PHOENIX Geophysics Ltd.

Time series file specifications for MTU-8A, RXU-8A, MTU-5C, MTU-2C and MTU-5D

Table of Contents

Applies to

<u>HARDWARE</u>
- MTU-8A, RXU-8A, MTU-5C, MTU-2C, MTU-5D and derivatives

<u>SOFTWARE/FIRMWARE</u>
- Firmware contained in the hardware listed above, v1.62 and later until new notice.

File location and fragmentation conventions within the instrument card

The MTU-8A, RXU-8A and MTU-5C and MTU-2C generate natively 24KS/s time series data, while MTU-5D generates 96 KS/s time series. The conventions described hereby are subject to modifications for future feature releases.

The following sections describe the standards and convention of file naming, file organization, and data payload generate by out instrument family: MTU-8A, RXU-8A, MTU-5C, MTU-2C and MTU-5D.

Recording folders

A recording is represented by a set of files that describe the configuration, operation and data recorded by the instrument. The instrument will place all recordings in a folder called "recdata" at the root folder of the SD card.

Each recording is stored in its own uniquely identified folder. The name of the folder containing a recording will have the following convention:

12345_2016-04-29-202957

Where, in the example above:

- *12345*: Is the 5 digit string indicating the serial number of the instrument.

- *2016-04-29*: Is the GPS date (UTC + leap seconds) when data started being recorded.

- *202957*: Is a 6-character string representing the GPS time when data started being recorded. The first two characters are the hour in 24-hour format, the next two characters are the minute, and the last two characters are the second.

Files and folders contained in a recording folder

- *backend.log* : Contains information regarding the operation of the instrument

- *config.json* : Contains the configuration that the receiver used for this particular recording

- *recmeta.json* : Metadata associated with the recording created by the instrument.

- *empower_recmeta.json* : If present, it means that EMpower™ processing software, has interacted with the recording, creating an enhanced copy of recmeta.json. This is required since more comments and information can be added by the person working on data evaluation and processing, allowing us to leave the copy generated by the instrument intact.

- *recmeta.json.bak* : Safety backup of recmeta.json

- *stats* : A file containing recording statistics (experimental).

- *One or more folders named as numbers* : These folders are the data folders. Each folder contains sequential files of the time series recorded by one channel. To know which front panel the time series within each folder comes from, consult the panel map in "*recmeta.json*".

Format for the storage of continuous time series

Data file names

The naming conventions for the files created by the instrument are as follows:

AAAAA_BBBBBBBB_C_DDDDDDDD.bin

Where:
- AAAAA: Is the 5 digit string indicating the serial number of the instrument.
- BBBBBBBB: 32-bit unsigned hexadecimal label indicating the unique ID of the recording, for this instrument, which is the GPS epoch at which the recording was started.
- C: A hexadecimal label indicating the channel id (same as the folder name that contains the file) that the file corresponds to, starting at zero.
- DDDDDDDD: 32-bit unsigned hex label indicating the index of this file in the sequence of fragmented files. Fragmentation happens every minute, allowing to store 8100 years worth of continuous data. This fragmentation period is fixed as of now, but might be configurable in the future is a good reason for changing it is detected.

Data file header contents

Each sequential file contains a header. The header specifies meta-data used to configure the board while recording. Data payload is stored right after the header. This ensures that useful processed information can be extracted from a single file, even if the rest of the recording files are lost. The format of the header is described in *Table 1*.

The header fields will be packed as little endian unless otherwise specified in the notes in *Table 1*.

Data payload specification

The payload data in each continuous time series file is arranged in consecutive blocks of 64 bytes, called frames. The contents of each 64-byte frame is as follows:

DataFrame
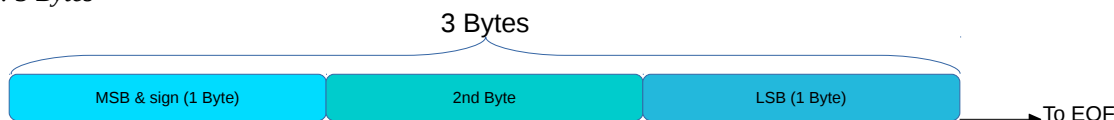*Total length: 64 Bytes = 20 samples + Footer*

- 60 bytes worth of sampled data, arranged as twenty consecutive 3-byte (24-bit) sample values, signed, big endian.
- 4 bytes containing a frame footer, arranged as an unsigned little endian integer, from which:
    - The most significant bit is a flag for internal usage.
    - The 3 next most significant bits are used as a hardware saturation count for the frame (explained below)
    - The remaining bits are used as a sequential, unsigned frame counter, which increases every frame. This counter is used to detect frame transfer loss during transmission.
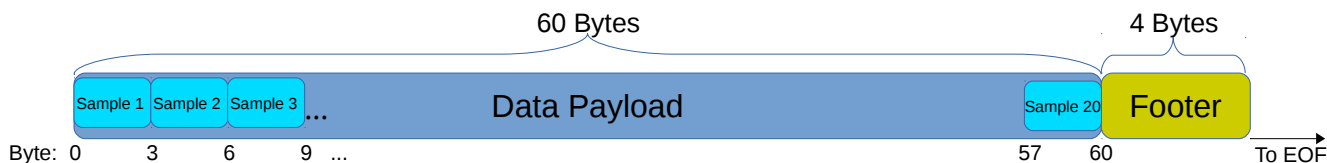
Consideration: Note that the sample data and the frame counter have different endianess while unpacking data.

DataFrame.Payload.Sample

Total length: 3 Bytes

DataFrame.Footer

Total length: 4 Bytes = 28-bit little endian sample counter + X1 1-bit flag + 3-bit saturation counter

DataFrame.Footer.SampleCounter

The sample counter is packed as little-endian. To unpack, copy the 4 bytes of the footer in an unsigned 32-bit integer and mask the result with (& 3F) to obtain the value of the counter.

DataFrame.Footer.AnalogSat

These flags indicate if there was saturation detected in the analog front end of electric channels. If any of the bits of this field is non-zero, it indicates that the front end saturated at some point while acquiring data for this frame (20 samples).

NOTE that this field does not flag digital saturation, and it is up to the higher processing layer to detect when digital values are close to the limits of the A/D range to detect digital saturations.

DataFrame.Footer.X1

This field is for internal hardware usage only.

Format for the storage of decimated time series

Objective

The principal objective of decimating the continuous time series at the original A/D sampling rate is to reduce the amount of data in mass storage, enabling the user to transfer a recording in shorter time, and to use less storage capacity for recording and archiving data.

Data file names

The naming conventions for the file created by the instrument are as follows:

$$AAAAA\_BBBBBBBB\_C\_DDDDDDDD.td\_E$$

Where:
- AAAAA: Is the 5 digit string indicating the serial number of the instrument.
- BBBBBBBB: 32-bit unsigned hexadecimal label indicating the unique ID of the recording, for this instrument, which is the GPS epoch at which the recording was started.
- C: A hexadecimal label indicating the channel id (same as the folder name that contains the file) that the file corresponds to, starting at zero.
- DDDDDDDD: 32-bit unsigned hex label indicating the index of this file in the sequence of fragmented files. Fragmentation happens every six minutes, allowing to index more than 10,000 years worth of decimated data files. This fragmentation period is fixed as of now, but might change in the future.
- E: Is a multi-character string that indicates the sampling rate of the payload contained by the file. For instance, the file containing segmented data decimated and sampled at 24000 samples/second will have the extension ".td_24K", and the file for the same recording containing (potentially continuous) data decimated and re-sampled at 150 samples/second will have the extension ".td_150".

## Format specification

DecimatedFile:
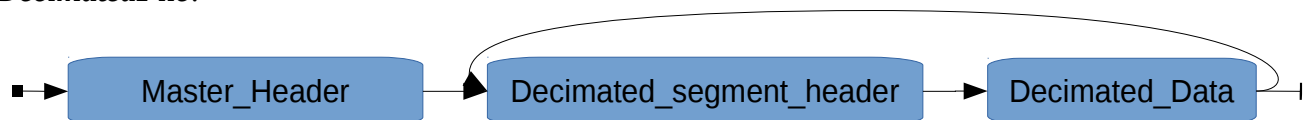
The decimation routine will store a file per decimation level, with file names similar to the names used for continuous time series, but a different extension, refer to the "decimation scheme definition" section below for more details.

DecimatedFile.MasterHeader:

Defined in *Table 2*, at the end of this document.

DecimatedFile.DecimatedSegment.Header:
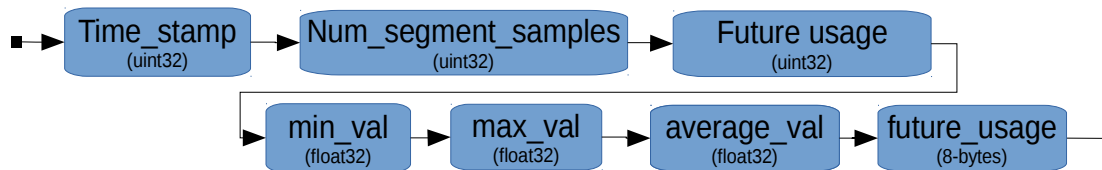
Each segment should inherit the time stamp of the first frame (or segment) of the "parent" time series. Note that this time-stamping scheme will not account for digital processing delays.

The fields **min_val**, **max_val** and **average_val** contain the minimum, maximum and average voltages measured within this data segment, in Volts.

DecimatedFile.DecimatedSegment.DecimatedData:
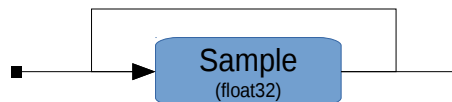
Decimated data will be packed as a raw sequence of IEEE 754 single precision floating point (32 bits) samples, scaled to "Volts at instrument input" by default.

Tables

Field name	Size [bytes]	Offset [bytes]	Data Type	Value and notes
File type	1	0	uns. Byte	0x1 (1206 *MT continuous time series file)
File version	1	1	uns. Byte	0x3
Header length	2	2	uint16	=128, for version 0x3
Instrument assembly type	8	4	char[]	Example: XX999FFF, the last three characters should be spaces for released products
Instrument serial number	8	12	char[]	Example: 99999 (Last two characters should be filled with null chars)
Recording ID	4	20	uint32	Timespamp at which the recording started, as the number of seconds elapsed since January 1st, 1970, 00:00:00 hours GPS time
Channel ID	1	24	uns. Byte	Starting at 0
File sequence	4	25	uint32	Starting at 0, increase every file
Fragmentation period	2	29	uint16	Time period covered (length) of each sequential file [s]
Acquisition board model	8	31	char[]	Example: XX999FFF, the last three characters should be spaces for released boards
Acquisition board serial	8	39	char[]	Example: 99999 (Last two characters should be filled with null chars)
Acquisition board firmware fingerprint [R]	4	47	uint32	Pack as binary, converting the string returned by the instrument to 32 bit uint.
Hardware configuration fingerprint	8	51	Byte[] (binary)	Hardware configuration flags, internal usage
Sampling rate base	2	59	uint16	Examples: 24000 (or 9600 for MTU-5D, use exponent)
Sampling rate exponent	1	61	sig. Byte	'=0 for sampling rates between 1-65000, non-zero otherwise
Bytes per sample [R]	1	62	uns. Byte	(3) for undecimated time series
Frame size	4	63	uint32	MSByte = Footer length (4) 3 LSBytes = Total frame size (64)
Reserved (Decimation node)	2	67	uint16	Default to 0
Frame count rollovers	2	69	uint16	Starts at 0, +=1 every frame count overflow
GPS Longitude	4	71	float32	WGS84
GPS Latitude	4	75	float32	WGS84
GPS Elevation above mean sea level	4	79	float32	WGS84
GPS Horizonral resolution (future)	4	83	uint32	[mm]
GPS Vertical resolution (future)	4	87	uint32	[mm]
Timing and location status	4	91	Byte	FlagsByte + NumSatByte + Stability_uint16
Future expansion	2	95	int16	(to be defined)
Future expansion	4	97	int32	(to be defined)
Count of saturated frames in the file	2	101	uint16	To read this value, if most significant bit is 0 = Read the number as is (i.e. 16-bit unsigned integer) 1 = Read the next 15 bits as a number, and multiply x16
Count of missing frames in the file	2	103	uint16	Stops counting at (overflow-1)
Battery Level	2	105	uint16	[mV]
Signal Level, Minimum in this file	4	107	IEEE 754 float	Volts at instrument input
Signal Level, Maximum in this file	4	111	IEEE 754 float	Volts at instrument input
Future Expansion	13	115	-	

Total Bytes 128

Table 1. Continuous data file header format

Field name	Size [bytes]	Offset [bytes]	Data Type	Value and notes
File type	1	0	uns. Byte	0x2 (1206 *MT decimated time series file)
File version	1	1	uns. Byte	0x2
Header length	2	2	uint16	=128, for version 0x2
Instrument assembly type	8	4	char[]	Example: XX999FF, the last two characters should be spaces for released products
Instrument serial number	8	12	char[]	Example: 99999 (Last two characters should be filled with null chars)
Recording ID	4	20	uint32	Taken from the hex UNIX time stamp of the time at which the recording started
Channel ID	1	24	uns. Byte	Starting at 0
File sequence	4	25	uint32	Starting at 1, increase every file
Fragmentation period	2	29	uint16	Time period covered (length) of each sequential file [s]
Acquisition board model	8	31	char[]	Example: XX999FF, the last two characters should be spaces for released boards
Acquisition board serial	8	39	char[]	Example: 99999 (Last two characters should be filled with null chars)
Acquisition board firmware fingerprint [R]	4	47	uint32	Pack as binary, converting the string returned by the instrument to 32 bit uint.
Hardware configuration fingerprint	8	51	Byte[] (binary)	Hardware configuration flags, internal usage
Sampling rate base	2	59	uint16	For the data payload contained for this file. Examples: 24000, 150, or for 96KS/s, it would read 9600, use exponent
Sampling rate exponent	1	61	sig. Byte	Base 10, use 0 unless field above is not sufficient
Bytes per sample [R]	1	62	uns. Byte	(4) for decimated time series
Future expansion	8	63	Byte[]	(to be defined)
GPS Longitude	4	71	float32	WGS84
GPS Latitude	4	75	float32	WGS84
GPS Elevation above mean sea level	4	79	float32	WGS84
GPS Horizonral resolution (future)	4	83	uint32	[mm]
GPS Vertical resolution (future)	4	87	uint32	[mm]
Timing and location status	4	91	Byte	FlagsByte + NumSatByte + Stability_uint16
Future expansion	6	95		(to be defined)
Reserved	4	101		Reserved
Battery Level	2	105	uint16	[mV]
Reserved	12	107		Reserved
Decimation scheme ID	4	119	uint32	Numerical identifier of the decimation scheme used for this time series
Future expansion	5	123		(to be defined)

Table 2. Decimated data file header format