specifies that gating region 2 is associated with gating parameter 3. The gated events must range between channels 345.5 and 366.8 inclusive. If the $RnI keyword value has two values, then the window exists in a bivariate plot and it is specified in the $RnW keyword as a polygon. The x and y coordinates of the first point in the polygon are the pair f1,f2. The next point is separated from the first by a ';' character and is represented as f3,f4 above. The polygon can contain any number of points separated by semicolons. The first point and the last point are assumed to be connected.

For example, $R1I/2,3/$R1W/310,205;515,304;480,615;240,514;354,542/ specifies that region 1 is defined in parameter 2 and 3 and that the region 1 window is a 5-sided polygon in this 2-parameter space. The $GATING keyword will specify the way the windows will be used (AND, OR, etc.).

**$SMNO/string/ $SMN0/7874A5/**

This keyword is a placeholder for identification of the specimen (sample). It may include any vendor or user specific identification. Generally, multimple specimens may be related to the same source (see $SRC). The value of $SMNO may also include a tube. See also $PLATEID, $PLATENAME, and $WELLID for plate and well identification. In the example, the specimen is identified as 7874A5.

**$SPILLOVER/n,string1,string2,...,f1,f2,f3,f4,.../**

The $SPILLOVER keyword is used to specify the information necessary to carry out fluorescence compensation in the form of a spillover matrix (see below). The matrix gives the spillover that is recommended to be used to calculate fluorescence compensated data for post acquisition analysis from the uncompensated data stored in the file. The spillover matrix references parameters that are stored uncompensated in the data set (the stored data is always uncompensated in FCS 3.1). Only one spillover matrix may be provided for the data set. Note that compensation can also be provided separately using the Gating-ML standard (6).

The keyword value includes 1 positive integer value, two or more string values and four or more floating point values with the following semantics:

$SPILLOVER/n,$Str_1$,$Str_2$,…,$Str_n$,$S_{11}$,$S_{12}$,…,$S_{1n}$,$S_{21}$,$S_{22}$,…,$S_{nn}$/:

n (a single positive integer value): n represents the number of parameters in the matrix.

$Str_i$ (n character strings): Stri represents the parameter name corresponding to the "i"-th parameter in the spillover matrix. The parameter name is understood as the exact value of its $PnN keyword (including type suffices such as '-A', '-H', '-W').

$S_{ij}$ - (n*n floating point values): $S_{ij}$ represents the spillover coefficient from parameter "i" to parameter "j". The spillover coefficient from parameter "i" to parameter "j" is the ratio of the amount of signal in the "j" channel to the amount of signal in the "i" channel for particles carrying only the "i" parameter dye.

**Explanation:**

This formulation allows for a subset of the collected parameters to be included in the matrix (the number of parameters in the matrix is specified by the first element). The matrix will have a size of n x n. The total number of elements in the keyword value must be "1 + n + n x n"; the first value is a positive integer value, the next n values are strings of characters referencing compensated

parameters by their $PnN names, and the remaining n*n floating point values represent their spillover matrix. All values are comma-delimited.

The first element in the value is the number of parameters. The value of this element must be greater than 1 and less than or equal to the number of parameters in the file ($PAR).

The next n elements identify the parameters included in the matrix. The parameters can be present in any order. Each row of the spillover matrix corresponds to the spillovers associated with the corresponding parameter to all the parameters.

The next "n x n" elements specify the spillover coefficient values. The values are specified as floating point (with optional scientific notation). For example: "3.6e-2"; "0.04", "0.56", etc.

**Example:**

$P3N/FL1-A/ $P3S/FITC/

$P4N/FL2-A/ $P4S/PE/

$P5N/FL3-A/ $P5S/Cy5PE/

$SPILLOVER/2,FL1-A,FL2-A,1.0,0.1,0.03,1.0/

This keyword specifies a 2-way compensation between parameters FL1-A ("FITC") and FL2-A ("PE"), with a spillover of 10% from FITC to PE and 3% from PE to FITC.

**Spillover Matrix 1 – Example spillover matrix specifying a 2-way compensation between parameters FL1-A ("FITC") and FL2-A ("PE"), with a spillover of 10% from FITC to PE and 3% from PE to FITC.**

| | Detector | |
|---|---|---|
| Fluorochrome | FITC | PE |
| FITC | 1.0 | 0.1 |
| PE | 0.03 | 1.0 |

**Example:**

$P3N/FL1-A/ $P3S/FITC/

$P4N/FL2-A/ $P4S/PE/

$P5N/FL3-A/ $P5S/Cy5PE/

$SPILLOVER/3,FL2-A,FL1-A,FL3-A,1.0,0.03,0.2,0.1,1.0,0.0,0.05,0,1.0/

This keyword specifies a 3-way compensation including parameters FL1-A (“FITC”), FL2-A (“PE”), and FL3-A (“Cy5PE”). The spillover matrix should be constructed as follows:

**Spillover Matrix 2 – Example spillover matrix specifying a 3-way compensation including parameters FL1-A (“FITC”), FL2-A (“PE”), and FL3-A (“Cy5PE”).**

| | Detector | | |
|---|---|---|---|
| Fluorochrome | PE | FITC | Cy5PE |
| PE | 1.0 | 0.03 | 0.2 |
| FITC | 0.1 | 1.0 | 0.0 |
| Cy5PE | 0.0 | 0.05 | 1.0 |

Computation of the compensated values is accomplished by:

(1) Constructing a column vector *e* of the PE, FITC, and Cy5PE values (event from the file, scale values);

(2) Inverting the spillover matrix *S* - create matrix $S^{-1}$

(3) Transposing the matrix $S^{-1}$ - create matrix $T(S^{-1})$

(4) The resulting column vector having the compensated PE, FITC, and Cy5PE values can be obtained by matrix multiplication of $T(S^{-1})$ x *e*.

The same compensation can be achieved with row vectors as follows:

(1) Constructing a row vector *e* of the PE, FITC, and Cy5PE values (event from the file, scale values);

(2) Inverting the spillover matrix S - create matrix $S^{-1}$

(3) The resulting row vector having the compensated PE, FITC, and Cy5PE values can be obtained by matrix multiplication of *e* x $S^{-1}$.

**$SRC/string/ $SRC/J. Doe, HIV positive patient/**

This keyword specifies the source of the specimen. Note that this keyword value could contain patient information, which may be protected by region specific laws and guidelines. The acquiring laboratory may choose to use encoded information for this keyword value.

**$SYS/string/ $SYS/Mac OS X v10.5/**

This keyword specifies the type of computer and the operating system under which the data set was collected. Here the data set was collected on a Mac OS X v10.5.

**$TIMESTEP/f/ $TIMESTEP/0.01/**

The presence of this keyword indicates that time has been collected as one of the parameters in the data set. Comparing values of $PnN against the string "TIME" allows to identify, which parameter represents the time. $TIMESTEP specifies the time step in seconds.

In this example, the time step is 1/100 of a seconds. For this example, an implementor specifies $DATATYPE/I/$P6N/TIME/$P6B/16/$P6R/65536/$TIMESTEP/0.01/. When the first event in the data set is captured by the computer, a zero value is entered into parameter 6 in this first event. When an event e is captured s seconds after the first event, the value s*100 is entered into parameter 6 in the event e. In this example, the maximum time range is 65536/100 seconds (10 minutes and 55.35 seconds).

**$TOT/n/ $TOT/25000/ [REQUIRED]**

This keyword specifies the total number of events in the data set. This data set contains 25000 events.

**$TR/string,n/ $TR/FSC-H,54/**

This keyword specifies the parameter name (the value of the $PnN keyword) which serves as the trigger signal for an event. The number, n, is the channel number of the threshold signifying an event. When the threshold is exceeded, an event is declared. Here forward scatter (FSC-H) is the trigger signal and the event threshold is at channel 54.

**$VOL/f/ $VOL/50500/**

This keyword specifies the volume of sample that was consumed during data acquisition. The volume is expressed as a floating point number in nanoliters. In the example, the data set was acquired in a sample volume of 50500 nanoliters. This equals to 50.5 µL (microlitres).

**$WELLID/string/ $WELLID/A07/**

This keyword specifies the location on the plate of the well that was the source of the data set.

The numeric portion of the well ID has leading zeros such that for all wells for a given plate type use the same number of characters to represent the location (for example, 96 well plate has 8 rows labeled A - H, and 12 columns numbered 01-12; a 384 well plate has 16 rows labeled A-P and 24 columns numbered 01 - 24, etc.). In the example, A07 is the well identifier.

## 3.3 DATA Segment

The DATA segment contains the raw data in one of three modes (list, correlated or uncorrelated) described in the primary TEXT segment by the $MODE keyword value. The data are written to the DATA segment in one of four allowed formats (binary, floating point, double precision floating point or ASCII) described by the $DATATYPE keyword value (see also the description of the $DATATYPE keyword).

For list mode storage with the binary integers data type ($MODE/L/$DATATYPE/I/), the $PnB set of keywords specify the bit width for the storage of each parameter. The $PnR set of keywords specify the channel number range for each parameter. For example, $P1B/16/ $P1R/1024/ specifies a 16-bit field for parameter 1 and a range for the values of parameter 1 from 0 to 1023, which corresponds to 10 bits. Implementors should use a bit mask when reading these list mode parameter values to insure that erroneous values are not read from the 6 unused bits. If $PnR is not a power of 2, the next higher power of 2 should be used to create the bit mask and any information in the bits above the next higher power of 2 should be ignored.

For list mode storage with the floating point data type ($MODE/L/$DATATYPE/F/ or $MODE/L/$DATATYPE/D/), the $PnB set of keywords also specify the bit width for the storage of each parameter; however, this width is required to correspond to 32 and 64 bits respectively. No bit mask shall be applied when reading these list mode parameter values. Negative values as well as values exceeding the value of the $PnR keyword are legitimate.

For list mode storage with data stored as ASCII-encoded integers (($MODE/L/$DATATYPE/A/), the $PnB set of keywords specifies the number of bytes allocated per value (one byte per character) unless $PnB/*/ is used. Please note that $DATATYPE/A/ is deprecated and see the description of $DATATYPE for further details.

## 3.4 ANALYSIS segment

ANALYSIS is an optional segment that, when present, contains the results of data processing. It is often the case that analysis is performed off-line, after the data has been collected and stored in a data set. Therefore, the ANALYSIS segment typically contains information added to a copy of the original file. For examples, the results of cell cycle analysis or immunophenotype determinations often involve more complex analyses than can be performed in "real time" as the data is collected and stored. The ANALYSIS segment has the same structure as the TEXT segment; i.e., it consists of a series of keyword-value pairs. There are no required keywords for the ANALYSIS segment. The optional FCS keywords are listed in 3.4.1 with one line descriptions and in 3.4.2 with full descriptions and examples. Implementors may add their own keywords.

The ANALYSIS segment may be used for identifying cell subsets, determined either by region drawing or by some partitioning method such as cluster analysis (7). This may be particularly useful for immunophenotyping data. Three approaches to identifying cell subsets are supported by this standard as discussed below. The first two use the least space in the data set but require the cell subsets be disjoint. The third approach adds a parameter to each event and supports overlapping cell subset assignments.

- In method 1, the implementor uses the TEXT segment keyword-value pairs $CSMODE/1/ and $CSTOT/n/ to specify that there is one group of cell subsets containing n disjoint subsets of cells. The TEXT segment keyword-value pair $CSVBITS/8/ is used to indicate that the cell subset assignments for each event are stored in a binary vector of unsigned

characters (8 bits each) whose length is the number of events in the data set. This vector is stored in another segment following the ANALYSIS segment. The DATA segment contains a copy of the original data with the events written in the same order as in the original data set. In the ANALYSIS segment, $CSnNUM is used to count the number of cells in each of the n subsets.

- In method 2, the implementor uses the TEXT segment keyword-value pairs $CSMODE/1/ and $CSTOT/n/ as above but does not use the $CSVBITS keyword. In the DATA segment, the events are written out one cell subset at time rather than in the original event order. In the ANALYSIS segment, $CSnNUM is used to count the number of cells in each of the n subsets. No other segment is required.
- Method 3 creates an additional cell subset (CS) parameter for each event in the data set. Cell subsets may be defined by the method, e.g., cluster analysis, neural network, Boolean gates on combinations of parameters, hyperplanes in n-dimensional space, etc. The value of the parameter may encode a single subset identifier number for each event ($CSMODE/1/) or more than one identifier number per event (value of $CSMODE greater than 1). The meanings of the identifier numbers are specified by the values of the $CSnNAME keywords in the TEXT and ANALYSIS segments. If the value of the CS parameter is 0 (zero), that event is unclassified by the definitions used to assign cell subsets. If the classification scheme creates unique non-overlapping populations, e.g., CD4 T cells, CD8 T cells, B cells, monocytes/macrophages, neutrophils, etc., then the simplest approach is to set the value of $CSMODE to "1" and use 1 == CD4 T cell, 2 == CD8 T cells, etc. In some situations, it may be useful to be able to assign a single cell to more than one defined subset. For example, to extend the preceding example, subset identifiers 1 - 5 would correspond the definitions listed above with 6 == lymphocytes and 7 == mononuclear cells. This scheme would require $CSMODE/3/ since a single cell could belong to three defined subsets. Operationally, assuming that an event in the data set is a CD4 T cell, then the first bit field would encode a value of 1 (CD4 T cell), the second bit field would encode a value of 6 (lymphocyte), and the third bit field would encode a value of 7 (mononuclear cell). The bit fields and their interpretations in these cases would be defined by the values of the $CSVBITS and the $CSVnFLAG keywords as outlined in the reference (7). Method 3 also supports the creation of an ANALYSIS segment that includes a summary for the results written as the values for the keywords pertaining to the numbers of cells in each subset, etc. Method 3 has the size "cost" of an additional parameter, but it permits one to include a complete and explicit record of an analysis as an integral part of a data set.

### 3.4.1 Optional FCS ANALYSIS segment keyword list

The optional FCS ANALYSIS segment keywords are as follows:

| | |
|---|---|
| **$CSDATE** | Cell subset analysis date. |
| **$CSDEFFILE** | Cell subset definition file name. |
| **$CSEXP** | Name of person who performed the cell subset analysis. |
| **$CSnName** | Name of cell subset number n. |
| **$CSnNUM** | Number of cells in cell subset number n. |

### 3.4.2 Optional FCS ANALYSIS segment keywords

For all the keywords below 'n', 'n1', 'n2', etc. represent ASCII-encoded integer values, the character 'f', 'f1', 'f2', etc. represents ASCII-encoded floating point numbers (with lexical representation as described in 3.2.20), the word 'string' represents an UTF-8 encoded TEXT string that can be of any length greater than zero, and the character 'c' represents a single UTF-8-encoded character (may consist of more than one byte). The '/' character (ASCII 47) is used here as the delimiter for illustrative purposes.

**$CSDATE/dd-mmm-yyyy/ $CSDATE/26-OCT-94/**

Cell subset date. This keyword specifies the date on which the data set containing the cell subset analysis was created. The format is of the date is the same as that for $DATE. This data set was created on 26 October 1994.

**$CSDEFFILE/string/ $CSDEFFILE/c:\filename.dat/**

Cell subset definition file. The string is the name of the file containing the information needed to define each of the cell subsets. In the example the cell subset definition file is named filename.dat and is located on drive c.

**$CSEXP/string/ $CSEXP/A. Smith/**

Cell subset experimenter. Name of the person who performed the cell subset analysis. Here, A. Smith performed the cell subset analysis.

**$CSnName/string/ $CS2N/lymphocytes/**

Cell subset name. This is a string naming cell subset number n. In the example, cell subset 2 is named "lymphocytes".

**$CSnNUM/n1/ $CS2NUM/3456/**

This keyword specifies the number of cells, n1, in cell subset number n. In the example, cell subset 2 contains 3456 cells.

## 3.5 CRC Value

The CRC word is computed for the part of each data set beginning with the first byte of the HEADER segment and ending with the last byte of the final segment of the data set (which could be a TEXT, DATA, ANALYSIS or OTHER segment). The CRC word is a 16-bit cyclic redundancy check value (8). This 16-bit CRC word conforms to the CCITT standard (Comité Consultatif International Téléphonique et Télégraphique; 1993 renamed to ITU-T: the International Telecommunication Union - Telecommunication Standardization Sector). This CRC uses the CCITT polynomial X16 + X12 + X5 and requires that each input character be interpreted as its bit-reversed image. These requirements are satisfied by the icrc function (8) if the last two function arguments are 0 and -1, respectively. The CRC value will be placed as ASCII in the 8 bytes immediately after the last segment of the data set. If an implementor chooses not to compute and store a CRC word then the 8 bytes immediately after the last segment of the data set should be filled with ASCII '0' characters.

## 3.6 Other Segments

Implementors may create any number of OTHER segments as they choose.

# 4 References


1. Murphy RF, Chused TM: A proposal for a flow cytometric data file standard. Cytometry 5:553-555, 1984.
2. Dean PN, Bagwell CB, Lindmo T, Murphy RF and Salzman GC: Data File Standard for Flow Cytometry. Cytometry 11:323-332, 1990.
3. Seamer LC, Bagwell CB, Barden L, Redelman D, Salzman GC, Wood JC, Murphy RF. Proposed new data file standard for flow cytometry, version FCS 3.0. Cytometry. 1:28:118-22, 1997.
4. The Unicode Consortium: The UNICODE Standard, Version 1.0, vol. 1. Addison-Wesley Publishing Co. Inc., Reading, MA, 1991.
5. Institute of Electrical and Electronics Engineers: IEEE 754 - IEEE Standard for Floating-Point Arithmetic. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4610935.
6. Spidlen J, Leif RC, Moore W, Roederer M, Brinkman RR. Gating-ML: XML-based Gating Descriptions in Flow Cytometry. Cytometry A 2008; 73(12): 1151-1157.
7. Redelman D, Coder DM: Cell subset (CS) parameter to record the identities of individual cells in flow cytometric data. Cytometry 18:95-102, 1994.
8. Press WH, Teukolsky SA, Vetterling WT, Flannery BP: Numerical Recipes in C. 2nd ed. Cambridge University Press, Cambridge, UK, 1992.

# Appendix A: Calibration, compensation and related calculations

The $PnCALIBRATION keywords allow for a subset of the collected parameters to be calibrated to some well defined units, for example, mean equivalent soluble fluorochrome (MESF) or antibody molecules. If the spillover matrix (defined by the $SPILLOVER keyword) is normalized (i.e, $C_{ii}$ = 1) then calibration can be applied to either the uncompensated or compensated data. This information can be handled in a number of algebraically equivalent ways depending on the needs of different programs and in fact can be ignored altogether, which will result in behavior identical with the current common practice (ca 2009).

The first approach is to carry out compensation calculations as currently done and then use the calibration data to adjust only the scale values displayed on graphs and such descriptive statistics as are not scale invariant. For example the scale minimum and maximum, mean, median and standard deviation are multiplied by the appropriate conversion factor; the C.V and R.C.V are left unchanged; the variance needs to be multiplied by the conversion factor squared. The advantage of this approach is that data remain within a standard range, which simplifies computing display coordinates but the user sees calibrated units.

The second approach is to form a diagonal matrix $D$ with rows and columns defined as for the spillover matrix, the appropriate conversion factors in the diagonal position ($D_{ii}$) and 1s for any other diagonal elements and 0s elsewhere. If the inverse of the spillover matrix $S$ is right multiplied by the diagonal matrix $D$, i.e., $C$ = Inv($S$) * $D$, then the ordinary compensation calculations will also rescale the data to calibrated units. This corresponds to a column multiplication of the $S$ matrix on the column matching parameter n. Several other algebraically equivalent formulations of doing this exist of course, for example left multiplying the transpose of the inverse of the spillover matrix when column vectors are used instead of row vectors or an appropriate row multiplication. This approach is computationally efficient and likely to be useful when the data are passed into some general statistical discovery software, for example R, SAS or Mathematica.

# Appendix B: Major Differences between FCS 3.0 and FCS 3.1.

Changes marked by two stars (★★) are required to consider when implementing an FCS file reader. Changes marked by one or two stars (★ or ★★) are required to take into account when implementing an FCS writer.

★★ The FCS version identifier (data set offset 00-05) has been changed from "FCS3.0" to "FCS3.1".

★★ The $UNICODE keyword has been discontinued with all keyword values encoded in UTF-8.

★★ The $BTIM and $ETIM keywords are using the 1/100 of a second unit rather than the 1/60 of a second unit for the optional number of fractional seconds in the keyword value. Accordingly, the format of $BTIM and $ETIM keyword values has changed from hh:mm:ss[:tt] to hh:mm:ss[.cc].

★★ The format of the $PnL keyword changed from $PnL/n1/ to $PnL/n1[,n2,...]/ allowing for multiple co-axial lasers with different wavelengths to be associated with a single parameter.

★ The optional $SPILLOVER keyword is being used to specify compensation instead of the $COMP keyword. Note that in FCS 3.1, the $SPILLOVER keyword is the only standardized way to specify compensation.

★ The required values of the $PnN keyword have been removed except for the value "TIME" being required for a time parameter related to the value of the "$TIMESTEP" keyword ($PnN is still a required keyword). Comma is not allowed as part of a $PnN name.

★ The value of the $BYTEORD keyword has been restricted to either $BYTEORD/1,2,3,4/ (little endian) or $BYTEORD/4,3,2,1/ (big endian). Mixed byte orders are no longer supported.

★ The delimiter character has been restricted to any single-byte ASCII character from the range of 1-126 (01-7E hex).

★ If the floating point data type is used (either $DATATYPE/F/ or $DATATYPE/D/), then all parameters shall be stored as linear with $PnE/0,0/.

- The optional $PnD keywords have been introduced to specify the preferred display for a parameter.
- The optional $PnCALIBRATION keywords have been added to specify the conversion of arbitrary signal units, recorded as parameter values (uncompensated or compensated) to some well defined unit, for example, mean equivalent soluble fluorochrome (MESF) or antibody molecules.
- The optional $VOL keyword has been introduced to specify the sample volume.
- The $ORIGINALITY, $LAST_MODIFIER and $LAST_MODIFIED optional keywords have been introduced to distinguish between an original and an altered data set.
- The $PLATEID, $PLATENAME, and $WELLID optional keywords have been introduced for plate and well identification.

- The ASCII data type ($DATATYPE/A/) feature has been deprecated (i.e., still allowed but not suggested due to the possibility of being discontinued in next versions of FCS).
- The use of multiple data sets within a single data file has been deprecated unless these are derived from each other.
- The use of gating parameters ($GnE, $GnF, $GnN, $GnP, $GnR, $GnS, $GnT, and $GnV keywords) is deprecated in FCS 3.1. While these keywords are still technically allowed, implementors are being discouraged to use these since they may be discontinued in the next versions of FCS.
- The use of histograms ($MODE other than L) and the $PKn, $PKNn keywords are deprecated in FCS 3.1. Implementors are encouraged to avoid histograms since they may be discontinued in next versions of FCS.
- The documentation has been improved in several places, such as the description of the $PnE keyword.