ISAC
International Society for Advancement of Cytometry

FCS 3.2

Data File Standard for Flow Cytometry – Version 3.2

Josef Spidlen[*1], *Wayne Moore*[*2], *David Parks*[*3], *Michael Goldberg*[4], *Kim Blenman*[5], *James S Cavenaugh*[6], *the ISAC Data Standards Task Force*[†]

Ryan Brinkman[7]

* Contributed equally; [1]BD Life Sciences – FlowJo, Ashland, OR, USA; [2]Genetics Department, Stanford University School of Medicine, Stanford, CA, USA; [3]Stanford Shared FACS Facility, Stanford University, Stanford, CA, USA; [4]BD Biosciences, San Jose, CA, USA; [5]Yale School of Medicine, New Haven, CT, USA; [6]Consultant; [7]Terry Fox Laboratory, BC Cancer Agency, Vancouver, BC, Canada

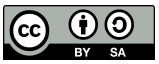
Version 3.2 – 2020-07-03

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, then in 1997 as FCS 3.0, and again in 2010 as FCS 3.1. This document is the FCS 3.2 specification, which includes support for new instrument types as well as several improvements suggested by the community.

[†]List of ISAC Data Standards Task Force members provided in Appendix D.

FCS 3.2 has undergone an extensive revision process and has been formally approved by the ISAC Data Standards Task Force. This document is 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.

Abstract

The flow cytometry data file standard provides the specifications needed to completely describe flow cytometry data sets within the confines of the file containing the experimental data. 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, in 1997 as FCS 3.0, and again in 2010 as FCS 3.1. We report here on the next version of the Flow Cytometry Standard data file format. FCS 3.2 is a revision based on suggested improvements from the community. The unchanged goal of the standard is to provide a uniform file format that allows files created by one type of acquisition hardware and software to be analyzed by any other type.

The FCS 3.2 standard retains the basic FCS file structure and most features of previous versions of the standard. Changes included in FCS 3.2 address potential ambiguities in the previous versions and provide a more robust standard.

The major changes include improved terminology and wording, new generic carrier identification, improved range description for floating point data, and formal identification of measurement type (such as scatter, fluorescence, index, time, calculated values, spectral values and other). A controlled vocabulary of fluorochrome and mass tags is introduced. Some unused, deprecated features have been removed, such as the support for encoding histograms rather than providing list mode data and support for data encodings other than little and big endian. Others have been deprecated and will be removed in a future version, such as the support for ASCII data type. Certain previously optional keywords become mandatory in this version. A CRC reference implementation is provided in order to clarify potential ambiguities and facilitate adoption by third parties, see Appendix B. Please see Appendix C for a complete list of changes.

Keywords

FCS, flow cytometry, cytometry, analytical cytology, data standard, file format, bioinformatics

Contents

List of Figures

List of Tables

List of Examples

1 Overview

1.1 Introduction

The goal of the Flow Cytometry Data File Standard is to facilitate the development of software for reading and writing flow cytometry data files in a standardized format. Application of a standard file format allows files created on one type of instrument (FCS writer) to be read and analyzed using any FCS compatible reader. The original FCS standard was published in 1984 as FCS 1.0 [1], amended in 1990 as FCS 2.0 [2, 3], then again in 1997 as FCS 3.0 [4], and finally in 2010 as FCS 3.1 [5, 6].

Over the past years, FCS 3.1 has served its purpose well, with only a few update requests from the scientific community. To address these requests, the International Society for the Advancement of Cytometry Data Standards Task Force (ISAC DSTF) has developed a new revision of the specification. Major changes include improved terminology and wording, new generic carrier identification, improved range description for floating point data, and formal identification of measurement type (such as scatter, fluorescence, index, time, calculated values, spectral values and other). A controlled vocabulary of fluorochrome and mass tags is introduced. In order to streamline the adoption of the file format, some unused, deprecated features have been removed, such as the support for encoding histograms rather than providing list mode data and support for data encodings other than little and big endian. Others have been deprecated and will be removed in a future version, such as the support for ASCII data type.

Certain previously optional keywords become mandatory in this version. A CRC reference implementation is provided in order to clarify potential ambiguities and facilitate adoption by third parties. A complete list of changes is provided in Appendix C. This document can be found at the `http://flowcyt.sourceforge.net/fcs/fcs32.pdf` web site.

1.2 Scope

This is version 3.2 of the Flow Cytometry Data File Standard (FCS 3.2). Previous versions of this specification can be found in references [1, 3–5]. 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. This document is primarily intended for cytometry software and hardware developers.

1.3 Conformance

1.3.1 File Conformance

To be conformant with FCS 3.2, 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 any segments not described in the HEADER segment of the data set.

1.3.2 Software and hardware conformance

To be compliant with FCS 3.2, a software application or hardware instrument shall be able to *read*, *write*, or *read and write* FCS 3.2 conformant files, and it shall meet the following criteria:

- When FCS 3.2 files are produced (written), then these shall be FCS 3.2 conformat as per section 1.3.1.
- When FCS 3.2 files are read, then the software application or the hardware instrument shall be able to correctly read and interpret all of the data contained in any *minimum FCS 3.2 conformant file*. A *minimum FCS 3.2 conformant file* is an FCS file conformant with FCS 3.2 as per section 1.3.1 with only the required keyword, value pairs

in the TEXT segment of the file, a DATA segment, but no data in the ANALYSIS segment and no OTHER segments.

1.4 Normative References

The following referenced documents are indispensable for the application of this standard. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

- Bradner S, The Internet Engineering Task Force, Network Working Group. Request for Comments: 2119 – Key words for use in RFCs to Indicate Requirement Levels. 1997. [7]
- American National Standards Institute. ANSI INCITS 4-1986 (R2007). Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII). 1986 Edition, January 1, 1986. [8]
- Yergeau F, The Internet Engineering Task Force, Network Working Group. Request for Comments: 3629 – UTF-8, a transformation format of ISO 10646. 2003. [9]
- Institute of Electrical and Electronics Engineers. IEEE Std 754-2008 – IEEE Standard for Floating-Point Arithmetic. 2008. [10]

1.5 The Content of this Specification

This specification consists of the following parts:

(a) **Normative:** This document providing a detailed description of the FCS 3.2 file format.

(b) **Informative:** A reference implementation of CRC value calculation that is compatible with the FCS 3.2 file format as specified in section 3.7.

This standard is available from `http://flowcyt.sourceforge.net/fcs/fcs32.pdf`. Upon manuscript publication, the specification may also be downloaded from ISAC web pages available at `http://www.isac-net.org/`.

2 Terminology and Requirements

2.1 Acronyms and Abbreviations

Table 1 lists acronyms and abbreviations used in this specification.

Table 1: Acronyms and abbreviations used in this specification.

Abbreviation	**Description**
ADC	Analog to Digital Converter
ASCII	American Standard Code for Information Interchange
CV	Coefficient of Variation
CCITT	Comité Consultatif International Téléphonique et Télégraphique
CRC	Cyclic Redundancy Check
DSTF	Data Standards Task Force
FCS	Flow Cytometry Standard
ISAC	International Society for Advancement of Cytometry
JSON	JavaScript Object Notation
MESF	Mean Equivalent Soluble Fluorochrome
RFC	Request for Comments
URL	Uniform Resource Locator
UTF	Unicode Transformation Format
XML	Extensible Markup Language

2.2 Terminology

2.2.1 FCS file

An FCS file is a data file that is conformat with this specification as per section 1.3.1. An FCS data file consists of one or more FCS data sets.

2.2.2 FCS data set

An FCS data set (or just a data set) is the collection of information produced by an analytical instrument as it carries out its measurements on some number of particles. In this document the terms particles, cells and objects may be used more-or-less interchangeably to indicate the sample material analyzed to provide measurements.

2.2.3 FCS segment

An FCS segment is a section of an FCS data set. The FCS specification recognizes the HEADER segment, the *primary* and the *supplemental* TEXT segments, the DATA segment, the ANALYSIS segment and user-defined OTHER segments. The HEADER segment starts at the beginning of the FCS data set. The position of all the other segments is defined by offset information in the HEADER segment and in the *primary* TEXT segment as described in sections 3.1 and 3.2, and in Table 2.

2.2.4 Electronic event

An event occurs when an object of interest, usually a cell, arrives at the interrogation region of the instrument. There some external influence is applied, usually a laser, sometimes an electric field and in mass cytometry metal tags are liberated by plasma or ion beams. Detectors arranged around the interrogation point capture the results, ultimately as electronic signals. Even when there are multiple interrogation points, such as a multi-laser system, all the signals are considered related to a single event, referred to as an electronic event.

2.2.5 FCS measurement

An FCS measurement is a numerical estimate of some feature of one or more of the signals associated with an electronic event. For example, pulse height, area and width are common but other features are possible. The time an event occurred or the interval between events

may be measured. FCS measurements may also encompass mathematically transformed measurements *e.g.* log fluorescence, the ratio of two other measurements and spectrally compensated dimensions.

Note that an FCS measurement was called a "parameter" in previous versions of the FCS data file standard [1,3–5] and is referred to as FCS dimension in Gating-ML. For the purpose of this specification, we decided to abandon the term "parameter" due to the inconsistency of its meaning in the field of flow cytometry vs. common understanding in other fields, such as mathematics, statistics, probability, logic, computer science, engineering.

2.2.6 List mode data set

A list mode data set is a data set containing a sequential list of the measurements recorded for each electronic event, *i.e.*, stored one after the other in a list and for each event measurements are stored in a fixed order. 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 FCS measurements. In database terminology, the events would be the rows and the measurements the columns. List mode data files are typically used for event-by-event data acquisition. Currently, FCS is the most widely (and almost solely) used list mode data file format in flow cytometry. The FCS 3.2 file format supports list mode data only. Previous versions of the FCS file format [1,3–5] supported both list mode data and histograms.

2.2.7 Event value

An event value is a value recorded for a particular event in a particular FCS measurement. This specification distinguishes two types of event values: the *channel values* and the *scale values*. *Channel values* are values extracted from FCS files as described further in this specification taking keywords such as `$PAR`, `$TOT`, `$DATATYPE`, `$BYTEORD`, `$PnB`, and `$PnR` into account. *Scale values* are obtained by converting *channel values* using additional information

stored in the `$PnE` and `$PnG` keywords.

2.2.8 Keyword

A keyword is the label of a meta data field. Keywords are used to provide information about data captured in the FCS data file. For example, `$TOT` is a keyword specifying the total number of events in the data set. The FCS specification defines several required (section 3.2.21) and optional keywords (sections 3.2.22 and 3.2.23). Some of the optional keywords have been deprecated, which means that they may still be used, but their usage is not recommended as they may be removed in a future version of this specification. Vendors may also define their own custom keywords (section 3.2.17).

2.2.9 Keyword value

Each keyword is assigned a non-empty string value. The keyword definition will declare if this value is to be interpreted as a string, integer or floating point value. Some keyword values are themselves structured types, for example `$SPILLOVER`.

2.2.10 Keyword-value pair

A keyword-value pair is a keyword with its associated value. For example, a `12345` value of the `$TOT` keyword (section 3.3.65) indicates that there are 12345 events in the data set. Within this document, we use the `keyword`◇`value`◇ notation to indicate a keyword-value pair, for example `$TOT`◇`12345`◇. The ◇ symbol represents the delimiter character (see section 3.2.6). A keyword-value pair is a *required* keyword-value pair if and only if the keyword is a required keyword. A keyword-value pair is an *optional* keyword-value pair if and only if the keyword is an optional or custom (vendor-specific) keyword.

2.2.11 Implementor

An implementor is the entity that creates the software to read or write FCS conformant data files.

2.2.12 Probe

A probe is a reagent used in flow cytometry to identify and evaluate the amount of a target of interest in or on cells, e.g. mouse anti-human CD3 Alexa Fluor 488. Probes are composed of one or two parts. All probes have a target binder, which is a biomolecule or a chemical compound that directly binds a target of interest. Some probes have an intrinsic ability to be detected and visualized natively (e.g. DAPI) without additional components. In others, labels called tags (e.g. Alexa Fluor 488 or ^{141}Pr) have been added to detect or visualize target binders that lack these intrinsic abilities.

2.3 Conventions in this Document

2.3.1 Terms indicating requirement levels

The terms *shall*, *should*, and *may* in this document are to be interpreted as described in RFC 2119 [7]. The word *shall* is used to indicate mandatory requirements to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*). The word *should* is used to indicate that among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*). The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted to*).

2.3.2 Numerical values

Numerical values in this document shall be interpreted as base 10 unless otherwise specified.

2.3.3 Byte offsets

All byte offsets are referenced to the beginning of the FCS data set using zero origin indexing, *i.e.*, the first data set in an FCS file begins at byte zero of the FCS file.

2.4 General Requirements

2.4.1 FCS file

An FCS 3.2 data file shall consist of at least one data set. An FCS 3.2 data file should not consist of more than one data set unless the multiple data sets are derived one from another, for example, if one data set contains the data as acquired, and the other data set contains a post-processed version of those. This could, for example, mean raw fluorescence data in the first data set and compensated data in the second data set.

2.4.2 FCS data set

An FCS data set shall consist of at least two non-empty segments including a HEADER segment and a primary TEXT segment. It may (and normally will) contain additional segments including a DATA segment, an ANALYSIS segment, a supplemental TEXT segment and any number of implementor-defined OTHER segments. The ANALYSIS segment may be empty. The DATA segment may also be empty (*e.g.*, `$TOT`◇`0`◇).

2.4.3 HEADER segment

The HEADER segment shall identify the FCS data set as FCS 3.2 and it shall contain ASCII [8] encoded 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, thus the number of bytes in the segment equals to (1 + "last byte offset" - "first byte offset").

2.4.4 TEXT segments

The TEXT segments shall contain a series of keyword, value pairs that describe the format of the DATA segment and most of the experimental operating conditions. The *primary* TEXT segment shall contain all required . If present, the *supplemental* TEXT segment may contain optional (sections 3.2.22 and 3.2.23) and vendor-specific keyword, value pairs only (section 3.2.17).

2.4.5 DATA segment

The DATA segment shall contain list mode data. Previous versions of the FCS file format [1,3–5] supported both, list mode data and histograms. In FCS 3.2, histograms are no longer supported.

2.4.6 ANALYSIS segment

The ANALYSIS segment may contain the analysis of the data set. The format of the ANALYSIS segment is not restricted by the FCS specification and it may include any kind of data, including binary data, keyword, value pairs, an XML fragment, a JSON segment, URL links, etc.

3 FCS File Structure

3.1 HEADER Segment

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 (`FCS3.2`). Note that there is no space character between the FCS and the 3.2 in the identifier. The next 4 bytes (6–9) are occupied by space characters (ASCII code 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 3.2 these offsets remain limited to 8 bytes. Each ASCII [8] encoded integer offset is right justified (with ASCII spaces as necessary) in its 8 byte space. Leading zeros may or may not be used. As summarized in Table 2, 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 next offset (bytes 34 - 41) is that for the last byte of the DATA segment. The next offset (bytes 42 - 49) is that for the start of the ANALYSIS segment. The next offset (bytes 50 - 57) is that for the last byte of the ANALYSIS segment. If there is no ANALYSIS segment, then 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 might not be interpretable by others unless appropriate information is passed on by the implementor.

FCS 3.2 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, zeros shall be substituted in the HEADER segment for that segment's begin and end byte offset. The correct byte offsets shall then be provided as keyword, value pairs using the `$BEGINDATA`, `$ENDDATA`, `$BEGINANALYSIS` and `$ENDANALYSIS` keywords in the *primary* TEXT segment (section 3.2.2). When a segment is contained completely within the first 99,999,999 bytes of a data set, the byte offsets for that segment shall be provided in the HEADER segment and

duplicated in the TEXT segment as keyword values. Note that if the ANALYSIS offsets in the HEADER segment are zero, the `$BEGINANALYSIS` and `$ENDANALYSIS` keywords shall be checked to determine if an ANALYSIS segment is present.

The file structure is also demonstrated in Figures 1 and 2. Figure 1 shows an example FCS 3.2 file with 2 data sets. The first data set contains a *primary* TEXT segment, a DATA segment, an ANALYSIS segment and one other segment. All segments are located within the first 99,999,999 bytes of the data set. Keywords `$BEGINDATA`, `$ENDDATA`, `$BEGINANALYSIS` and `$ENDANALYSIS` shall duplicate the information from the HEADER segment (not shown). Figure 2 shows an example FCS 3.2 with a single data set containing a *primary* TEXT segment and a DATA segment. The DATA segment is located outside of the first 99,999,999 bytes of the data set. Keywords `$BEGINDATA` and `$ENDDATA` point to the offsets of the DATA segment. Keywords `$BEGINANALYSIS`, `$ENDANALYSIS`, `$BEGINSTEXT` and `$ENDSTEXT` shall either contain `0` or be omitted since there is no ANALYSIS segment, neither a *supplemental* TEXT segment.

In the event that no data are collected but a file is still produced, `$TOT` shall be 0 and the offsets to the data segment in the header and the `$BEGINDATA` and `$ENDDATA` keywords may contain any values, but those shall be ignored and considered undefined by FCS consumers.

Table 2: Contents of HEADER segment fields and the byte offsets to the beginning and end of each field. Each offset is right justified in its field. Leading zeros may or may not be used.

Contents	**Start byte position**	**End byte position**
`FCS3.2`	00	05
Four space characters (ASCII code 32)	06	09
ASCII-encoded offset to first byte of the *primary* TEXT segment	10	17
ASCII-encoded offset to last byte of the *primary* 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 (*filled with spaces if there is no ANALYSIS segment*)	42	49
ASCII-encoded offset to last byte of ANALYSIS segment (*filled with spaces if there is no ANALYSIS segment*)	50	57
ASCII-encoded offset to first byte of user-defined OTHER segment (*optional*)	58	65
ASCII-encoded offset to last byte of user-defined OTHER segment (*optional*)	66	73
... there may be 0 or any number of user-defined OTHER segments)	...	...

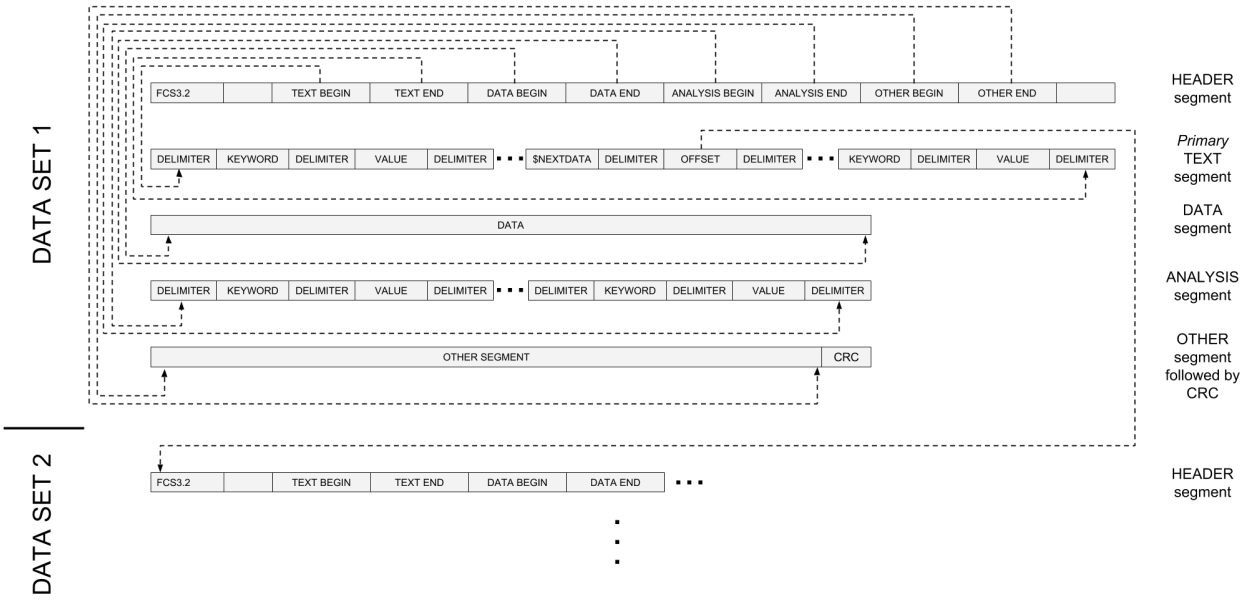


Figure 1: FCS 3.2 file structure example of an FCS file with 2 data sets. The first data set contains a *primary* TEXT segment, a DATA segment, an ANALYSIS segment and one other segment. All segments are located within the first 99,999,999 bytes of the data set. Keywords `$BEGINDATA`, `$ENDDATA`, `$BEGINANALYSIS` and `$ENDANALYSIS` shall duplicate the information from the HEADER segment (not shown).

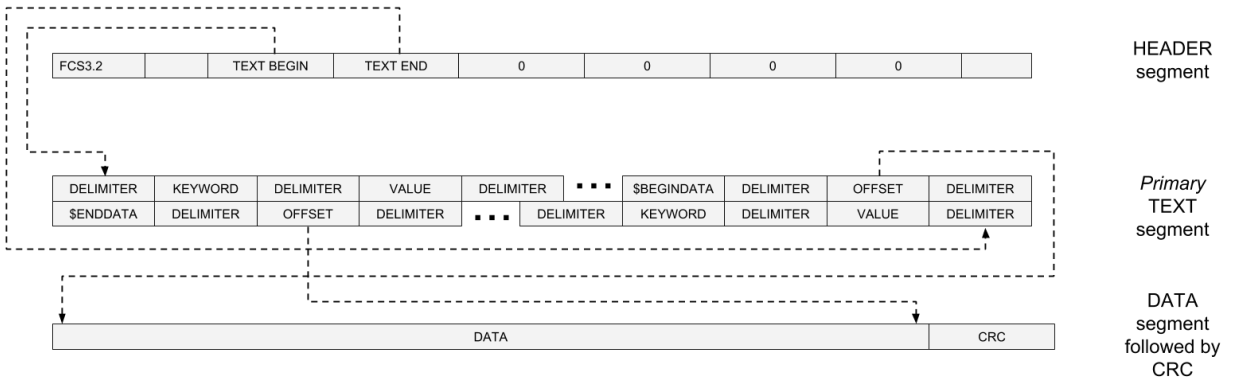


Figure 2: FCS 3.2 file structure example of an FCS file with a single data set containing a *primary* TEXT segment and a DATA segment. The DATA segment is located outside of the first 99,999,999 bytes of the data set.

Example 1 shows a HEADER segment describing a data set with DATA segment fully contained within the first 99,999,999 bytes of the data set and no ANALYSIS segment. The ∗ character is used to represent a space character in the example. The TEXT segment starts at byte 256 from the location of the 'F' in FCS3.2 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. Therefore, the length of the DATA segment equals to 200,664 (*i.e.*, 202455 + 1 - 1792). There is no ANALYSIS segment, so the start and end offsets are shown as zeros. They could also be left blank. Note that when reading the file, the `$BEGINANALYSIS` and `$ENDANALYSIS` keywords still need to be checked in order to determine that no ANALYSIS segment is present. Note also 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 (ASCII code 32). In this example, the segments are in the order HEADER, TEXT, DATA. 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.

The contents of the HEADER segment; the * character is used to represent a space character in this example:

```
FCS3.2*********256****1545****1792**202455*******0*******0
```

Example 1: HEADER segment describing a data set with DATA segment fully contained within the first 99,999,999 bytes of the data set and no ANALYSIS segment

Example 2 shows a HEADER segment describing a data set with DATA segment exceeding the first 99,999,999 bytes. This is indicated by the zeros in the begin DATA and end DATA offset positions. Therefore, the byte offsets to begin DATA and end DATA, are located only in the `$BEGINDATA` and `$ENDDATA` keyword values in the *primary* TEXT segment.

The contents of the HEADER segment; the * character is used to represent a space character in this example:

```
FCS3.2*********256****1545*******0*******0*******0*******0
```

Example 2: HEADER segment describing a data set with DATA segment exceeding the first 99,999,999 bytes of the data set

Example 3 shows a HEADER segment describing a data set in which the *primary* TEXT segment follows the DATA segment. This is possible as long as the *primary* TEXT segment is still fully contained within the first 99,999,999 bytes of the data set.

The contents of the HEADER segment; the * character is used to represent a space character in this example:

```
FCS3.2******202451**203140****1792**202450*******0*******0
```

Example 3: HEADER segment describing a data set with DATA segment before the TEXT segment

3.1.1 Notes on segment offset specification

As shown in Table 2, segment positions and sizes shall be specified by stating the offsets to the first and to the last bytes. Consider a DATA segment that begins after the first 1024 bytes of the file, and is 24 bytes long. In this case, the offset to the first byte of the DATA

segment is 1024 (*i.e.*, `$BEGINDATA◇1024◇`, see Section 3.3.3); there are 1024 bytes before the first byte of data, and the first byte of data occurs as the 1025^{th} byte). Similarly, the offset to the last byte of the DATA segment is 1047 (*i.e.*, `$ENDDATA◇1047◇`, see Section 3.3.17); there are 1047 bytes before the last byte of the data). Please note that for a non-zero length segment, the length of a segment in bytes equals 1 + the difference between the beginning and ending offsets. In our example, the length of the DATA segment is 1 + 1047 - 1024 = 24 bytes.

3.2 TEXT Segments

3.2.1 Contents of TEXT segments

The TEXT segments (*primary* and *supplemental*) shall contain a series of keyword, value pairs (see section 2.2.10) that describe various aspects of the data set. For example, `$TOT◇5000◇` is a keyword-value pair indicating that the total number of events in the file is 5,000. `$TOT` is the keyword, `5000` is its value, and ◇ represents the delimiter character (see section 3.2.6). The `$` character being the first character of the keyword indicates that `$TOT` is an FCS-defined keyword (sections 3.2.16, 3.2.21, 3.2.22 and 3.2.23).

3.2.2 Contents of the primary TEXT segment

An FCS data set shall contain a *primary* TEXT segment with all required keyword, value pairs (section 3.2.21). The *primary* TEXT segment may also contain any number of optional (sections 3.2.22 and 3.2.23) and custom keyword, value pairs (section 3.2.17).

3.2.3 Primary TEXT segment location

The *primary* TEXT segment shall be contained entirely in the first 99,999,999 bytes of the data set. The byte offset to the beginning and end of the *primary* TEXT segment shall be stated in the HEADER segment as shown in Table 2.

3.2.4 Contents of the supplemental TEXT segment

An FCS data set may contain an optional *supplemental* TEXT segment. A *supplemental* TEXT segment may contain optional (sections 3.2.22 and 3.2.23) and custom keyword, value pairs only (section 3.2.17) assuming those keywords have not been used in the *primary* TEXT segment already (see section 3.2.11 for keyword uniqueness requirements). Required keyword, value pairs shall not be placed in the *supplemental* TEXT segment.

3.2.5 Supplemental TEXT segment location

The *supplemental* TEXT segment may be placed anywhere in the data set after the HEADER segment. The byte offset to the beginning and end of the *supplemental* TEXT segment shall be provided in the required `$BEGINSTEXT` and `$ENDSTEXT` keyword, value pairs, which shall be located in the *primary* TEXT segment.

3.2.6 Delimiter

The delimiter is the first character of the *primary* TEXT segment and shall be subsequently placed in the primary TEXT segment to separate keywords from keyword values. This character shall also be used as the delimiter in the *supplemental* TEXT segments if those exist. The delimiter may be any ASCII [8] 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), but it is recommended to use the *Line Feed* character (ASCII code 10) as the delimiter. Since *Line Feed* is a non-printable character, we use the ◊ symbol to indicate the delimiter in the provided examples within this document. In order to separate keywords from keyword values, the delimiter shall be placed at the start and end of each keyword value. The delimiter shall not be the first character in any keyword or keyword value. If the delimiter appears in a keyword or keyword value, it must be immediately followed by a second delimiter. This sequence of two delimiters shall be interpreted as a single "regular" character rather than a delimiter. For

example, `$SYS◊RSX-11◊◊M◊` shows a value of `RSX-11◊M` for the keyword `$SYS`. In contrast, `Key◊◊M1◊56◊` assigns a value of `56` to the keyword `Key◊M1`. Note that since empty keywords or keyword values are not permitted, two consecutive delimiters can never occur between a value and a keyword.

3.2.7 Encoding keywords

The ASCII character code [8] shall be used for all keywords throughout an FCS 3.2 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.

3.2.8 Encoding keyword values

The UTF-8 [9] character code shall be used for all keyword values throughout an FCS 3.2 file. UTF-8 is backward compatible with ASCII [8] in that all characters 00 (hex) through 7F (hex) inclusive are encoded the same way in UTF-8 and ASCII. This means that any ASCII-encoded keyword value is automatically also compatible with this requirement. For example, as specified in section 3.2.9, numerical keyword values shall be ASCII-encoded, which is also UTF-8 compatible and therefore not in conflict with requirements of this section.

3.2.9 Numerical keyword values

All numeric values shall be encoded using their English ASCII representation without padding with non-numerical characters, including spaces and other white characters. Leading zeros (ASCII code 48) may be used. The comma character (ASCII code 44) shall not be used. Non-English numerals (*e.g.*, Chinese, Arabic, Cyrillic, ...) shall not be used to encode the values of numeric keywords (but those numbers/characters may be used in the text of string-based keyword values).

Integer values are indicated by symbols `m`, `n`, `p`, `d`, `n1`, `n2`, `n3`, `n4` etc. throughout section 3.3. Integer values shall be encoded using a standard lexical representation. The

standard lexical representation of an ASCII-encoded integer number consists of a finite-length sequence of decimal digits (ASCII codes 48–57). For example, an implementor may use `$BEGINDATA◇00012345◇` but shall not use `$BEGINDATA◇ 12345◇`.

Floating point values are indicated by symbols `f`, `f1`, `f2`, etc. throughout section 3.3. Floating point values shall be encoded using either a standard lexical representation or a scientific notation as described below. The standard lexical representation of an ASCII-encoded floating point number consists of a finite-length sequence of decimal digits (ASCII codes 48–57) separated by a dot (`.`, ASCII code 46) as a decimal indicator. An optional leading sign may be provided as either `+` (ASCII code 43) or `-` (ASCII code 45). If the sign is omitted, then `+` is assumed. Leading and trailing zeroes may also be used. If the fractional part is zero, the period and following zero(es) may be omitted. Alternatively, a standard scientific notation may be used. This means that the described number (called *coefficient* or *mantissa* in this case) may be followed by the character `E` (ASCII code 69) or `e` (ASCII code 101) and an *exponent*. An *exponent* is a finite-length sequence of decimal digits (ASCII codes 48–57) that may optionally be led by a sign character, either `+` (ASCII code 43) or `-` (ASCII code 45). If the sign is omitted, then `+` is assumed. The final resulting value shall be calculated as $coefficient \times 10^{exponent}$.

3.2.10 No additional characters in TEXT segments

The TEXT segments shall 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.

3.2.11 Uniqueness of keywords

Keywords shall be unique in data sets, i.e., there shall be no multiple instances of the same keyword in an FCS data set.

3.2.12 No empty keywords or keyword values

Keywords and keyword values must have lengths greater than zero.

3.2.13 Case insensitivity of keywords

Keywords are case insensitive, they may be written in a file in lower case, upper case, or a mixture of the two. An FCS file reader shall ignore the keyword case. For example, the `$TOT` keyword may also be written as `$tot` or `$Tot` etc.

3.2.14 Case sensitivity of keyword values

Keyword values are case sensitive.

3.2.15 Default keyword values

There are no default values for any keywords.

3.2.16 FCS-defined keywords

FCS-defined keywords shall begin with the `$` character (ASCII code 36). Only FCS-defined keywords may begin with the `$` character. Implementors shall not redefine FCS-defined keywords.

3.2.17 Custom keywords

Implementors may define and use their own custom keyword, value pairs to save additional information in the TEXT segments. Custom keywords shall not start with the `$` character (ASCII code 36). Although custom keywords are not standardized by the FCS specification, several vendors shared the description of their custom keywords. This information is publicly available as attachments to FlowRepository dataset *FCS collection for software testing*, dataset id `FR-FCM-ZZZ4` (`http://flowrepository.org/id/FR-FCM-ZZZ4`).

3.2.18 Required and optional keywords

There are required (section 3.2.21) and optional (section 3.2.22) FCS keyword, value pairs. Some of the optional keywords have been deprecated (section 3.2.23), which means that they may still be used, but their usage is not recommended as they may be removed in a future version of the specification. The required keyword, value pairs represent the minimum set needed to successfully read and write an FCS data set. In order for an FCS reader to be compliant it must be able to recognize all required FCS keywords, see section 1.3.2 for additional conformance requirements.

3.2.19 FCS measurement description keywords

FCS measurement description keywords are numbered consecutively in the order in which the FCS measurements are written to the file, beginning with number 1. FCS measurement description keywords are the following: `$P`*n*`B`, `$P`*n*`E`, `$P`*n*`N`, `$P`*n*`R`, `$P`*n*`ANALYTE`, `$P`*n*`CALIBRATION`, `$P`*n*`D`, `$P`*n*`DATATYPE`, `$P`*n*`DET`, `$P`*n*`F`, `$P`*n*`FEATURE`, `$P`*n*`G`, `$P`*n*`L`, `$P`*n*`O`, `$P`*n*`P`, `$P`*n*`S`, `$P`*n*`T`, `$P`*n*`TAG`, `$P`*n*`TYPE`, and `$P`*n*`V`. Due to legacy reasons, these keywords start with `$P` since *FCS measurements* were referred to as *parameters* in previous versions of the FCS specification. The *n* in those keywords shall be replaced with a decimal natural number, without leading zeros, corresponding to the FCS measurement number. For example, `$P1N` is the keyword capturing the short name of FCS measurement 1.

3.2.20 Keyword names with *n*

Besides the FCS measurement description keywords listed in section 3.2.19, there are additional sets of standard keywords where *n* is meant to be replaced with an ASCII-encoded integer. Those include the following: `$R`*n*`I`, `$R`*n*`W` (*n* refers to a gating region number).

3.2.21 List of required FCS keywords

Table 3 lists required FCS keywords. These keywords shall be included in the *primary* TEXT segment.

Table 3: Required FCS keywords

Keyword	**Description**
`$BEGINDATA`	Byte-offset to the beginning of the DATA segment
`$BYTEORD`	Byte order (endianness) that the data are saved in
`$CYT`	Type of flow cytometer
`$DATATYPE`	Default type of data in DATA segment (integer or floating point)
`$ENDDATA`	Byte-offset to the last byte of the DATA segment
`$NEXTDATA`	Byte offset to next data set in the file
`$PAR`	Number of FCS measurements
`$P`n`B`	Number of bits reserved for FCS measurement number n
`$P`n`E`	Amplification type for FCS measurement number n
`$P`n`N`	Short name for FCS measurement number n
`$P`n`R`	Range for FCS measurement number n
`$TOT`	Total number of events in the data set

3.2.22 List of optional FCS keywords

Table 4 lists all optional FCS keywords. These keywords may be included in the *primary* or *supplemental* TEXT segments. See section 3.3 for a detailed description of all the keywords.

Table 4: Optional FCS keywords

Keyword	**Description**
$ABRT	Number of events lost due to limits of signal processing electronics
$BEGINANALYSIS	Byte-offset to the beginning of the ANALYSIS segment
$BEGINDATETIME	Time stamp at the beginning of data acquisition
$BEGINSTEXT	Byte-offset to the beginning of a *supplemental* TEXT segment
$CARRIERID	Carrier identifier
$CARRIERTYPE	Carrier type
$CELLS	Description of objects measured
$COM	Comment
$CYTSN	Flow cytometer serial number
$ENDANALYSIS	Byte-offset to the last byte of the ANALYSIS segment
$ENDDATETIME	Time stamp at the end of data acquisition
$ENDSTEXT	Byte-offset to the last byte of a supplemental TEXT segment
$EXP	Name of investigator initiating the experiment
$FIL	Name of the data file containing the data set
$FLOWRATE	Acquisition flow rate settings
$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 the last modification
$LOCATIONID	Carrier location identifier
$LOST	Number of events lost due to computer being busy
$MODE	Data mode (list mode only is supported)
$OP	Name of the flow cytometry operator
$ORIGINALITY	Information whether the FCS data set has been modified

$P*n*ANALYTE	Target molecule or process for FCS measurement *n*
$P*n*CALIBRATION	Conversion of measured values to well defined units
$P*n*D	Suggested visualization scale for FCS measurement *n*
$P*n*DATATYPE	The type of data in DATA segment for FCS measurement *n*.
$P*n*DET	Detector name for FCS measurement *n*
$P*n*F	Name of optical filter for FCS measurement *n*
$P*n*FEATURE	Signal feature (e.g., Area) of FCS measurement *n*
$P*n*G	Amplifier gain used for acquisition of FCS measurement *n*
$P*n*L	Excitation wavelength(s) for FCS measurement *n*
$P*n*O	Excitation power for FCS measurement *n*
$P*n*S	Long name (description) used for FCS measurement *n*
$P*n*T	Detector type for FCS measurement *n*
$P*n*TAG	The dye or other readout marker used for FCS measurement *n*
$P*n*TYPE	The type of measurement captured in FCS measurement *n*
$P*n*V	Detector voltage for FCS measurement *n*
$PROJ	Name of the experiment project
$SMNO	Sample, tube or other identifier
$SPILLOVER	Fluorescence spillover (spectral overlap) matrix
$SRC	Source of the specimen (*e.g.*, patient identifier, cell types)
$SYS	Type of computer and its operating system
$TIMESTEP	Time step for the time measurement
$TR	Trigger measurement and its threshold
$UNSTAINEDCENTERS	A vector of central values of an unstained sample
$UNSTAINEDINFO	Information on the origin and use of $UNSTAINEDCENTERS
$VOL	Volume of sample run during data acquisition

3.2.23 List of deprecated FCS keywords

Table 5 lists all deprecated FCS keywords. These keywords may be included in the *primary* or *supplemental* TEXT segments, but they are obsolete and their use is not recommended. These keywords may be removed from the future version of the FCS specification. See section 3.3 for a detailed description of all the keywords.

Table 5: Deprecated FCS keywords

Keyword	**Description**
`$BTIM`	Use $BEGINDATETIME instead
`$DATE`	Use $BEGINDATETIME instead
`$ETIM`	Use $ENDDATETIME instead
`$GATING`	Region combinations used for gating
`$PLATEID`	Plate identifier
`$PLATENAME`	Plate name
`$P`*n*`P`	Percent of emitted light collected by FCS measurement *n*
`$R`*n*`I`	Gating region for FCS measurement *n*
`$R`*n*`W`	Window settings for gating region *n*
`$WELLID`	Well identifier

3.3 Alphabetical Listing and Description of FCS Keywords

For all the keywords described in the section, `m`, `n`, `p`, `d`, `n1`, `n2`, `n3`, `n4`, etc. represent ASCII-encoded integer values, and symbols `f`, `f1`, `f2`, etc. represent ASCII-encoded floating point numbers. See section 3.2.9 for information on how numerical values shall be encoded. The terms `string`, `string1`, `string2`, etc. represent a UTF-8 [9] encoded character string that can be of any length greater than zero. The character `c` represents a single UTF-8 [9] encoded character, which may consist of more than one byte. Square brackets (*i.e.*, `[]`) enclosing any term mark that term as optional. The term `hh:mm:ss[.cc]` represents the time expressed in the 24-hour clock format as hours:minutes:seconds.centiseconds. A centisecond is 1/100 of a second. The centisecond may or may not be provided, which is indicated by enclosing the optional part in square brackets. The term `dd-mmm-yyyy` represents a date in the format

of *day-month-year* where the day is expressed using a two digit number between 01 and 31 (inclusive, leading zeros shall be present), the month is expressed using a 3 character abbreviation, one of `JAN`, `FEB`, `MAR`, `APR`, `MAY`, `JUN`, `JUL`, `AUG`, `SEP`, `OCT`, `NOV` or `DEC`, and the year is expressed using a 4 digit number. The term `yyyy-mm-ddThh:mm:ss[TZD]` represents a time stamp in a ISO 8601 [11] compatible complete extended combined date and time of day representation format (see ISO 8601:2004 [11], section 4.3.2). The time zone designator (TZD) may or may not be provided, which is indicated by enclosing the optional part in square brackets. The ◊ symbol is used to indicate a delimiter character in this section, see section 3.2.6.

3.3.1 $ABRT

`$ABRT◊n◊`

`$ABRT◊1265◊`

Abort Count. The number of events detected on the instrument that could not be evaluated due to restrictions in the electronic processing of the data. The specific interpretation depends on the instrument and software, so it is important for these to be documented. There were 1265 aborted events in the example. See also 3.3.29 `$LOST` for a count of events lost in computer processing.

3.3.2 $BEGINANALYSIS

`$BEGINANALYSIS◊n◊`

`$BEGINANALYSIS◊123456789◊`

The value of this keyword 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, this keyword shall either not be used, or a `0` shall 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 information 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 actual offset information is stored in the value of this keyword only. This keyword is optional as of FCS 3.2. If this keyword is not present, then the position and presence of the ANALYSIS segment are determined by the relevant information the HEADER segment. In this example, the ANALYSIS segment begins at byte 123,456,789.

3.3.3 $BEGINDATA [REQUIRED]

`$BEGINDATA◊n◊`

`$BEGINDATA◊123456789◊`

The value of this keyword 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 actual offset information is stored in the value of this keyword only. For zero-length DATA segments (i.e., if the value of the `$TOT` keyword equals to `0`), the `$BEGINDATA` keyword may contain any value and such value shall be ignored by FCS consumers. In this example, the DATA segment begins at byte 123,456,789.

3.3.4 $BEGINDATETIME

`$BEGINDATETIME◊yyyy-mm-ddThh:mm:ss[TZD]◊`

`$BEGINDATETIME◊2016-04-13T16:29:19◊`

`$BEGINDATETIME◊2016-04-13T16:29:19Z◊`

`$BEGINDATETIME◊2016-04-13T16:29:19+00:00◊`

`$BEGINDATETIME◇2016-04-13T16:29:19-05:00◇`

`$BEGINDATETIME◇2016-04-13T16:29:19+01:00◇`

`$BEGINDATETIME◇2016-04-13T16:29:19+01◇`

The value of this keyword contains the time stamp at the beginning of data acquisition. If this keyword is present, the value shall be provided in an ISO 8601 [11] compatible *complete extended combined date and time of day representation* format, see ISO 8601:2004, section 4.3.2. Specifically, the date and time of day expression shall be written as a sequence of year, month, day of the month, time designator, hour, minute, second, and zone designator. The zone designator shall be empty if use is made of local time, it shall be the UTC designator `Z` if use is made of the Coordinated Universal Time (UTC) of day and it shall be the difference-component if use is made of local time and the difference from UTC. The minute part of the difference-component of the zone designator may or may not be included. The character `T` shall be used as time designator to indicate the start of the representation of the time of day component in the expression. The hyphen (`-`) and the colon (`:`) shall be used as separators within the date and time of day expressions, respectively.

Historically, the `$DATE` (section 3.3.15), `$BTIM` (section 3.3.6) and `$ETIM` (section 3.3.20) keywords have been used to indicate the beginning and end of data acquisition, but these keywords do not capture the information properly in cases when data are acquired over a period of several days. The `$BEGINDATETIME` and `$ENDDATETIME` (section 3.3.18) keywords have been introduced in FCS 3.2 to address this issue. Those keywords are intended to replace the `$DATE`, `$BTIM` and `$ETIM` keywords in a future version of the standard.

In the first example, the data acquisition began at 16 hours, 29 minutes and 19 seconds local time on April 13, 2016. In the second and third examples, the data acquisition began at 16 hours, 29 minutes and 19 seconds Coordinated Universal Time on April 13, 2016. In the fourth example, the data acquisition began at 16 hours, 29 minutes and 19 seconds US Eastern Standard Time on April 13, 2016. In the fifth and sixth examples, the data

acquisition began at 16 hours, 29 minutes and 19 seconds Central European Time on April 13, 2016.

3.3.5 $BEGINSTEXT

`$BEGINSTEXT◇n◇`

`$BEGINSTEXT◇123456789◇`

The value of this keyword 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, this keyword shall either not be used, or a `0` shall be placed in this keyword value. In this example, the *supplemental* TEXT segment begins at byte 123,456,789.

3.3.6 $BTIM [DEPRECATED]

`$BTIM◇hh:mm:ss[.cc]◇`

`$BTIM◇14:22:10.77◇`

The value of this keyword contains the clock time at the beginning of data acquisition. The value shall be provided in the 24-hour clock format as hours:minutes:seconds.centiseconds. A centisecond is 1/100 of a second. The centisecond part is optional and may or may not be provided. In this example, the data acquisition began at 14 hours, 22 minutes, 10 seconds, and 77/100 of a second.

This keyword is deprecated in favor of `$BEGINDATETIME`, which follows ISO standards. If keyword `$BEGINDATETIME` also exists it shall be consistent.

3.3.7 $BYTEORD [REQUIRED]

`$BYTEORD◇n1,n2,n3,n4◇`

`$BYTEORD◇4,3,2,1◇`

The value of this keyword specifies the endianness of the data, *i.e.*, the byte order used to binary store numeric data values in the DATA segment of 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. The numbers are separated by commas (ASCII code 44). Only two distinct values may be used:

(i) `$BYTEORD◇1,2,3,4◇` – little endian, *i.e.*, the least significant byte is written first, e.g., x86 based personal computers.

(ii) `$BYTEORD◇4,3,2,1◇` – big endian, *i.e.*, the least significant byte is written last, e.g., PowerPC including older Apple Macintosh computers prior to switching to Intel-based architecture.

One of these values shall be used to specify the endianness even if the size of data values is different from 32 (*e.g.,* `$P`*n*`B◇16◇$DATATYPE◇I◇`, `$P`*n*`B◇64◇$DATATYPE◇D◇`).

3.3.8 $CARRIERID

```
$CARRIERID◇string◇
$CARRIERID◇27e029f-3d35-4bda-bda4-72cffeae8b2a◇
$CARRIERID◇001◇
```

The value of the `$CARRIERID` keyword specifies the identifier associated with the carrier which held the sample that was the source of the data set. In the first example, 27e029f-3d35-4bda-bda4-72cffeae8b2a is the carrier identifier; in the second example, 001 is the carrier identifier. Note that it is not required that the identifier be unique.

3.3.9 $CARRIERTYPE

```
$CARRIERTYPE◇string◇
$CARRIERTYPE◇96 well plate standard round bottom◇
$CARRIERTYPE◇40 tube rack◇
```

The value of the $CARRIERTYPE keyword specifies the type of the carrier which held the sample that was the source of the data set. In the first example, 96 well plate standard round bottom is the carrier type; in the second example, 40 tube rack is the carrier type.

3.3.10 $CELLS

```
$CELLS◊string◊
$CELLS◊Normal human peripheral blood◊
$CELLS◊6-level multi-dye beads
```

The value of this keyword specifies the type of cells or other objects measured. In the first example, normal human peripheral blood was used as specimen. In addition to biological cells, this keyword is appropriate for the identification of bead samples and other non-cell analysis samples. The second example illustrates this for a multi peak standardization sample.

3.3.11 $COM

```
$COM◊string◊
$COM◊Incubation time was 47 minutes.◊
```

The value of this keyword may be used to attach a comment to the data set. It shall not to be used as a substitute for other standard keywords. The example shows the use of this keyword to add a brief note about the incubation time for the sample.

3.3.12 $CYT [REQUIRED]

```
$CYT◊string◊
$CYT◊FACScan◊
```

The value of this keyword contains the name of the instrument used to acquire data for the data set. The example shows that FACScan was used.

3.3.13 $CYTSN

$CYTSN◇string◇

$CYTSN◇400E370◇

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

3.3.14 $DATATYPE [REQUIRED]

$DATATYPE◇c◇

$DATATYPE◇F◇

$DATATYPE◇I◇

The value of this keyword describes the default type of data written in the DATA segment of the data set. By default all measurements will be stored as this type unless the optional $P*n*DATATYPE keyword is specified for a given measurement, in which case the value of the $P*n*DATATYPE keyword describes the data type for measurement *n* (see 3.3.41 $P*n*DATATYPE for details). The following values are supported:

(i) $DATATYPE◇I◇ – unsigned binary integer data type,

(ii) $DATATYPE◇F◇ – single precision floating point data type,

(iii) $DATATYPE◇D◇ – double precision floating point data type.

(iv) $DATATYPE◇A◇ – ASCII encoded integer data type. [DEPRECATED]

$DATATYPE◇I◇ indicates that event values are stored as unsigned binary integers. For each measurement in an event, both the maximum length in bits allocated for storage of the measurement value (*i.e.*, value of the $P*n*B keywords) and the actual integer range used by the measurement within that allocation (*i.e.*, value of the $P*n*R keywords) are needed to decode measurement values. The number of bits per measurement is specified by the

Pn$B keyword, value pairs. For example, $P1B◇16◇ specifies that 16 bits are allocated for FCS measurement 1 for each event. The integer range for that measurement is specified by the Pn$R keyword, value pairs. For example, $P1R◇1024◇ specifies that 1024 distinct values ranging from 0 to 1023 (inclusive) may be stored as values of FCS measurement 1. See sections 3.4, 3.3.51 and 3.3.38 on additional information on decoding unsigned binary integers and compression of the data.

$DATATYPE◇F◇ indicates that event values are stored as single precision floating points in the IEEE 754 standard format [10]. In this case, the value of the Pn$B keywords shall be set to 32 for each FCS measurement. The value of the Pn$R keywords should represent the original resolution to acquire the data, which may or may not correspond to the range where data are stored. The value of the Pn$E keywords shall be set to 0,0 for all FCS measurements when $DATATYPE◇F◇ is used. See sections 3.3.38 and 3.3.43 for detailed descriptions of the Pn$B and Pn$E keywords respectively.

$DATATYPE◇D◇ indicates that event values are stored as double precision floating points in the IEEE 754 standard format [10]. In this case, the value of the Pn$B keywords shall be set to 64 for each FCS measurement. The value of the Pn$R keywords should represent the original resolution to acquire the data, which may or may not correspond to the range where data are stored. The value of the Pn$E keywords shall be set to 0,0 for all FCS measurements when $DATATYPE◇D◇ is used. See sections 3.3.38 and 3.3.43 for detailed descriptions of the Pn$B and Pn$E keywords respectively.

$DATATYPE◇A◇ is deprecated in FCS 3.2. While still allowed, the users are discouraged from using $DATATYPE◇A◇ as it will not be supported by future revisions of the FCS standard.

$DATATYPE◇A◇ means that the data are written as ASCII encoded integer values. In this case, the keyword Pn$B 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 measurement 1 would be 9999. Data are stored in a continuous byte stream, with

no delimiters. If the value of the $P*n*B keyword is the * character, e.g., $P1B◇*◇, the data are free format and number of characters per measurement 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.

3.3.15 $DATE [DEPRECATED]

$DATE◇dd-mmm-yyyy◇

$DATE◇05-OCT-2015◇

The value of this keyword specifies the date on which the data from this data set was acquired. If the beginning and end of data acquisition occurred on different dates, the value of the $DATE keyword should correspond to the beginning of acquisition. The value of this keyword shall be stored as dd-mmm-yyyy, which represents a date in the format of *day-month-year*, where the day shall be expressed using a two digit number between 01 and 31 (inclusive, leading zeros shall be present), the month shall be expressed using a 3 character abbreviation, one of JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV or DEC, and the year shall be expressed using a 4 digit number.

This keyword is deprecated in favor of $BEGINDATETIME, which follows ISO standards. If keyword $BEGINDATETIME also exists it shall be consistent.

3.3.16 $ENDANALYSIS

$ENDANALYSIS◇n◇

`$ENDANALYSIS◇123456789◇`

The value of this keyword contains the byte offset from the beginning of the data set to the end of the optional ANALYSIS segment. If there is no ANALYSIS segment, this keyword shall either not be used, or a `0` shall 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 information 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 actual offset information is stored in the value of this keyword only. This keyword is optional as of FCS 3.2. If this keyword is not present, then the position and presence of the ANALYSIS segment are determined by the relevant information the HEADER segment. In this example, the ANALYSIS segment ends at byte 123,456,789.

3.3.17 $ENDDATA [REQUIRED]

`$ENDDATA◇n◇`

`$ENDDATA◇123456789◇`

The value of this keyword 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 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 actual offset information is stored in the value of this keyword only. For zero-length DATA segments (i.e., if the value of the `$TOT` keyword equals to `0`), the `$ENDDATA` keyword may contain any value and such value shall be ignored by FCS consumers. In this example, the DATA segment ends at byte 123,456,789.

3.3.18 $ENDDATETIME

`$ENDDATETIME◇yyyy-mm-ddThh:mm:ss[TZD]◇`

```
$ENDDATETIME♢2016-04-13T16:29:19♢
$ENDDATETIME♢2016-04-13T16:29:19Z♢
$ENDDATETIME♢2016-04-13T16:29:19+00:00♢
$ENDDATETIME♢2016-04-13T16:29:19-05:00♢
$ENDDATETIME♢2016-04-13T16:29:19+01:00♢
$ENDDATETIME♢2016-04-13T16:29:19+01♢
```

The value of this keyword contains the time stamp at the end of data acquisition. If this keyword is present, the value shall be provided in an ISO 8601 [11] compatible *complete extended combined date and time of day representation* format, see ISO 8601:2004, section 4.3.2. Specifically, the date and time of day expression shall be written as a sequence of year, month, day of the month, time designator, hour, minute, second, and zone designator. The zone designator shall be empty if use is made of local time, it shall be the UTC designator `Z` if use is made of the Coordinated Universal Time (UTC) of day and it shall be the difference-component if use is made of local time and the difference from UTC. The minute part of the difference-component of the zone designator may or may not be included. The character `T` shall be used as time designator to indicate the start of the representation of the time of day component in the expression. The hyphen (`-`) and the colon (`:`) shall be used as separators within the date and time of day expressions, respectively.

Historically, the `$DATE` (section 3.3.15), `$BTIM` (section 3.3.6) and `$ETIM` (section 3.3.20) keywords have been used to indicate the beginning and end of data acquisition, but these keywords do not capture the information properly in cases when data are acquired over a period of several days. The `$BEGINDATETIME` (section 3.3.4) and `$ENDDATETIME` keywords have been introduced in FCS 3.2 to address this issue. Those keywords are intended to replace the `$DATE`, `$BTIM` and `$ETIM` keywords in a future version of the standard.

In the first example, the data acquisition ended at 16 hours, 29 minutes and 19 seconds local time on April 13, 2016. In the second and third examples, the data acquisition ended

at 16 hours, 29 minutes and 19 seconds Coordinated Universal Time on April 13, 2016. In the fourth example, the data acquisition ended at 16 hours, 29 minutes and 19 seconds US Eastern Standard Time on April 13, 2016. In the fifth and sixth examples, the data acquisition ended at 16 hours, 29 minutes and 19 seconds Central European Time on April 13, 2016.

3.3.19 $ENDSTEXT

`$ENDSTEXT◇n◇`

`$ENDSTEXT◇123456789◇`

The value of this keyword 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, this keyword shall either not be used, or a `0` shall be placed in this keyword value. In this example, the *supplemental* TEXT segment ends at byte 123,456,789.

3.3.20 $ETIM [DEPRECATED]

`$ETIM◇hh:mm:ss[.cc]◇`

`$ETIM◇14:22:10.77◇`

The value of this keyword contains the clock time at the end of data acquisition. The value shall be provided in the 24-hour clock format as hours:minutes:seconds.centiseconds. A centisecond is 1/100 of a second. The centisecond part is optional and may or may not be provided. In this example, the data acquisition ended at 14 hours, 22 minutes, 10 seconds, and 77/100 of a second.

This keyword is deprecated in favor of `$ENDDATETIME`, which follows ISO standards. If keyword `$ENDDATETIME` also exists it shall be consistent.

3.3.21 $EXP

$EXP◊string◊

$EXP◊A. Smith◊

The value of this keyword contains the name of the person initiating the experiment. In this example, the experiment was initiated by A. Smith. See also 3.3.57 $PROJ for the experiment name and 3.3.32 $OP for the name of the operator.

3.3.22 $FIL

$FIL◊string◊

$FIL◊S01_B06_YDR357C_4-7.fcs◊

The value of this keyword contains the name of the file this data set 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 S01_B06_YDR357C_4-7.fcs.

3.3.23 $FLOWRATE

$FLOWRATE◊string◊

$FLOWRATE◊high◊

The value of this keyword captures the acquisition flow rate setting. It should contain the name of the setting as used by the data acquisition instrumentation or software. Note that this keyword is not intended to capture the actual flow rate as a volume/time value.

3.3.24 $GATING [DEPRECATED]

$GATING◊string◊

$GATING◊R1◊

$GATING◊R1 AND (R2.OR.R3)◊

The value of this keyword specifies the gating conditions under which the data in the data set have been acquired. The conditions shall be set through Boolean operations among regions defined using the `$R`*n*`I` and `$R`*n*`W` keywords (see sections 3.3.58 and 3.3.59). The Boolean operations shall be specified using the `AND`, `OR`, or `NOT` operators that shall be interpreted as in Table 6. The operands of these operators are the regions `R`*n* as defined by the `$R`*n*`I` keywords. Operators shall be separated from operands or other operators by spaces (ASCII code 32) or periods (ASCII code 46). Operator precedence is from left to right unless overridden with parentheses. Only characters '`(`' (ASCII code 40) and '`)`' (ASCII code 41) may be used as parentheses.

In the first example, data were collected using gating region R1. Events with measurement values falling outside R1 were excluded from the data set. In the second example, an event is included in the data set if and only if it is inside R1 and at the same time, it is inside R2 or R3 or both, R2 and R3.

Table 6: Truth table for `AND`, `OR`, and `NOT` Boolean Operations.

R1	**R2**	**R1 AND R2**	**R1 OR R2**	**NOT R1**
FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE	FALSE
TRUE	TRUE	TRUE	TRUE	FALSE

3.3.25 $INST

`$INST◇string◇`

`$INST◇Terry Fox Laboratory, BC Cancer Agency◇`

The value of this keyword specifies the institution or laboratory in which the data were collected. In the example, the data were collected at the Terry Fox Laboratory in BC Cancer Agency.

3.3.26 $LAST_MODIFIED

`$LAST_MODIFIED◇dd-mmm-yyyy hh:mm:ss[.cc]◇`

`$LAST_MODIFIED◇05-SEP-2015 15:22:10.47◇`

The value of this keyword specifies the time stamp of the last modification of the data set. The `$LAST_MODIFIED` keyword may only be present if the `$ORIGINALITY` keyword is also present and the value of the `$ORIGINALITY` is different from `Original`. Any altering of the data file is considered modification; please see also the `$ORIGINALITY` and the `$LAST_MODIFIER` keywords in sections 3.3.33 and 3.3.27, respectively. Note that performing any modifications to FCS data sets generally represents bad practices and the presence of the `$LAST_MODIFIED` keyword is not meant to encourage these practices. The original file should always be kept and any modifications should be saved in a copy of the original file.

The time stamp shall be provided as day-month-year, followed by a space, followed by a 24-hour clock time indication as hours:minutes:seconds.centiseconds. The day shall be expressed using a two digit number between 01 and 31 (inclusive, leading zeros shall be present), the month shall be expressed using a 3 character abbreviation, one of `JAN`, `FEB`, `MAR`, `APR`, `MAY`, `JUN`, `JUL`, `AUG`, `SEP`, `OCT`, `NOV` or `DEC`, and the year shall be expressed using a 4 digit number.

The time part shall be provided in the 24-hour clock format as hours:minutes:seconds.centiseconds. A centisecond is 1/100 of a second. The centisecond part is optional and may or may not be provided. In the example, the data were last modified on September 2, 2015 at 15 hours, 22 minutes, 10 seconds and 47/100 if a second.

3.3.27 $LAST_MODIFIER

`$LAST_MODIFIER◇string◇`

`$LAST_MODIFIER◇Jack Evil◇`

The value of this keyword specifies the name of the person performing the last modification

of the data set. The optional $LAST_MODIFIER keyword may only be present if the value of the $ORIGINALITY keyword is different from Original. Any altering of the data file is consider modification; please see also the $ORIGINALITY and the $LAST_MODIFIED keywords in sections 3.3.33 and 3.3.26, respectively. 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 practices. 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.

3.3.28 $LOCATIONID

$LOCATIONID◇string◇

$LOCATIONID◇E8◇

$LOCATIONID◇A07◇

The value of the $LOCATIONID keyword specifies the location on the carrier of the sample that was the source of the data set. In the first example, E8 is the location identifier. This may be a well located in the fifth row and eighth column of a 40 tube rack in which the manufacturer labels the rows A - E and the numbers the columns 1 - 8. In the second example, A07 is the location identifier. This could be the well located in the first row and seventh column of a 96 well plate in which the manufacturer labels the rows A - H and the numbers the columns 01 - 12.

3.3.29 $LOST

$LOST◇n◇

$LOST◇457◇

The value of this keyword specifies the number of events lost during data acquisition because the computer was busy with other tasks. In this example, 457 events were so lost. See also

3.3.1 $ABRT for a count of events lost in the signal processing electronics.

3.3.30 $MODE [DEPRECATED]

$MODE◇c◇

$MODE◇L◇

The value of this keyword specifies the mode in which the data were stored in this data set. If this keyword is used, the value of it shall always be set to L indicating *list mode* data. In list mode, for each event, the value of each FCS measurement is stored in a fixed order, usually defined by the instrument manufacturer. Other modes, such as correlated and uncorrelated multivariate histograms (supported by previous versions of the FCS file format), are no longer supported by FCS version 3.2. This keyword used to be required in previous versions of FCS. In this version, it has been deprecated and will be removed in a future version.

3.3.31 $NEXTDATA [REQUIRED]

$NEXTDATA◇n◇

$NEXTDATA◇0◇

$NEXTDATA◇123456◇

The value of the $NEXTDATA keyword specifies 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, or 0 if this is the final or the only data set in the FCS file. As shown in Figure 1, an FCS 3.2 data file may consist of one or more data sets; however, the usage of multiple data sets within a single data file is deprecated unless these data sets are derived one from another. If multiple data sets are used then each data set shall contain all required keyword, value pairs.

3.3.32 $OP

$OP◇string◇

$OP◇John Smith◇

The value of the $OP keyword specifies the name of the operator of the instrument (*e.g.*, flow cytometer) that was used to acquire this data set. In this example, John Smith is the name of the person who operated the instrument. See also 3.3.21 $EXP for the name of the person initiating the experiment and 3.3.57 $PROJ for the name of the experiment.

3.3.33 $ORIGINALITY

$ORIGINALITY◇string◇

$ORIGINALITY◇Original◇

The value of the $ORIGINALITY keyword specifies whether the FCS data set has been modified or is original as acquired by the instrument. See also the description of the $LAST_MODIFIER and the $LAST_MODIFIED keywords in sections 3.3.27 and 3.3.26, respectively. If the $ORIGINALITY keyword is present, then one of the following values shall be used:

(i) $ORIGINALITY◇Original◇ – original data set,

(ii) $ORIGINALITY◇NonDataModified◇ – no modifications in the DATA segment,

(iii) $ORIGINALITY◇Appended◇ – additions only in the DATA segment,

(iv) $ORIGINALITY◇DataModified◇ – modified data set.

$ORIGINALITY◇Original◇ indicates that the full data set is as acquired or originally created by an instrument or software. No part of the data set 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 data set.

`$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) FCS measurements (*e.g.*, classification results, FL2-A/FL2-H ratio, etc.) or adding new events. The TEXT segment shall be modified to reflect changes in the DATA segment.

`$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.

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.

3.3.34 $PAR [REQUIRED]

`$PAR◇n◇`

`$PAR◇12◇`

The value of the `$PAR` keyword specifies the total number of FCS measurements stored in each event in the data set. In this example, data for 12 FCS measurements are stored for each event.

3.3.35 $PLATEID [DEPRECATED]

`$PLATEID◇string◇`

`$PLATEID◇27e029f-3d35-4bda-bda4-72cffeae8b2a◇`

The value of the $PLATEID keyword specifies the unique identifier associated with the plate from which a well was the source of the data set. The $PLATEID keyword has been deprecated in FCS 3.2; keywords $CARRIERID (section 3.3.8), $CARRIERTYPE (section 3.3.9) and $LOCATIONID (section 3.3.28) should be used to identify the carrier holding the sample that was the source of a data set.

3.3.36 $PLATENAME [DEPRECATED]

$PLATENAME◊string◊

$PLATENAME◊Plate1 96 Well - U bottom◊

The value of the $PLATENAME keyword specifies the name of the plate from which a well was the source of the data set. The $PLATENAME keyword has been deprecated in FCS 3.2; keywords $CARRIERID (section 3.3.8), $CARRIERTYPE (section 3.3.9) and $LOCATIONID (section 3.3.28) should be used to identify the carrier holding the sample that was the source of a data set.

3.3.37 $PnANALYTE

$PnANALYTE◊string◊

$P1ANALYTE◊CD4◊

The values of the Pn$ANALYTE keywords specify the target molecule or process for FCS measurement n. In the example, FCS measurement 1 is labeled as CD4.

It has been common practice to use the Pn$N and Pn$S keywords to capture a combination of the detector name, fluorochrome, measurement type and the analyte or marker being measured. The way this was encoded was not standardized and varied between systems and users. Therefore, non-reliable heuristics were required in order to extract those details. The PnDET, PnTAG, Pn$ANALYTE, Pn$TYPE and Pn$FEATURE keywords have been introduced in order to facilitate simple robust extraction of the detector name, fluorochrome, measurement

target, measurement type and signal feature (e.g., Area) associated with each of the FCS measurements

$P*n*ANALYTE is properly applicable to compensated measurements and other measurements that will not be used in compensation. However, in FCS 3.2 $P*n*TAG and $P*n*ANALYTE may be assigned to the primary measurement corresponding to a reagent's TAG, and, if so, shall be carried over to the appropriate compensated measurement.

3.3.38 $PnB [REQUIRED]

$PnB◇n1◇

$P3B◇16◇

The values of the $P*n*B keywords specify the number of bits allocated for storage of values of FCS measurement number n.

For $DATATYPE◇F◇, the value of all $P*n*B keywords shall be set to 32 indicting 32 bits per value.

For $DATATYPE◇D◇, the value of all $P*n*B keywords shall be set to 64 indicting 64 bits per value.

For $DATATYPE◇I◇, the value of the $P*n*B keywords specify the number of bits allocated for storage of values of FCS measurement number n. These keywords are used in conjunction with the $P*n*R (range) keywords (section 3.3.51) to determine how the data are stored. Appropriate bit masks shall be used while reading data with more bits allocated than required by the specified range. Usage of values divisible by 8 is recommended for interoperability and performance reasons. Values that are not divisible by 8 are deprecated as of FCS 3.2, but they enable tight bit packing of events. For example, if storage was specified by $P*n*B◇10◇$P*n*R◇1024◇ for every FCS measurement in an event, then fewer bits would be wasted in storing each event. However, $P*n*B◇16◇$P*n*R◇1024◇ is typically be used for 10 bit data.

In this example, data values for FCS measurement 3 would be stored as two bytes (16 bits). This keyword value is used in conjunction with the `$P3R` keyword value to determine how the data are stored and what bit mask shall be applied. A flow cytometer with 10-bit analog-to-digital converters (ADCs) would typically use `$P3R◇1024◇`. This indicates that a 10-bit number is stored in the 16-bit space allocated by `$P3B◇16◇` leaving 6 empty bits per value. Implementors shall use a bit mask when reading these values to insure that erroneous values are not read from the unused bits. If `$P`*n*`R` is not a power of two, the next higher power of 2 shall be used to determine the bit mask and any information in the bits above the next higher power of 2 should be ignored.

3.3.39 $PnCALIBRATION

`$PnCALIBRATION◇f1[,f2],string◇`

`$P1CALIBRATION◇1.234,MESF◇`

`$P2CALIBRATION◇1.234,100,MESF◇`

The values of the `$P`*n*`CALIBRATION` keywords specify the conversion of arbitrary signal units, recorded as FCS measurement values, to some well defined unit, for example, mean equivalent soluble fluorochrome (MESF) or antibody molecules. The keyword value shall contain the following comma-separated components:

- `f1` - a positive floating point value specifying the multiplier (slope) of the calibration;
- `f2` - floating point value specifying the offset (intercept) of the calibration (optional, considered to be 0 if not present);
- `string` - a UTF-8 [9] encoded character string capturing the name of the units corresponding to calibrated values.

If both `f1` and `f2` are present, then the calibrated values shall be calculated from FCS measurement values v as $v_c = v{*}$`f1` + `f2`. If `f2` is not present, then $v_c = v{*}$`f1`. This

transformation shall be applied to the scale data value, *i.e.*, after any "antilog" (`$P`*n*`E`) or gain (`$P`*n*`G`) scaling has been performed. This calibration may be applied to compensated data only when the spillover matrix is normalized to unity on the diagonal.

In the example, FCS measurement 1 has 1.234 MESF per scale unit. For measurement 2, you also need to add 100 to obtain a value in MESF, e.g., 50 scale units correspond to 161.7 MESF.

3.3.40 $PnD

`$PnD◇string,f1,f2◇`

`$PnD◇Linear,0,1023◇`

`$PnD◇Logarithmic,4,0.1◇`

The values of the `$P`*n*`D` keywords specify the recommended visualization for FCS measurement *n*. Software tools displaying the data may or may not follow the suggested visualization scale. Typically, the `$P`*n*`D` values would be used to set reasonable defaults, but the end user may still be able to change the scale to visualize the FCS measurement.

The value of the `string` part shall be one of `Linear` or `Logarithmic`; it encodes the type of display transformation, which should map to the transformation that was applied to the data during acquisition. Note that this may be different from the scale that the data are saved on (several modern instruments save all data on a linear scale with `$P`*n*`E◇0,0◇` so that the original transformation is not maintained without use of the `$P`*n*`D` keyword). Each of the transformations has a different measurement list (floating point numbers `f1` and `f2`) that specifies how to construct the transformation.

- Linear scale: `$P`*n*`D◇Linear,f1,f2◇`. Data should be displayed as linear scale. Note that the `f1` and `f2` measurement values are in "scale" units, not "channel" units (see below for details). The `f1` and `f2` floating point values shall be used as follows:
  - `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,0,1023◇` specifies a linear display ranging from 0 to 1023 (scale value).

Example `$P5D◇Linear,100,200◇` specifies that the values to be displayed run from 100 to 200.

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

Example `$P3B◇8◇$P3R◇256◇$P3G◇4◇$P3E◇0,0◇$P3D◇Linear,0,32◇`

This is a linear measurement with channel values going from 0 to 255. Taking the gain into account, 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.

Example `$P4B◇16◇$P4R◇1024◇$P4E◇4,1◇$P4D◇Linear,0,1000◇`

The `$P4D` 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.

- Logarithmic scale: `$P`*n*`D◇Logarithmic,f1,f2◇`. Data should be displayed with logarithmic scaling. Note that the `f1` is a pure number and `f2` measurement values are in "scale" units, not "channel" units (see below for details). The `f1` and `f2` floating point values shall be used as follows:

  - `f1`: The number of decades to display
  - `f2`: The scale value corresponding to the left edge of the display

Example `$P3D◇Logarithmic,4,0.1◇` specifies a linear display ranging from 0.1 to 1000 (scale value), which is 4 decades of display width.

Example `$P3D◇Logarithmic,5,0.1◇` specifies a linear display ranging from 0.01 to 1000 (scale value), which is 5 decades of display width.

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

Example `$P4B◇16◇$P4R◇1024◇$P4E◇4,1◇$P4D◇Logarithmic,3,1◇`

The `$P4D` 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 ($1024 \times 3/4$).

Please refer also to [6] for additional guidance on how to use the suggested visualization scale.

3.3.41 $PnDATATYPE

`$PnDATATYPE◇c◇`

`$PnDATATYPE◇I◇`

`$PnDATATYPE◇F◇`

`$PnDATATYPE◇D◇`

The value of this keyword describes the type of data for FCS measurement n written in the DATA segment of the data set. If specified, this keyword indicates that FCS measurement n is stored as a different datatype than the default datatype specified by the `$DATATYPE` keyword. For example, this allows an application to better support FCS measurements that are intrinsically integers (such as Time or indices), while storing the remaining FCS measurements as single precision floating point values. This is an issue for measurements that represent indices, for example, because single precision floating point data only have 24 bits of precision, so the maximum number of positive integers that can be uniquely represented is 2^{24}, or 16,777,216, whereas the maximum number of integers that can be uniquely represented by an unsigned integer is $2^{32} - 1$, or 4,294,967,295 .

Another example would be to allow one or more measurements to be represented as a double precision floating point value while the remaining measurements are represented by single precision floating point values.

The following values are supported:

(i) `$PnDATATYPE◇I◇` – unsigned binary integer data type,

(ii) `$PnDATATYPE◇F◇` – single precision floating point data type,

(iii) `$PnDATATYPE◇D◇` – double precision floating point data type.

Note that datatype ◇A◇, while supported (but deprecated) for the `$DATATYPE` keyword, is not supported for this keyword.

3.3.42 $PnDET

`$PnDET◇string◇`

`$P1DET◇FSC◇`

`$P2DET◇FL1◇`

The values of the `$P`*n*`DET` keywords specify detector names for FCS measurement *n*. In the example, FCS measurement 1 is measured on a forward scatter detector and FCS measurement 2 on the FL1 detector.

It has been common practice to use the `$P`*n*`N` and `$P`*n*`S` keywords to capture a combination of the detector name, fluorochrome, measurement type and the label or marker being measured. The way this was encoded was not standardized and varied between systems and users. Therefore, non-reliable heuristics were required in order to extract those details. The `$P`*n*`DET`, `$P`*n*`TAG`, `$P`*n*`ANALYTE`, `$P`*n*`TYPE` and `$P`*n*`FEATURE` keywords have been introduced in order to facilitate simple robust extraction of the detector name, fluorochrome, measurement target, measurement type and signal feature (e.g., Area) associated with each of the FCS measurements.

3.3.43 $PnE [REQUIRED]

`$PnE◇f1,f2◇`

`$P3E◇4,0.01◇`

The values of the `$P`*n*`E` keywords specify whether FCS measurement number *n* is stored on linear or logarithmic scale, and it includes details about the logarithmic amplification if logarithmic scale is used. When linear scale is used, `$P`*n*`E◇0,0◇` shall be entered. Linear scale shall always be used on time measurements. If either the single precision or double precision floating point data type is used (either `$DATATYPE◇F◇` or `$DATATYPE◇D◇`), then all FCS measurements shall be stored as linear with `$P`*n*`E◇0,0◇` specified. When logarithmic scale is used, the value of `f1` specifies the number of logarithmic decades and `f2` represents the linear value that would have been obtained for a signal with a log value of 0. In the example above, the data for FCS measurement 3 were collected using a four-decade logarithmic amplifier and the 0 channel represents the linear value, 0.01. Note that values of `f1` and `f2` shall either be both zero or both positive numbers. Especially, `$P`*n*`E◇f1,0◇` with `f1` > 0 is not a valid entry and shall not be used. If this entry is found in an FCS file, it is recommended to handle it as `$P`*n*`E◇f1,1◇`.

Explanation: Entries such as `$P`*n*`E◇4,0◇` have never been correct. Unfortunately, the lack of clear explanation in older versions of this specification made these widely used. For `$P`*n*`E◇f1,f2◇`, `f1` > 0, `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, `$P`*n*`E◇4,1◇` indicates 4 decades log reaching from 1 to 10000; `$P`*n*`E◇5,0.01◇` indicates 5 decades log reaching from 0.01 to 1000.

Converting from channel values on logarithmic scale to linear scale values: For `$P`*n*`R◇r◇`, `r`> 0, `$P`*n*`E◇f1,f2◇`, `f1`> 0, `f2`> 0, *n* is a logarithmic measurement with channel values

going from 0 to r-1, and scale values going from f2 to $\texttt{f2} \times 10^{\texttt{f1}}$. A channel value x_c can be converted to a scale value x_s as $x_s = 10^{(\texttt{f1} \times x_c / \texttt{r})} \times \texttt{f2}$. Examples of converting from channel values on logarithmic scale to linear scale values are shown in Example 4.

$P1R◇1024◇$P1E◇4,1◇
These keyword, value pairs specify a logarithmic FCS measurement with channel values going from 0 to 1023, and scale values going from 1 to approximately 10000. A channel value x_c can be converted to a scale value x_s as $x_s = 10^{(4 \times x_c / 1024)} \times 1$.

$P2R◇256◇$P2E◇4.5,0.1◇
These keyword, value pairs specify a logarithmic FCS measurement with channel values going from 0 to 255, and scale values going from 0.1 to approximately $10^{3.5}$ (≈ 3162). A channel value x_c can be converted to a scale value x_s as $x_s = 10^{(4.5 \times x_c / 256)} \times 0.1$.

Example 4: Converting log channel to scale values

3.3.44 $PnF

$PnF◇string◇

$P2F◇515,545◇

The value of the Pn$F keyword specifies the optical filter that was used for the light reaching the detector for FCS measurement n. For backwards compatibility, the value of this keyword may be any string values, but it is recommended that the filter information is encoded as Pn$F◇$n_1$,$n_2$◇ where n_1 is the lowest wavelength in nm passed on by the filter, and n_2 is the highest wavelength in nm passed on by the filter. This example shows that a 530/30 band pass optical filter was used for the second FCS measurement. Note that historically, the format of the filter specification was not standardized and therefore, you may encounter different entries including $P2F◇530/30◇. The Pn$F keywords should not be supplied for compensated FCS measurements.

3.3.45 $PnFEATURE

$PnFEATURE◇string◇

$P2FEATURE◊Area◊

$P2FEATURE◊Height◊

$P2FEATURE◊Width◊

The value of the $P*n*FEATURE keyword specifies a feature of the signal from the detector that was sampled for FCS measurement *n*. The values Area, Height and Width shall be used when appropriate.

It has been common practice to use the $P*n*N and $P*n*S keywords to capture a combination of the detector name, fluorochrome, measurement type and the analyte or marker being measured. The way this was encoded was not standardized and varied between systems and users. Therefore, non-reliable heuristics were required in order to extract those details. The $P*n*DET, $P*n*TAG, $P*n*ANALYTE, $P*n*TYPE and $P*n*FEATURE keywords have been introduced in order to facilitate simple robust extraction of the detector name, fluorochrome, measurement target, measurement type and signal feature (e.g., Area) associated with each of the FCS measurements.

3.3.46 $PnG

$PnG◊f◊

$P2G◊10.0◊

The values of the $P*n*G keywords specify the gain that was used to amplify the signal for FCS measurement *n*. The example shows that FCS measurement 2 was amplified 10.0-fold before digitization. This keyword shall be used only for integer data, i.e., only with $DATATYPE◊I◊ and not with F or D. This means that channel values and scale values are the same for floating point data. Gain prior to logarithmic transform simply shifts $P*n*E *f2*, the minimum signal, while gain afterwards changes the number of decades *f1*. Therefore $P*n*G◊ gain shall not be used in combination with logarithmic amplification, *i.e.*, shall not be used together with $P*n*E different from $P*n*E◊0,0◊. The $P*n*G keywords should not be supplied for time

or compensated FCS measurements.

Converting linearly amplified data from channel values to scale values: For `$P`*n*`G`◇`g`◇, `g`> 0, `$P`*n*`E`◇`0,0`◇, a channel value x_c can be converted to a scale value x_s as $x_s = x_c/g$. An example of converting from channel values on linear scale to scale values is shown in Example 5.

`$P1R`◇`1024`◇`$P1E`◇`0,0`◇`$P1G`◇`8.0`◇
These keyword, value pairs specify an FCS measurement with channel values going from 0 to 1023, and scale values from 0 to approximately 128. In this case, a channel value x_c can be converted to a scale value x_s as $x_s = x_c/8$.

Example 5: Converting linear channel to scale values

3.3.47 $PnL

`$PnL`◇`n1[,n2,n3,...]`◇

`$P1L`◇`488`◇

`$P3L`◇`560,625`◇

The values of the `$P`*n*`L` keywords specify the excitation wavelength(s) in nm (nanometers) for FCS measurement n. In the first example, the excitation wavelength was 488 nm for FCS measurement number 1. Typically, a single wavelength is related to each of the FCS measurements; however, some of the modern instruments allow for multiple lasers with different wavelengths related to a single measurement. In the second example, two co-axial lasers with different wavelengths (560 nm and 625 nm, respectively) have been used in a way that the result of their excitation activity was detected using the detector related to FCS measurement number 3. The `$P`*n*`L` keywords should not be supplied for compensated FCS measurements.

3.3.48 $PnN [REQUIRED]

`$PnN`◇`string`◇

$P3N◊FL1-H◊

$P4N◊PE-A◊

$P6N◊B530-A◊

$P14N◊La139◊

$P1N◊EV◊

The values of the $P*n*N keywords specify the short name of FCS measurement n. These names shall be unique, *i.e.*, each name (value of a $P*n*N keyword) may occur only once in the data set. The character ',' (comma, ASCII code 44) is not allowed to be part of the value of this keyword (comma in the short name would introduce a conflict with other keywords, such as $TR (section 3.3.66) or $SPILLOVER (section 3.3.61)).

The value Time (case sensitive, see discussion in section 3.3.64) shall be used as the short measurement name in order to indicate a time measurement with steps provided in the $TIMESTEP keyword (section 3.3.64). Use of the (new in FCS 3.2) keywords $P*n*DET, $P*n*TAG, $P*n*ANALYTE, $P*n*TYPE and $P*n*FEATURE can clarify measurement decriptions and help avoid confusing choices for $P*n*N measurement names (e.g. $P5N◊APC-A◊ when the actual dye in use is Alexa Fluor 647).

As shown in the first example, measurement number 3 has a short name of FL1-H, indicating pulse height (as indicated by the -H part of the name) on the first fluorescence detector, a common historical pattern. In the second example, the short name of FCS measurement number 4 is PE-A indicating that the detector is optimized for phycoerythrin and the pulse area is measured (as indicated by the -A part of the name), another common pattern historically. However, as the number of dyes and detectors has greatly expanded these patterns have become less appropriate. In the third example, the name of the FCS measurement number 6 indicates that a blue laser (B, excitation wavelength between 481 and 520 nm) was used in conjunction with an optical filter that has a center of its emission band around 530 nm, and the area of the signal peak was captured in FCS measurement number

6. The fourth example shows metal isotope La 139 used in FCS measurement number 14. Names of metal isotopes are commonly used as FCS measurement names in mass cytometry (CyTOF) data sets. Finally, the fifth example shows that electronic volume (EV) is captured in FCS measurement 1. Note that those 5 examples come from 5 different instruments; FCS measurement names like those wouldn't typically be present in a single data set.

It has been common practice to use the `$P`*n*`N` and `$P`*n*`S` keywords to capture a combination of the detector name, fluorochrome, measurement type and the analyte or marker being measured. The way this was encoded was not standardized and varied between systems and users. Therefore, non-reliable heuristics were required in order to extract those details. The `$P`*n*`DET`, `$P`*n*`TAG`, `$P`*n*`ANALYTE`, `$P`*n*`TYPE` and `$P`*n*`FEATURE` keywords have been introduced in order to facilitate simple robust extraction of the detector name, fluorochrome, measurement target, measurement type and signal feature (e.g., Area) associated with each of the FCS measurements.

3.3.49 $PnO

`$PnO`◇`n1`◇

`$P2O`◇`200`◇

The values of the `$P`*n*`O` keywords specify the excitation power, `n1`, in milliwatts for the light source associated with the measurements for FCS measurement number *n*. In the example, 200 mW was used to produce the signal associated with FCS measurement number 2. The `$P`*n*`O` keywords should not be supplied for compensated FCS measurements.

3.3.50 $PnP [DEPRECATED]

`$PnP`◇`n1`◇

`$P4P`◇`50`◇

The values of the `$P`*n*`P` keywords specify the amount of light collected by the detector for

FCS measurement 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 FCS measurement 4.

As of FCS 3.2, the PnP keywords have been deprecated as those values are not very meaningful and this keyword has not been widely used in the past. The PnP keywords should not be supplied for compensated FCS measurements.

3.3.51 $PnR [REQUIRED]

$PnR◇n1◇

$P2R◇1024◇

For $DATATYPE◇I◇, the values of the Pn$R keywords specify the maximum range, n1, of FCS measurement n, which typically corresponds to the ADC range. In this case, the channel values of FCS measurement n can range from 0 to $n1 - 1$. For $DATATYPE◇I◇, powers of 2 are preferred. The values of the Pn$R keywords also indicate the bit mask that shall be used when reading values. The bit mask shall be set so that only bits allowing for values up to $n1 - 1$ are considered when reading channel values of FCS measurement n. When the value is not a power of 2 the mask shall be computed from the next higher power of 2. See also the description of the Pn$B keywords (section 3.3.38) for more information.

For $DATATYPE◇F◇ and $DATATYPE◇D◇, the values of the Pn$R keywords specify the maximum expected or maximum valid range, *n1*, of FCS measurement n. However, the measurement values stored in the data set may exceed this range on both sides of the zero to *n1* interval. Specifically, there may be negative values as well as values greater than or equal to $n1$. The appropriate range for dimensions resulting from compensation or other data transformation may not correspond to $n1$ values. No bit mask shall be applied when reading floating point ($DATATYPE◇F◇ or $DATATYPE◇D◇) data.

3.3.52 $PnS

$PnS◊string◊

$P1S◊CD161-B525-A◊

$P1S◊CD161-A◊

$P1S◊CD161-B525◊

$P1S◊CD161-PE◊

The values of the $P*n*S keywords specify the long name (full name) of FCS measurement *n*. In current usage, $P*n*S is often the keyword that includes reagent or binding specificity information. It often also includes fluorochrome, tag or detector information. This name may be used as an axis label in a plot of FCS measurement *n*. If possible, the *fluorochrome* (*dye*) should be selected from the "Preferred Short Names" of the ISAC Probe Tag Dictionary [12].

Unlike $P*n*N, $P*n*S values may include commas and are not required to be unique among the FCS measurements in the data set.

It has been common practice to use the $P*n*N and $P*n*S keywords to capture a combination of the detector name, fluorochrome, measurement type and the analyte or marker being measured. The way this was encoded was not standardized and varied between systems and users. Therefore, non-reliable heuristics were required in order to extract those details. The $P*n*DET, $P*n*TAG, $P*n*ANALYTE, $P*n*TYPE and $P*n*FEATURE keywords have been introduced in order to facilitate simple robust extraction of the detector name, fluorochrome, measurement target, measurement type and signal feature (e.g., Area) associated with each of the FCS measurements.

3.3.53 $PnT

$PnT◊string◊

$P2T◊PMT9524◊

The values of the $P*n*T keywords specify the detector type for FCS measurement number

n. In the provided example, photomultiplier tube (PMT) of type 9524 was used for FCS measurement 2. The `$P`*n*`T` keywords should not be supplied for compensated FCS measurements.

3.3.54 $PnTAG

`$PnTAG`◇`string`◇

`$P3TAG`◇`AF633`◇

The values of the `$P`*n*`TAG` keywords specify the dye (fluorochrome) or isotope being used for FCS measurement *n*. In the provided example, *Alexa Fluor 633* was used for FCS measurement number 3. If present, the values of these keywords should be selected from the "Preferred Short Names" of the ISAC Probe Tag Dictionary [12].

`$P`*n*`TAG` is properly applicable to compensated measurements and other measurements that will not be used in compensation. However, in FCS 3.2 `$P`*n*`TAG` and `$P`*n*`ANALYTE` may be assigned to the primary measurement corresponding to a reagent's TAG, and, if so, shall be carried over to the appropriate compensated measurement.

3.3.55 $PnTYPE

`$PnTYPE`◇`string`◇

`$P1TYPE`◇`Forward Scatter`◇

`$P2TYPE`◇`Raw Fluorescence`◇

`$P3TYPE`◇`Mass`◇

`$P6TYPE`◇`Time`◇

`$P11TYPE`◇`Index`◇

`$P12TYPE`◇`Classification`◇

The value of the `$P`*n*`TYPE` keyword formally specifies the type of measurement *n*. The values `Forward Scatter`, `Side Scatter`, `Raw Fluorescence`, `Unmixed Fluorescence`, `Mass`,

`Time`, `Electronic Volume`, `Classification` and `Index` shall be used when appropriate.

In the provided example, FCS measurement 1 captures forward scatter, FCS measurement 2 capturess a conventional fluorescence-based signal, FCS measurement 3 captures mass cytometry values, FCS measurement 6 captures a time measurement, FCS measurement 11 captures an index sorting event, FCS measurement 12 captures a sort decision or clustering results. Please note that this example has been created to demonstrate all these different FCS measurement types although it makes little sense to have those all present in a single FCS data set.

It has been common practice to use the `$P`*n*`N` and `$P`*n*`S` keywords to capture a combination of the detector name, fluorochrome, measurement type and the analyte or marker being measured. The way this was encoded was not standardized and varied between systems and users. Therefore, non-reliable heuristics were required in order to extract those details. The `$P`*n*`DET`, `$P`*n*`TAG`, `$P`*n*`ANALYTE`, `$P`*n*`TYPE` and `$P`*n*`FEATURE` keywords have been introduced in order to facilitate simple robust extraction of the detector name, fluorochrome, measurement target, measurement type and signal feature (e.g., area) associated with each of the FCS measurements.

3.3.56 $PnV

`$PnV◊f◊`

`$P2V◊445.5◊`

This keyword specifies the detector control setting. When the detector is a photomultiplier tube (PMT) with applied voltage specified in volts, the values of the `$P`*n*`V` keywords shall specify the detector voltage. In the provided example, the detector for FCS measurement 2 was set to 445.5 volts.

Instruments using other detector technologies, such as avalanche photo diodes (APDs), as well as some PMT-based instruments may have detector control settings that are not

specified as a voltage. In such cases, the detector control setting information shall be stored in the Pn$V keyword, and the meaning of this keyword is broadened to apply to this situation.

Pn$V keywords shall not be supplied for compensated FCS measurements.

3.3.57 $PROJ

$PROJ◇string◇

$PROJ◇AML patient study◇

In prior versions, the value of the $PROJ keyword specifies the name of the project, but there was no clear place for the name of an experiment. With this version, this is clarified to say the keyword should be used to specify the name of the experiment, "AML patient study" in the provided example. See also 3.3.21 $EXP and 3.3.32 $OP for the names of the experimenter and operator respectively.

3.3.58 $RnI [DEPRECATED]

$RnI◇string1[,string2]◇

$R3I◇P2,P4◇

$R2I◇P3◇

The values of the $RnI keywords associate a gating region number n with one or two FCS measurements as indicated by string1 (for one FCS measurement) or string1,string2 (for two FCS measurements), respectively. The string(s) shall be in the form of Pi where i is an ASCII encoded integer number between 1 and the number of FCS measurements (*i.e.*, the value of the $PAR keyword, section 3.3.34). The string Pi references FCS measurement number i. In the first example, gating region number 3 is associated with FCS measurements number 2 and 4. The $RnW keyword (section 3.3.59) specifies the shape of the related gating region. In the second example, gating region number 2 is associated with FCS measurement number 3.

3.3.59 $RnW [DEPRECATED]

`$RnW◊f1,f2[;f3,f4;...]◊`

`$R2W◊345.5,366.8◊`

`$R3W◊310,205;515,304;480,615;240,514;354,542◊`

The values of the `$RnW` keywords specify the window settings for gating region number n. This window setting is useful only if the corresponding `$RnI` keyword (section 3.3.58) is also specified. The `$GATING` keyword (section 3.3.24) specifies the way windows defined by the `$RnW` keywords were used.

If the `$RnI` keyword has a single value only, then `f1` and `f2` specify the inclusive lower and upper bounds for the window in a univariate histogram of FCS measurement specified by the value of the `$RnI` keyword. An example of this case is provided in example 6.

`$R2I◊P3◊$R2W◊345.5,366.8◊`
These keyword, value pairs specify that gating region number 2 is associated with FCS measurement number 3. The gated events for FCS measurement 3 must range between channel values 345.5 and 366.8 inclusive.

Example 6: 1-dimensional gating region

If the corresponding `$RnI` keyword value contains 2 strings specifying 2 FCS measurements, then the window is specified by a polygon in a bivariate plot of those 2 measurements. The coordinates of this polygon (in channel values) are encoded in the `$RnW` keyword. The `f1`,`f2` pair specifies the x and y coordinates of the first vertex in the polygon. The coordinates of the next vertex are specified by the `f3`,`f4` pair, which is separated by a semicolon (`;`) character (ASCII code 59) from the coordinates of the previous point. The coordinates of the next vertex follow analogically with vertices separated from one another by semicolon character. The polygon may contain any number of vertices; the first and last vertex are assumed to be connected. An example of this case is provided in example 7.

```
$R1I◇P2,P3◇$R1W◇310,205;515,304;480,615;240,514;354,542◇
```
These keyword, value pairs specify that gating region 1 is defined in FCS measurements 2 and 3, and that the region 1 window is a 5-sided polygon in this 2-dimensional space.

Example 7: 2-dimensional gating region

3.3.60 $SMNO

```
$SMNO◇string◇
```

```
$SMNO◇7874A5◇
```

The value of the $SMNO keyword specifies an identifier of the specimen (sample). It may include any vendor or user specific identification. Generally, multiple specimens may be related to the same source (see also $SRC keywords in section 3.3.62). See also $CARRIERID, $CARRIERTYPE, and $LOCATIONID keywords (sections 3.3.8, 3.3.9 and 3.3.28, respectively) for carrier identification. In the provided example, the specimen is identified as 7874A5.

3.3.61 $SPILLOVER

```
$SPILLOVER◇n,string1,string2,...,f1,f2,f3,f4,...◇
```

```
$SPILLOVER◇2,FL1-A,FL2-A,1.0,0.1,0.03,1.0◇
```

The value of the $SPILLOVER keyword specifies the information necessary to carry out traditional fluorescence compensation in the form of a spectral matrix. The $SPILLOVER keyword shall not be used in data sets containing spectral data. The matrix gives the spillover that is recommended to be used to calculate fluorescence compensated data for post acquisition analysis from uncompensated data stored in the file. The spectral matrix references FCS measurements that are stored uncompensated in the data set. Only a single matrix may be provided for the data set. Please refer to [6] for guidance on how to use a single matrix when multiple measurement types, such as the height, area and width of a signal are involved. Note that compensation may also be provided separately, for example using the Gating-ML standard [13].

The value of the $SPILLOVER keyword shall contain 1 positive integer value, 2 or more string values, and 4 or more floating point values, all separated by commas (ASCII code 44). Values of $SPILLOVER◇$n$,$\texttt{string}_1$,$\texttt{string}_2$,...,$\texttt{string}_n$,$\texttt{S}_{11}$,$\texttt{S}_{12}$,...,$\texttt{S}_{1n}$,$\texttt{S}_{21}$,$\texttt{S}_{22}$,...,$\texttt{S}_{nn}$◇ shall be interpreted as follows:

- n (a single positive integer value): n represents the number of FCS measurements in the spillover matrix.
- $\texttt{string}_i$ ($i \in 1..n$, *i.e.*, n character strings): $\texttt{string}_i$ represents the FCS measurement name corresponding to the i-th measurement in the spillover matrix. These FCS measurement names shall correspond exactly to short measurement names as specified by the Pn$N keywords (section 3.3.48), including any suffixes, such as -A, -H, -W, etc.
- $\texttt{S}_{ij}$ ($i \in 1..n$, $j \in 1..n$, *i.e.*, $n \times n$ floating point values): $\texttt{S}_{ij}$ represents the spillover coefficient from FCS measurement i to FCS measurement j. The spillover coefficient from FCS measurement i to FCS measurement 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 channel dye.

This formulation allows for a subset of the collected FCS measurements to be included in the spillover matrix. The number of FCS measurements in the matrix, n, is specified by the first element. This shall be greater than 1 and less than or equal to the number of FCS measurements in the file (*i.e.*,for $PAR◇$m$◇, $n \leq m$). The matrix will have a size of $n \times n$. The total number of comma-separated elements in the $SPILLOVER keyword value shall correspond to $1 + n + n \times n$; the first value is a positive integer value (n, the number of FCS measurements in the matrix), the next n values are strings of characters referencing FCS measurements by their short names (Pn$N keyword values, section 3.3.48), and the remaining $n \times n$ floating point values represent their spillover matrix row by row. The FCS measurements may be present in any order. Examples are provided in examples 8 and 9. The compensation for example 9 may be computed with row vectors as follows:

```
$P3N◊N535-A◊$P3S◊Fluorescein◊
$P4N◊G560-A◊$P4S◊Phycoerythrin◊
$P5N◊G660-A◊$P5S◊PE-Cy5◊
$SPILLOVER◊2,B525-A,G560-A,1.0,0.1,0.03,1.0◊
```

These keyword, value pairs specify a 2-way compensation between measurements B525-A ("Fluorescein") and G575-A ("Phycoerythrin"), with a spillover of 10% from B525 to G575 and 3% from G575 to B525.

	Detector	
Fluorochrome	**B525-A**	**G575-A**
Fluorescein	1.0	0.1
Phycoerythrin	0.03	1.0

Example 8: 2-way compensation spectral matrix

```
$P3N◊B525-A◊$P3S◊Fluorescein◊
$P4N◊G575-A◊$P4S◊Phycoerythrin◊
$P5N◊G660-A◊$P5S◊PE-Cy5◊
$SPILLOVER◊3,G575-A,B525-A,G660-A,1.0,0.03,0.2,0.1,1.0,0.0,0.05,0,1.0◊
```

These keyword, value pairs specify a 3-way compensation including measurements B525-A ("Fluorescein"), G575-A ("Phycoerythrin") and G660-A ("PE-Cy5").

	Detector		
Fluorochrome	**G575-A**	**B525-A**	**G660-A**
Phycoerythrin	1.0	0.03	0.2
Fluorescein	0.1	1.0	0.0
PE-Cy5	0.05	0.0	1.0

Example 9: 3-way compensation spectral matrix

1. Construct a row vector $\mathbf{e}$ of the G575-A, B525-A, and G660-A values (scale values of the event from the FCS file);
2. Create matrix $\mathbf{S}^{-1}$ as the inverse of the spectral matrix $\mathbf{S}$;
3. Perform matrix multiplication of $\mathbf{eS}^{-1}$, which will result in a row vector of compensated Phycoerythrin, Fluorescein, and PE-Cy5 values.

To compute with column vectors, simply transpose the spillover matrix

1. Construct a column vector $\mathbf{e}$ of the G575-A, B523-A, and G660-A values (scale values of the event from the FCS file);
2. Create matrix $\mathbf{S}^{T}$ as the transpose of the spectral matrix $\mathbf{S}$;

3. Create matrix $(\mathbf{S}^T)^{-1}$ as the inverse of matrix $\mathbf{S}^T$;
4. Perform matrix multiplication of $(\mathbf{S}^T)^{-1}\mathbf{e}$, which will result in a column vector of compensated Phycoerythrin, Fluorescein, and PE-Cy5 values.

3.3.62 $SRC

`$SRC◇string◇`

`$SRC◇J. Doe, SP123456◇`

The value of the `$SRC` keyword specifies the source of the specimen. Note that this keyword value could contain patient information, which may be protected by region specific laws and guidelines. The acquiring laboratory may choose to use encoded information for this keyword value. Generally, multiple specimens may be related to the same source. See also description of the `$SMNO` (section 3.3.60), and the `$CARRIERID`, `$CARRIERTYPE`, and `$LOCATIONID` keywords (sections 3.3.8, 3.3.9 and 3.3.28, respectively) for carrier identification. In the provided example, the specimen source was labeled as J. Doe, SP123456.

3.3.63 $SYS

`$SYS◇string◇`

`$SYS◇Mac OS X v10.5◇`

The value of the `$SYS` keyword specifies the type of computer and the operating system under which the data set was collected. In the provided example, the data set was collected on a Mac OS X v10.5.

3.3.64 $TIMESTEP

`$TIMESTEP◇f◇`

`$TIMESTEP◇0.01◇`

The value of the $TIMESTEP keyword specifies the time step of the time measurement in seconds. The time measurement is the measurement that is named Time (*i.e.*, the value of its P*n*N keyword is Time). The time step value indicates how many seconds correspond to one unit of the time measurement. In the provided example, the time step was 1/100 of a second. For illustration purposes, the following information may also be encoded in the TEXT segment: $DATATYPE◇I◇$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 FCS measurement 6 in this first event. When an event e is captured s seconds after the first event, the value $s \times 100$ is entered into measurement 6 in the event e since the time unit is hundredths of a second. In this example, the maximum time range would be 65535/100 seconds (10 minutes and 55.35 seconds).

As of FCS version 3.2, it is recommended that a time measurement should be present if applicable. If the time measurement is present, then a $TIMESTEP keyword shall also be used in that FCS data set. Time measurements must always have $PnE◇0,0◇, i.e., linear and must not specifiy $PnG.

Note that although keyword values shall be considered case sensitive, both Time and TIME have been historically used by implementors of previous FCS version to indicate the time measurement. It is therefore recommended that, when reading an FCS file, values of P*n*N keywords are compared to Time in a case insensitive way in order to identify the time measurement. When writing an FCS file, the time measurement shall be named Time.

3.3.65 $TOT [REQUIRED]

$TOT◇n◇

$TOT◇1000000◇

The value of the $TOT keyword specifies the total number of events in the data set. In the provided example, there are 1,000,000 events in the data set. Note that the value of this

keyword may be 0 to indicate an empty DATA segment.

3.3.66 $TR

$TR◊string,f◊

$TR◊FSC-H,54.5◊

$TR◊SSC,800◊

The value of the $TR keyword specifies the FCS detector signal that serves as the trigger signal for an event by a corresponding $P*n*N keyword value followed by a number, f, specifying the trigger or threshold signal level. When the detector signal exceeds the threshold level, processing for an event is initiated. In the first example, FSC-H is the event-triggering measurement, and the FSC-H signal level value must exceed 54.5 in order for an event to be triggered and saved in the FCS file. In the second example the signal from the SSC detector must exceed 800 to trigger evaluation of an event.

3.3.67 $UNSTAINEDCENTERS

$UNSTAINEDCENTERS◊m,string1,...,stringn,f1,...,fn◊

$UNSTAINEDCENTERS◊3,FL1-A,FL2-A,FL3-A,123.45,234.56,65.4321◊

The $UNSTAINEDCENTERS keyword represents the average background and autofluorescence of a selected unstained cell or particle population, where m is the number of dimensions included, the string values are the $P*n*N values for the relevant measurements and each f_i value is the median, mean or other central tendency for that measurement dimension. Normally, the population chosen to define $UNSTAINEDCENTERS will be representative of a population of interest in the current sample. The $UNSTAINEDCENTERS vector could be used as an offset in the primary measurement data to translate the specified point to the data origin. This point remains at the origin in the compensated output so that the overall data space is translated relative to the result of compensation of unadjusted data. This results in

dye estimates that are relative to the $UNSTAINEDCENTERS values, but it does not affect the differences between data points.

However, caution is advised in using a $UNSTAINEDCENTERS vector as an offset in computing compensated results. In particular, some current software, particularly Beckman Coulter products, uses an averaged background and autofluorescence vector as an offset to measurement data before applying the compensation matrix calculation and then adds the same averaged background and autofluorescence vector to the compensated result. This last step is invalid in that it adds a vector specified in initial measurement space to the compensated results that are specified in dye estimate space. The net result is similar but not equivalent to the output of the typical method for compensation that does not attempt any adjustment for background and autofluorescence.

3.3.68 $UNSTAINEDINFO

```
$UNSTAINEDINFO◊string◊
$UNSTAINEDINFO◊medians of scatter gated unstained sample◊
```

This text string should be used to describe the cell or particle source, gating, type of statistic used to obtain the $UNSTAINEDCENTERS vector.

3.3.69 $VOL

```
$VOL◊f◊
$VOL◊12345.6◊
```

The value of the $VOL keyword specifies the volume in nanoliters (nL) of sample that was consumed during data acquisition. In the provided example, the data set was acquired in a sample volume of 12345.6 nL (or 12.3456 μL).

3.3.70 $WELLID [DEPRECATED]

$WELLID◇string◇

$WELLID◇A7◇

The value of the $WELLID keyword specifies the location on the plate of the well that was the source of the data set. In the provided example, A7 is the well identifier.

The $WELLID keyword has been deprecated in FCS ; keywords $CARRIERID (section 3.3.8), $CARRIERTYPE (section 3.3.9) and $LOCATIONID (section 3.3.28) should be used to identify the carrier holding the sample that was the source of a data set.

3.4 DATA Segment

The DATA segment contains the list mode data described in the primary TEXT segment. The position of the DATA segment is specified in both, the HEADER segment and the primary TEXT segment (see sections 3.1, 3.2). The DATA segment may therefore be placed outside of the first 99,999,999 bytes of the data set.

The data are written to the DATA segment in one of four allowed formats (unsigned binary integer, single precision floating point, double precision floating point and ASCII encoded integers) described by the $DATATYPE keyword in section 3.3.14). The ASCII data type is deprecated and will be removed in a future version. Values in the DATA segment are stored in a sequence of $v_{1,1}$, $v_{1,2}$, ..., $v_{1,n}$, $v_{2,1}$, $v_{2,2}$, ..., $v_{2,n}$, ..., $v_{m,1}$, $v_{m,2}$, ..., $v_{m,n}$, where the value $v_{i,j}$ is the value of FCS measurement number j for event number i; $j \in [1, n]$, $i \in [1, m]$, n is the value of the $PAR keyword (section 3.3.34), m is the value of the $TOT keyword (section 3.3.65). That means that all values of the first event are stored first, followed by all values of the second event, etc.

For list mode storage with unsigned binary integers ($DATATYPE◇I◇), the Pn$B set of keywords specify the bit width for the storage of each FCS measurement value. The Pn$R set of keywords specify the range for each FCS measurement. For example, $P1B◇16◇$P1R◇1024◇

specifies a 16-bit field for FCS measurement 1 and a range for the values of FCS measurement 1 as 1024 distinct values from 0 to 1023, which corresponds to 10 bits ($\log_2(1024)$). Therefore, implementors shall use a bit mask when reading these list mode values to insure that erroneous values are not read from the 6 unused bits. If the `$P`*n*`R` value 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 floating point data type (`$DATATYPE`◇`F`◇ or `$DATATYPE`◇`D`◇), the `$P`*n*`B` set of keywords also specify the bit width for the storage of each FCS measurement value; however, this width shall correspond to 32 and 64 bits, respectively. No bit mask shall be applied when reading floating point data. Negative values as well as values exceeding the value of the `$P`*n*`R` keyword may be present and are legitimate.

3.5 ANALYSIS Segment

The ANALYSIS segment, when present, contains the results of data processing. The position of the ANALYSIS segment is specified in both, the HEADER segment and the primary TEXT segment (see sections 3.1, 3.2). The ANALYSIS segment may therefore be placed outside of the first 99,999,999 bytes of the data set.

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 are collected and stored. The ANALYSIS segment may contain any user-defined contents; the format of the ANALYSIS segment is not restricted by the FCS specification.

3.6 OTHER Segments

Implementors may create any number of OTHER segments as they choose. The position of the OTHER segments is specified in the HEADER segment only and therefore, all OTHER segments shall be placed within the first 99,999,999 bytes of the data set (see section 3.1). The OTHER segments may contain any user-defined contents; the format of the OTHER segment is not restricted by the FCS specification.

3.7 CRC Value

If an implementor chooses to compute and store a CRC word, then the CRC word shall be 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 may be a TEXT, DATA, ANALYSIS or OTHER segment). The CRC word shall be a 16-bit cyclic redundancy check value. The 16-bit CRC word shall conform to the CCITT standard. It shall use the CCITT polynomial $X^{16} + X^{12} + X^5 + 1$ with each input character interpreted as its bit-reversed image. For testing purposes, note that the result of any compatible CRC calculation of the string `CatMouse987654321` shall return 49805 (C28D hex). In addition, see Appendix B for reference implementations of a compatible CRC. The CRC value shall be encoded in ASCII [8] and placed in the 8 bytes immediately after the last segment of the data set. This value shall be left-padded with `0` in order to fill out the 8 bytes reserved for the CRC. For example, a CRC value of 49805 (C28D hex) shall be encoded as `00049805` in the FCS file. 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 shall be filled with eight ASCII-encoded `0` characters (*i.e.*, `00000000`).

3.8 White Space

All space within an FCS file that is not contained in:

(i) the HEADER,

(ii) a segment specified in the HEADER or,

(iii) the CRC

shall be filled with a space character (ASCII code 32). This includes all 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 data set or file (if any).

Appendix A References

[1] Murphy RF and Chused TM. A proposal for a flow cytometric data file standard. Cytometry, 1984:5(5):553–555.

[2] Dean PN, Bagwell CB, Lindmo T, Murphy RF, and Salzman GC. Introduction to flow cytometry data file standard. Cytometry, 1990:11(3):321–322.

[3] Dean PN, Bagwell CB, Lindmo T, Murphy RF, Salzman GC, and Data File Standards Committee of the Society for Analytical Cytology. Data file standard for flow cytometry. Cytometry, 1990:11(3):323–332.

[4] Seamer LC, Bagwell CB, Barden L, Redelman D, Salzman GC, Wood JC, and Murphy RF. Proposed new data file standard for flow cytometry, version FCS 3.0. Cytometry, 1997:28(2):118–122.

[5] Spidlen J, Moore W, Parks D, Goldberg M, Bray C, Bierre P, Gorombey P, Hyun B, Hubbard M, Lange S, Lefebvre R, Leif R, Novo D, Ostruszka L, Treister A, Wood J, Murphy RF, Roederer M, Sudar D, Zigon R, and Brinkman RR. Data File Standard for Flow Cytometry, version FCS 3.1. Cytometry A, 2010:77(1):97–100.

[6] Bray C, Spidlen J, and Brinkman RR. FCS 3.1 Implementation guidance. Cytometry A, 2012:81(6):523–526.

[7] Bradner S, The Internet Engineering Task Force, Network Working Group. Request for Comments: 2119 – Key words for use in RFCs to Indicate Requirement Levels, 1997. Available at `http://www.ietf.org/rfc/rfc2119.txt`.

[8] American National Standards Institute. ANSI INCITS 4-1986 (R2007). Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII), 1986 edition, 1986.

[9] Yergeau F, The Internet Engineering Task Force, Network Working Group. Request for Comments: 3629 – UTF-8, a transformation format of ISO 10646, 2003. Available at `https://tools.ietf.org/html/rfc3629`.

[10] Institute of Electrical and Electronics Engineers. IEEE Std 754-2008 – IEEE Standard for Floating-Point Arithmetic, 2008. Available at `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=4610935`.

[11] International Organization for Standardization. International Standard ISO 8601: Data elements and interchange formats – Information interchange – Representation of dates and times, 2004. Third edition, 2004-12-01, available at `http://www.uai.cl/images/sitio/biblioteca/citas/ISO_8601_2004en.pdf`.

[12] Blenman K, et al. ISAC Probe Tag Dictionary. International Society for the Advancement of Cytometry, Data Standards Task Force, 2020. Available at `https://github.com/ISAC-DSTF/ProbeTagDictionary`.

[13] Spidlen J, Moore W, Brinkman RR, Brinkman R, Almarode J, Anderson E, Blenman K, Bray C, Buscher M, Cavenaugh J, Goldberg M, Hyun B, Kripal D, Krouse K, Leif R, Moore W, Novo D, Parks D, Spidlen J, Treister A, Wood J, and Zordan M. ISAC's Gating-ML 2.0 data exchange standard for gating description. Cytometry A, 2015: 87(7):683–687.

Appendix B CRC Reference Implementation

Two reference implementations of a compatible CRC check sum are provided along with this specification in order to guide implementors and facilitate FCS compliance.

- A C# implementation can be downloaded from `http://flowcyt.sf.net/fcs/FCSCRCReferenceImplementation_CS.zip`. This is a direct implementation of an FCS compatible CRC16 CCITT check sum without any additional dependencies.

- A C++ implementation can be downloaded from `http://flowcyt.sf.net/fcs/FCSCRCReferenceImplementation_CPP.zip`. This implementation demonstrates the use of the Boost library to calculate FCS compatible CRC16 CCITT check sum. As such, this implementation is dependent on the Boost library and is subject to the Boost Software License as stated in the provided source code.

Upon approval of the FCS specification, these reference implementations will also be published at ISAC website `http://www.isac-net.org`.

Appendix C Major Changes Since FCS 3.1

This informative Appendix provides a brief summary of the major changes to the FCS specification since FCS version 3.1. A short rationale for these changes is also included.

1. The file version identifier has been changed from FCS3.1 to FCS3.2.
2. The terminology and structure of the specification has been improved and several ambiguities have been clarified. Former *FCS parameters* are now referenced as *FCS measurements*. The concepts of *probes* and *detectors* have been added. The ISAC Probe Tag Dictionary located at `https://github.com/ISAC-DSTF/ProbeTagDictionary` may be leveraged to populate those in a standardized way.
3. The CRC calculation has been clarified and reference implementations provided.
4. The *Line Feed* character (ASCII code 10) is recommended to be used as the delimiter character; the ◇ symbol is used to indicate the delimiter in examples throughout this specification.
5. The format of the ANALYSIS segment has been relaxed and related keywords (`$CSMODE`, `$CSVBITS` and `$CSV`*n*`FLAG`) removed.
6. Previously deprecated gating keywords, such as `$GATE`, `$G`*n*`E`, `$G`*n*`F`, `$G`*n*`N`, `$G`*n*`P`, `$G`*n*`R`, `$G`*n*`S`, `$G`*n*`T`, and `$G`*n*`V` have been removed.
7. Previously deprecated support for histograms has been removed along with related keywords, such as `$PK`*n* and `$PKN`*n*.
8. Keywords `$GATING`, `$R`*n*`I` and `$R`*n*`W` have been deprecated.
9. The `$P`*n*`P` keywords have been deprecated.
10. Keywords `$DATE`, `$BTIM` and `$ETIM` have been deprecated in favor of `$BEGINDATETIME` and `$ENDDATETIME`, which follow ISO standards.

11. The $PLATEID, $PLATENAME and $WELLID keywords have been deprecated; new more generic keywords $CARRIERID, $CARRIERTYPE and $LOCATIONID have been added to identify the carrier holding the sample that was the source of a data set.
12. The $MODE has been deprecated and if used, the value of this keyword has been fixed to L as list mode is the only supported mode. This keyword will be removed in a future version since it now longer servers any purpose.
13. Values of $P*n*B keywords that are not divisible by 8 have been deprecated.
14. The description of the $BEGINSTEXT and $ENDSTEXT keyword has been corrected.
15. The $P*n*G keyword has been restricted to integer data type only. This means that channel values and scale values are the same for floating point data.
16. An optional offset has been added to the format of the value of the $P*n*CALIBRATION keywords.
17. The $CYT keyword has been made required.
18. The $BEGINSTEXT and $ENDSTEXT keywords have been made optional.
19. The $BEGINANALYSIS and $ENDANALYSIS keywords have been made optional.
20. The optional $P*n*DET keywords have been added to specify the detectors' names.
21. Optional $P*n*TYPE and $P*n*FEATURE keywords have been added to formally specify FCS measurement signal types and evaluation features (e.g., Area), respectively.
22. Optional $P*n*TAG and $P*n*ANALYTE keywords have been added to specify the dyes used for FCS measurements and the target molecules or processes associated with FCS measurements, respectively. The ISAC Probe Tag Dictionary located at https://github.com/ISAC-DSTF/ProbeTagDictionary may be leveraged to populate the values of those keywords in a standardized way.
23. The optional $UNSTAINEDCENTERS and $UNSTAINEDINFO keywords have been added to

capture the vector of central values of an unstained sample and document how they were obtained.

24. The `$FLOWRATE` keyword has been added.

25. The `$BEGINDATETIME` and `$ENDDATETIME` keywords have been added.

26. The `$P`*n*`DATATYPE` keyword has been added to allow FCS measurements that are intrinsically integers (such as Time or indices) to be more correctly represented as integer values when the remaining measurements are more correctly represented as single precision floating point values, or to allow one or more measurements to be represented as a double precision floating point value while the remaining measurements are represented by single precision floating point values.

Appendix D ISAC DSTF Members

All the ISAC Data Standards Task Force Members have reviewed and voted on the FCS 3.2 specification; many of the members have provided useful comments that lead to various improvements throughout the specification. The current list of ISAC Data Standards Task Force members consists of the following individuals in alphabetical order:

Ryan Brinkman, BC Cancer Agency (chair)

Somi Afiuni, Fluidigm

Ernie Anderson, Beckman Coulter Life Sciences

Kim Blenman, Yale University

Martin Büscher, Miltenyi Biotec

Chris Bray, Verity Software House

Brian Capaldo, National Cancer Institute

James S. Cavenaugh

Kamila Czechowska, Spiden AG

Veronika Kortisova-Descamps, BIO-RAD

Michael Goldberg, BD Biosciences

Zen Gong, Cytek Biosciences

Joseph Hanson, Roswell Park Comprehensive Cancer Center

Barry Hurt, Thermo Fisher Scientific

Robert Leif, Newport Instruments

Wayne A. Moore, Stanford University

David Novo, DeNovo Software

David R. Parks, Stanford University

Matthew Shallice, Thermo Fisher Scientific

Josef Spidlen, BD Life Sciences – FlowJo

Stefanie Trop, Beckman Coulter Life Sciences

Logan Wu, Agilent Technologies

Michael Zordan, Sony Biotechnology