# A C3D format extension

This proposal is based on a conversation when the C3D format was first developed, it adopts a concept that was discussed when the C3D format first appeared, and is designed to extend the C3D file format in a way that maintains a functional degree of compatibility for existing applications, while offering motion capture systems the option to create new C3D compatible files that support storing markerless data, inertial measurement unit data, and other new sensor data recorded in the 3D motion, biomechanics, gait analysis, research, and animation environments using multiple sample rates while maintaining the basic C3D format.

This concept was originally described in a conversation with Douglas McGuire, an ADTECH programmer working with Andrew Danis, as part of the C3D file format design idea but was probably not adopted in the NIH development environment because it may have create larger files that required complex FORTRAN processing.

The extension aims to make it easy for application programmers to add support to existing applications and C3D file environments by documenting an implementation of the original design discussion, adapted after many years of seeing storage issues in programmers C3D environments. The ideas discussed simply possibilities for the future.

Any change to the C3D format must provide support for all C3D environments, a factor that makes any change to the C3D format complex because the C3D file organization is specific to the type of data stored, while data and parameters may only be read by programs that have a complete knowledge of the file structure and the interpretation of the stored data. Unfortunately it is common to see applications illustrate an incomplete understanding the capabilities and flexibility of the file structure. For example:

* Programmers may assume that the first two bytes of the parameter block in a C3D file contain the same bytes that an ADTECH parameter file requires, even though the C3D format has always specified that the first two bytes of the parameter block should be ignored when stored in a C3D file. When their incorrect assumption causes an application to fail to read a C3D file, they report (incorrectly) that the file does not conform to the C3D format.
* There have been some attempts to add a second parameter block to a C3D file after seeing the original ADTECH option descriptions. But the C3D file header only documents the location of a single parameter block so existing applications normally ignore an additional parameter block and may overwrite it if any changes are made to the new file. Additional parameter blocks are possible in the ADTECH format but are not supported by the C3D format.
* There have been some attempts to add a second data block to a C3D file. While this is easy to do, existing applications will not be able to read both data blocks and may fail to preserve the second block when they are determining the C3D file size because the C3D header only documents the existence of a single data block.
* There have attempts to change the C3D data block format to add unique data types described by new parameter groups. While this is supported by the ADTECH file format, any application that does not fully understand the new data section formats will be unable to access and interpret the data when it is not stored using the standard 3D point format, defined, and fully documented in the C3D user guide.

These issues are a result of C3D files being related to the more general ADTECH file format which was designed to hold multiple data types and the associated parameters within a single file. The C3D format is a subset of the ADTECH format and, as a result, it has a precisely described structure created to provide universal access to a large variety of data and associated parameters in a single defined format.

While the C3D format possesses many of the ADTECH file format options the major advantage is that, unlike the ADTECH format, the C3D format is documented. This has resulted in an environment that anyone can access but since the C3D format is formally documented, major changes cannot be made. Creating files with additional parameter and data sections is supported by the ADTECH format design but any interpretation of the new parameters and data section format will require that they are fully documented and existing applications are updated to interpret the changes. As a result existing applications will see the new file format as unreadable until they are updated, which will not happen until the changes are documented and applications are updated to access and support the changes.

## Potential C3D extensions

It is important that anyone reading this section treats it as simply a discussion of some potential options, not documentation of the current C3D format.

The C3D file format has been used since the mid 1980’s and initially there were changes to the format supported by AMASS, but these were all well documented and as a result any application written today that meets the C3D standard can read all C3D files that have been created more than 30 years ago. The C3D format has a large number of application sources and companies offering applications that support the format, so any extension to the C3D format must be documented to maintain its compatibility and implementation standards. When the C3D format was created there were plans to support multiple file data sections to support a range of data environments. This plan was discussed but not originally documented as the public accessibility of the data became the major feature.

The optional extensions described here are all designed to require relatively little effort to implement and offer a format improvement while maintaining standard C3D file accessibility for existing users.

#### C3D header section updates

The 12 word C3D header record, at the start of the C3D file header section, enables all applications to quickly determine the size and structure of each C3D file so any changes to the header record can affect the interpretation of the C3D file.

The first byte of the C3D file header is a pointer to the start of the parameter section which normally the second 512-byte block in the file, so updating the pointer to the location of the third block in the C3D file has the potential to double the size of the header section if this would offer any advantages in future.

The second byte of the C3D file header has always been 0x50h (80 decimal, an ASCII **P** character) and has been documented as defining the C3D file format; essentially it defines the C3D file header record structure and the data section format. Application programmers who are not familiar with the ADTECH format may believe that it cannot be changed; but essentially it defines the C3D file contents so any major content changes can update header byte #2, allowing all applications to determine the format of the data stored in the C3D file. Changing the second byte will prevent existing applications from reading the C3D file until they are updated to read and interpret the new data section format.

Any change to the second byte in the C3D file allows the new format to modify the C3D file header record values as well as the data section structure. The current C3D header contains a single header record that stores the original raw data frame range as two 16-bit integers (words 4 and 5). For many years these have been used to calculate the number of 3D frames in a C3D file but they only support a maximum of 65535 frames, leading to a complex method of determining the actual frame count in many applications. Replacing these two 16-bit integers with a single 32-bit floating-point 3D frame count has the potential to extend the C3D format considerably but will cause issues with when existing applications read the two header record words as two raw frame numbers instead of a single 32-bit floating point value. So this is an option for the future, not something that can be done to C3D files without causing problems.

#### C3D data section updates

Any changes to the C3D data section must meet the basic requirements of the C3D format, and support the storage of data as scaled 16-bit integers and 32-bit floating point values. Both methods are affected by the C3D endian type. All new C3D data section formats need to continue to support all three endian formats and the storage of data as integers or floating point to maintain universal C3D accessibility.

Before making any significant changes to the C3D data section format that would require rewriting all applications to even read the data section, it is worth considering simply updating the interpretation of the existing 3D point format which stores both 3D data points with residuals, and associated analog data data samples. The analog data samples are stored in a synchronized sequence with each 3D data frame so simple changing the interpretation of the “analog” values by creating a new ANALOG parameter to define the storage method means that a new sequence of data can be stored with no C3D data section format changes at a binary level. This means that updating C3D applications to read the new data storage type would only need an update of the interpretation and processing of the data.

For example, there has been a new data section format proposed that stores each 3D location as 17 floating point values, defining a 4x4 matrix and one additional value, requiring that all C3D applications are rewritten to first read the new format and then interpret and process the data. But 17 values could be written as 17 sequential values in a one analog channel and a new ANALOG:MATRIX\_CHANNEL parameter created to define the channels associated with the new data interpretation – each 3D frame having at least 17 analog samples (a 60Hz 3D point rate would need to have an analog sample rate of 1020Hz (60x17) or higher) and the interpretation defined to describe the sample sequence related to the matrix values. This would mean that all existing applications can read the “analog channel” data and would only need to be updated to interpret the data, simply reading the first 17 values of each defined MATRIX\_CHANNEL as the matrix associated with the 3D frame, not a temporal sequence of analog samples.

An additional complexity is that the C3D format supports the storage of analog data values as both pre-scaled floating point values and scaled integer value, each with an associated unique floating point scale factor which limits the resolution of values stored as 16-bit integers. The C3D format requires that the option to support both floating point and integers storage exists so this needs to be supported – if the integer method is noticeably inaccurate then an option would be to store each sample as a 32-bit integer by doubling the analog sample rate. This method preserves the standard 3D point location and analog storage format, allowing public access to the “new” data format once applications determine the interpretation of the data values stored as “analog samples” - a change requiring relatively little effort for most users if the new format is fully described.

## An Extension proposal

The traditional C3D format supports a file with one header section, one parameter section and one data section. The original concept that was considered was that a C3D file could contain one header section and multiple collections of data sections, each with an associated parameter section. This has the potential of supporting multiple 3D and Analog data sources, sample rates, and data storage formats, storing data from a wide range of different sensors in a single file. When this idea was originally considered the collection of syncronized data was very complex and data storage space limited – environments that have changed in the last 30 years.

The proposal documentd a method of supporting 12 data sections, each with a related parameter section. This extends the C3D file storage in multiple ways:

1. A standard C3D file (with one data section) can be processed and updated while retaining the original data, by storing the processed data in a separate pair of parameter/data sections, while allowing existing applications to be presented with either the new results or the original data. Multiple pairs of data/parameters sections are supported allowing clinical and research users to evaluate and update the data processing methods after comparing each set of results to the original or previous process results in a single environment.
2. The new C3D environment allows each data section to have a unique data sample rate and potentially a new structure, while retaining the ability to store standard C3D points and analog data. This feature could support multiple synchronized IMU and USB data collections, each with a unique sample rate and data storage format, allowing applications the option to process the data and create another data/parameter section using the standard 3D point/data format that all existing applications can access.
3. In an animation environment, multiple subjects can be stored in a single synchronized file, allowing applications to process and modify one subject while maintaining a group of subject interactions from separate sources.

The extension is designed to support a degree of compatibility that should allow all existing applications to read the new files but not modify them in any way. The proposal has the “benefit” of supporting multiple new data collection environments but this will add considerable complexity to the data analysis. The original C3D format was designed to store data in a simple format that guaranteed data synchronization with a single 3D environment and data collection sampling ratio; the new environment offers the ability to support unique 3D data environments and multiple sampling rates, each with a new data storage format in synchronization. As a result, storing data will be easy but the processing may be complex.

This is a proposal, not a format definition, and is open to being updated as programmers start to create applications.

#### The C3D environment

One of the most significant features of the C3D file format has been its resistance to technological obsolescence. The format of media used to store files has changed significantly over the lifetime of the C3D format which originally was written with data stored on 9-track ANSI magnetic tape, 10Mb hard disks (DEC RL02 drives, 15 inches in diameter), and 8-inch floppy disks. These days, disk drives with 10Tb or more storage are common and everyone has access to virtually unlimited Cloud storage. As a result of the original design effort, a C3D file created in 1987 by a FORTRAN application running on a PDP-11, with a single 16-bit CPU and 64kb of memory stored on an RL02, can be opened and processed by all modern C3D applications written today.

This is an achievement matched by few formats in any industry, so this level of compatibility needs to be retained in any extension to ensure that all C3D files can remain readable by all existing applications to avoid disrupting users worldwide.

#### Extending the C3D format

|  |  |
| --- | --- |
| This extension expands a concept discussed with Dr. Andrew Danis when the C3D format was created. The original idea has been updated to add support for larger files. | When the C3D format was developed, computer disk storage access was much slower, multicore processors did not exist, and the storage of data associated with a single frame in different sections of a file was considered to be both inefficient and unreliable, but now the advance of computing technology makes the option to add additional sections to the C3D file feasible. The ideas described were discussed when the C3D format was designed, but were not described as part of the original C3D format documentation, which was written to define the basic C3D structure for multiple motion capture systems in different computing environments. |

The original C3D documentation states that all applications start by reading the first 512-bytes of a C3D file to locate the parameter and data sections in a C3D file which describe the formats used and provided detailed descriptions of the information stored in the file. This is the standard C3D file structure described in the original public AMASS format documentation which described the C3D file format used by the AMASS software, but did not document the options that had been designed to support the C3D environment because these were part of the original ADTech file format family of which the C3D format was just a member. The documentation simply described the format of the file created by the AMASS photogrammetry software which automatically generate 3D marker locations from raw camera data.

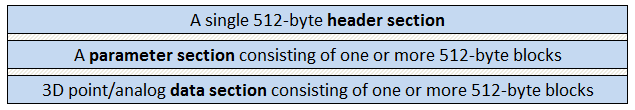


Figure 1 - The original C3D file structure

The original documentation mentioned the option of extending the C3D format by adding multiple parameter and data sections, but this was not described as part of the C3D file description or implementation. As a result, programmers have occasionally thought that they could add new sections to C3D files but because the location and description of additional sections in a C3D file has never been part of the format definition, when a user creates a new section, other applications fail to read the contents because the ADTech format methods were not formally documented.

The basic ADTech concept is that an application reads the first two bytes of a C3D file to learn what the file might contain – if the first byte is greater than 0x01h then the file may contain data and parameters, with the first byte pointing to the parameter section. If the first byte is 0x01h then the file may contain parameters but does not contain a header section. When the first byte of the file is 0x02h or greater then it locates the parameter section which defines the endian storage format and contains information that describes the data section contents enabling full data access.

Once the file is known to be a C3D format file, the second byte in the file describes the format of the data section. The traditional format, storing 3D points as XYZ coordinates with residuals, synchronized with interleaved analog data samples, is defined when the second byte in the file is 0x50h (an ASCII “P” character). There are currently no other data section formats formally documented but when a motion capture system creates a new data section format, the second byte of the file header record must record a new byte value to define the new format. Originally this byte was described as “defining the C3D format” but in fact, when it is 0x50h, it only defines the format used to store 3D Points and analog data in data section.

#### Describing the extension

All motion data capture systems that create new data section storage formats must define a unique second byte character and publically document the new format if it is to be accessible by all applications that read C3D files. This means that applications can open a C3D file header and immediately discover if the format is supported by the application. This feature is simply documentation of the C3D file format design, it is not an extension.

The proposed extension allows applications to add up to eleven additional new parameter/data sections to the standard C3D format that supports a standard 3D Point data section. Each pair of new parameter/data sections is capable supporting the original 3D Point format as well as new storage formats in a manner that maintains a basic level of accessibility for new C3D applications by maintaining the basic C3D file structure.

C3D applications written prior to 2021 will retain the ability to read the first pair of parameter and 3D Point/analog data sections in the new C3D files when the first parameter/data section stores data using the traditional 3D point locations interleaved with optional analog data records. This means that new applications will be able to create new C3D files that are completely accessible by users existing applications.

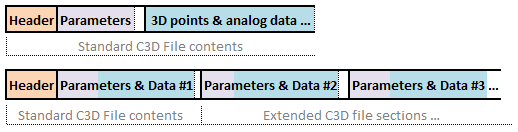


Figure 2 - The original structure and the proposed extension

Each of the eleven new storage sections has the same basic components of a single C3D file, with a unique 12-byte header, parameter section, and data section. Each new header is stored in the first 512-byte block of the C3D file which also stores a pointer to the location of each new pair of parameter/data sections. The additional sections are stored as sequential parameter/data sections in the C3D file allowing the extended C3D format to support up to twelve parameter/data sections.

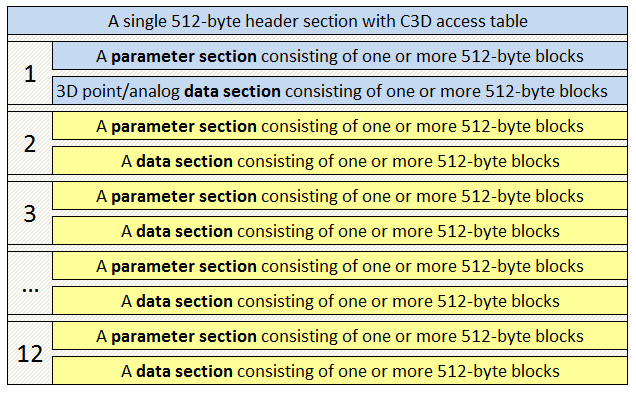


Figure 3 - The C3D format extension simply extends the existing C3D format.

Each of the new parameter/data sections can store data using the traditional C3D point data format or can use a new format defined by a motion capture company, enabling a C3D file to store multiple new data types.

#### Maintaining Compatibility

All C3D applications expect a C3D file to contain a header block, a parameter section and a 3D/analog data section containing 3D Point locations written in sync with analog data samples. The extension maintains this basic structure so the first parameter and 3D/analog data section in a new C3D file will remain readable by all current applications if the new file uses the existing point/analog data format definition defined when the second byte of the file header set to 0x50h.

The ability to add parameter/data sections to a C3D file means that an analysis can be performed on one data section and the results saved in the same file in a new data section, maintaining the relationships between the data sections. For example, in a clinical data analysis environment, a standard C3D file is normally created that contains the subject motion, force plate, and EMG signals, stored as 3D locations and analog voltages. This standard C3D data could be processed to generate clinical analysis results such as knee flexion/extension, joint moments and powers, etc., which could be stored in the C3D file as a new parameter/data section based on the ASCII format described by the original CAMARC (Computer Aided Movement Analysis in a Rehabilitation Context) standard which was the first common standard for the storage and exchange of clinical and research data, created in 1991. New standards can be created but defining a standard that works for all users is a complex process so it is much easier to add new parameter/data sections to the C3D file that use well documented common standards.

Another option in the modern world might be to collect 3D motion and analog data in a markerless data collection environment, storing the data in a new data section format with associated parameters, resulting in a file that existing applications, expecting a C3D file that contained 3D Point locations, would be unable to read.

If files only contain a single parameter/data section then the markerless data file could be processed to generate virtual 3D Point locations that document the subject motion, and be written to a separate C3D file using the standard 3D Point/analog format, resulting in two files containing the same data stored in different formats. However the proposed extension supports the creation of one C3D file by storing the calculated locations and associated analog data in the first parameter/data section of the C3D file using the 3D Point format, with the original raw markerless data moved to the second parameter/data section, resulting in a C3D file containing 3D Point data that all application can access as well as the original raw markerless data samples.

An additional analysis could then be performed to generate clinical results stored in the CAMARC format in a third parameter/data section. Alternatively the first analysis of the markerless data format could perform the clinical analysis of the markerless data and create a second parameter/data section storing the results in the CAMARC format for example.

The proposed C3D format extension is very flexible but each new storage format of the data sections in the C3D file must be documented if it is to be supported by applications to preserve the public accessibility of C3D files. Motion capture systems will have two options in future, they can create C3D files containing a single new data format that will be universally accessible, or they can store multiple data formats in a single C3D file.

### The C3D File Header extension

The first 12 words of the standard C3D file header form the standard C3D header record. Originally this record was treated as defining the contents of the C3D file, describing the number of 3D data points, analog samples, and sample rates, together with pointers that locate the standard parameter and data storage sections within the file. The function of first 12 words of a C3D file is maintained unchanged so that all existing applications can read the original C3D file contents, which are now treated as the first pair of parameter and data sections in the extended C3D file format.

|  |  |
| --- | --- |
| The extended C3D format supports twelve separate pairs of parameter and data sections. | The first 12-words are now treated as simply defining the first header record, not the entire file. When the C3D format was first designed, most of the additional words in the 512-byte header section were set to 0x00h and reserved to allow the option of adding more header records, allowing the C3D file to contain multiple parameter and data sections. As the format evolved event storage was added to the header section, as a result this new proposed extension can store up to a dozen 12 word header records in the first 512-byte block of the C3D file, with the original header at the start of the block maintaining a large degree of compatibility for all existing C3D applications which will read the first 12 words and ignore the new headers. |

Each additional header record will use the same basic structure if the associated data section uses the 3D Point format and header byte 2 is 0x50h. New data section formats will define their existence with a new value in header byte #2 which will control their use of the remaining 10 words in their header.

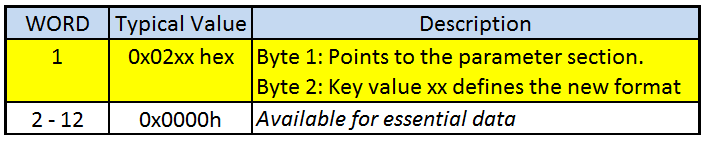


Figure 4 - New parameter/data formats are described by their new headers

Applications that access the new C3D format can read the first 144 words of the file header block to determine how many new parameter/data sections exist and the data formats used by each section. All parameter/data sections in a single C3D file must use the endian format documented in the first parameter/data section.

New data formats must store values in the header that allow applications to calculate the size of the new parameter/data section, using the structure defined by the second header byte.

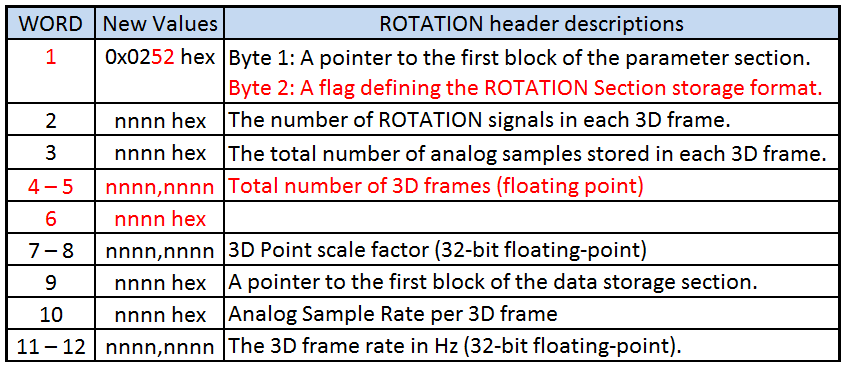


Figure 5 – The new (red items) header structure for additional new header formats

The header structure described enables the size of the parameter/data section to be easily calculated once the data storage format is documented so that applications opening a C3D file with multiple parameter/data sections defined by the C3D Access Table can calculate the size of all parts of the file and add or remove data while maintaining the C3D file format.

The recommended additional header structure is shown in Figure 5, new data storage formats need to define an ID byte (byte 2) that is unique to each format. This byte will be used to identify the “required” parameters needed to define the data section contents, the header format and the storage format for both location data samples and analog samples. These items must be documented by the entity creating the new parameter/data format in the C3D file. Word 6 in the new header is available for any essential item related to the new format, e.g. data interleave sample rate.

#### Universal factors

Individual C3D files have many different format options and while virtually all of these could be applied to the new extension environment they have the possibility of adding substantial complexity to applications data access.

* C3D files support data storage in both integer and floating-point formats, the choice is recorded by the sign of the POINT:SCALE parameter recorded in the first C3D header as a floating point value stored in words 7-8. The integer or floating point selection should apply to all data section storage – as a result these two header words are available to store new parameters in extended headers that are specific to the new formats.
* All parameter/data sections in the C3D file must use the endian format described by the parameter header in the first parameter/data section of the C3D file.
* All physical C3D measurements are recorded in millimeters – this is a C3D standard and as a result all new data storage formats and parameters must record all physical measurements in millimeters. The POINT:UNITS parameter documents the measurement unit used by every physical measurement recorded in the C3D file but does not control the measurement units.
* The C3D file header block can record 18 unique events; these events are considered “global events” that apply to all data sections in the C3D file. Each parameter section can also store data specific events by creating the standard EVENT group.
* All 3D data location records can written to the data section frame by frame with each frame also storing analog samples, the analog sample rate must always be an integer multiple of the 3D sample rate but each data section can define a unique sample rate so C3D files can now support multiple sample rates. This solves a common “problem” for motion capture systems but means that applications reading the data contents may need to be able to handle the data sample synchronization.

#### C3D access table implementation

|  |  |
| --- | --- |
| The extended C3D format documents the location of each pair of parameter data sections in the file. | The final 22 words in the C3D file header block (words 235 to 255) have always reserved for future use and words 235 to 245 are now used to record the locations of the eleven additional parameter/data sections. These words create an Access Table that allows applications reading the new format to locate each of the new extended sections in the file by recording the start of each parameter/data section added to the C3D file. |

The C3D extension is flagged by storing three bytes (0x43h,0x41h, 0x54h) starting at word 145 in the C3D file header, followed by a byte indicating the version of the C3D file (initially 0x01h) for future reference. The C3D Access Table extension supports a total of 11 additional parameter/data sections within a C3D file by creating a new method of locating each of the new parameter/data section in a way that works around the 512 byte record limit.

All C3D applications determine the default parameter and data section locations by reading the first 12 words of the C3D file header, these 12 header words are retained unchanged in the format extension to maintain compatibility. Eleven new 12-word header records can be stored immediately following the first header record; each new header record can be unique, defining the format of each related parameter/data section.

As a result the C3D file has the potential to support a total of 12 parameter/data sections header records – when a new application opens a C3D file it must read the first 12 header records to determine how many parameter/data sections exist, and their format. The Access Table records the location of each pair of parameter/data sections as a 16-bit integer associated with each additional header records, defining the locations as offset from the start of the C3D file.

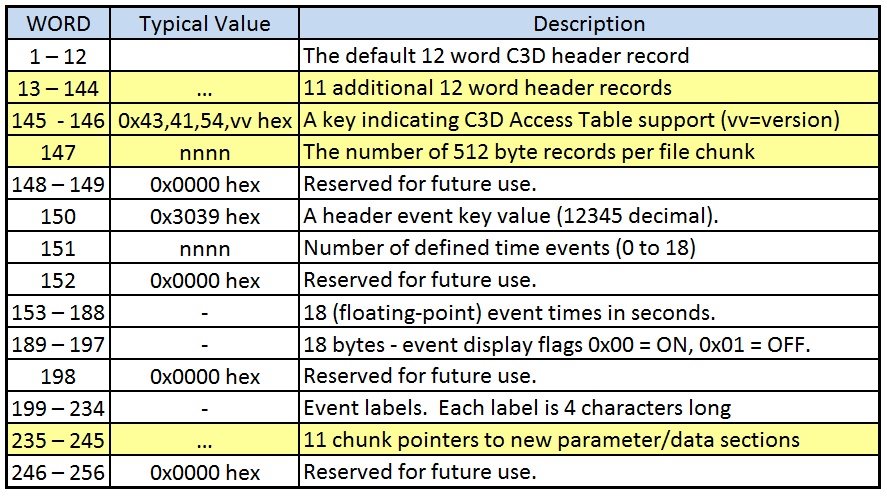


Figure 6 - The extended C3D file 512-byte header block layout highlighting the changes.

The C3D Access Table consists of eleven words in the header block (words 235 to 245) each containing a 16-bit unsigned integer that records the location of start each of the additional parameter/data sections in terms of “chunks” from the start of the C3D file. The first 512-byte block of the C3D file is the header block, this location defines the first “chunk” of the extended C3D file. The size of a “chunk” is defined by header word 147 – if this is 1 then a chunk is 512 bytes, the traditional block size that means that all parameter/data sections must start within the first 32Mb (e.g. the last 512-byte record location is 65535 \* 1). This limit is extended by increasing the number of 512 byte records define by word 147. Setting this to 3 makes the “chunk” size 1536 bytes (512\*3) allowing all sections to start within the first 96Mb.

The first parameter/data section in a C3D file always starts at the beginning of the C3D file. The first 12-bytes define the file header and result in a C3D file with a parameter/data section that can be read by all existing C3D applications if it is stores 3D Point data (indicated by the second header byte storing 0x50h). If an additional parameter/data section is stored in the file then the first byte of the next header record (words 13 to 24) is greater than 0x00h, indicating the location of the parameter section within the new parameter/data section and flagging the header existence. The second byte documents the format of the associated parameter/data section which also defined the format of the remaining 11 header words. If it is 0x50h then the new parameter/data section stores standard 3D point and analog data with standard C3D parameters and the remaining words in the header match the traditional C3D header values.

The location of each new additional parameter/data section is calculated by reading the related word from the C3D Access Table. Multiplying the pointer by the chunk count defined by word 147 creates a pointer to the 512-byte block in the C3D file containing the associated parameter/data section. The first byte of the extended header locates the start of the parameter block in the associated section, so it will normally be 0x01h indicating that the parameter block is the first 512-byte block in the parameter/data section. If the first byte is 0x02h then the parameter section is located at the second 512-byte block in the section.

Header word 9 is a 16-bit unsigned integer that locates the data section within the associated parameter/data section, all storage prior to this point is reserved for the parameter section so essentially, when the parameter section start is located, header word 9 also documents the parameter section size.

## Options for the future

The traditional C3D file format has a fixed description and does not have a way of documenting changes to the format. The addition of the C3D Access Table adds a version description byte to the C3D file header block that has always been set to 0x00h in the past. This will make it possible to update the existing C3D format in the future in a way that might cause problems with traditional applications, but would enable new applications to parse the C3D file correctly because they would know which items in the file have changed.

For example, the default C3D file format records two frame numbers in the header as 16-bit integers (words 4 and 5) that document the frame range of the raw data that created the C3D file. These have been traditionally used to calculate the frame count in the C3D file, limiting it to 65535 frames, resulting in one motion capture system manufacturer creating a work-around by storing the two frame numbers as a non-standard 32-bit integer parameter.

An alternative would be to store the frame count as a standard floating-point number stored in words 4 and 5of the C3D file header – this is recommended for all new data header formats and might be an option for the C3D point format (ID 0x50h) in the future that could be documented in the C3D file by updating the header version byte. However this could cause problems for existing applications that expect the two words to contain 16-bit integers.

#### Flexible Sample Rates

All of the parameter/data sections in a new file are potentially unique but need to be considered to be synchronized within the file, with a common physical environment and recorded with a common start and end time – a feature that allows applications to work around traditional C3D limits.

A C3D format requires that the analog sample rate is an integer multiple of the 3D location sampling rate so that each frame of location data can also store the same number of analog samples to maintain synchronization. This is unchanged by the extension except that it only applies to individual parameter/data section.

As a result, the extension give users the ability to create the first C3D parameter/data section containing 15 seconds of data sampled at a 60Hz frame rate with 10 analog data samples per channel per frame (an analog sampling rate of 600Hz resulting in 9,000 analog samples per channel). The second data section could store 15 seconds of synchronized data with a 100Hz frame rate and 14 analog data samples per frame (an analog sampling rate of 1400Hz with 21,000 analog samples per channel). The third parameter/data section could be created as a result of the data in the prior sections being analyzed and stored as 16-bit analog values with a sample rate 51 samples per gait cycle, resulting in 781 samples per analysis stored synchronized with the other two data sections.

The standard header words 150 to 234 define up to 18 event times in the C3D file, each identified by a 4 character ASCII label. These header events are now described as global events that exist in all data sections in the C3D file to support data section synchronization.

#### Flexible Data Storage

The ability to store multiple parameter/data sections in a single C3D file offers the ability to store samples derived from multiple data collection methods. The C3D format defines a single concept for 3D location storage and this should be the default because all standard applications that open a C3D file expect to find 3D points stored as XYZ location co-ordinates with associated residuals documenting the accuracy and source of the recorded data and potentially multiple analog samples stored in sync with the record 3D locations. This format is defined by the second byte in the C3D header which is 0x50 Hex, a key vale indicating that the data section contains the 3D Point formatted data.

New storage formats can be created and documented – this has to be done in a way that makes it very easy for an application to determine the data format used in the new data section. If a motion capture systems needs to create a new storage method then the second byte in the C3D file head must document that this is a new format, not the standard XYZ point format defined by 0x50 Hex. The best approach would be to create the format, publically document the storage contents and the relationship between the new format and the standard C3D format and then change the second byte in the header to flag the new format.

The potential here is to allow multiple C3D files to be created, each storing data in a different supported format. This will require that each format is fully described and publically documented. New formats need to meet the C3D design specifications, for example the data section format must support both integer and floating-point formats for all measurements and all measurements are stored as millimeters.