Date: December 15, 2021
For FoxTelem version: 1.12

AMSAT
Telemetry Designers Handbook

1 Introduction

This handbook explains how to work with FoxTelem's telemetry formats.

A spacecraft typically sends frames of data with telemetry packed into binary fields. This allows the maximum amount of data to be transmitted in a narrow bandwidth. The telemetry decoder must extract the telemetry channels from the frame, convert them into a human readable value and then format them for display on the screen.

To support multiple spacecraft and future requirements all aspects of the telemetry parsing process are soft coded. This configuration is stored in several files. For example, if a spacecraft was called ALPHA we could have all of the configuration files shown below:

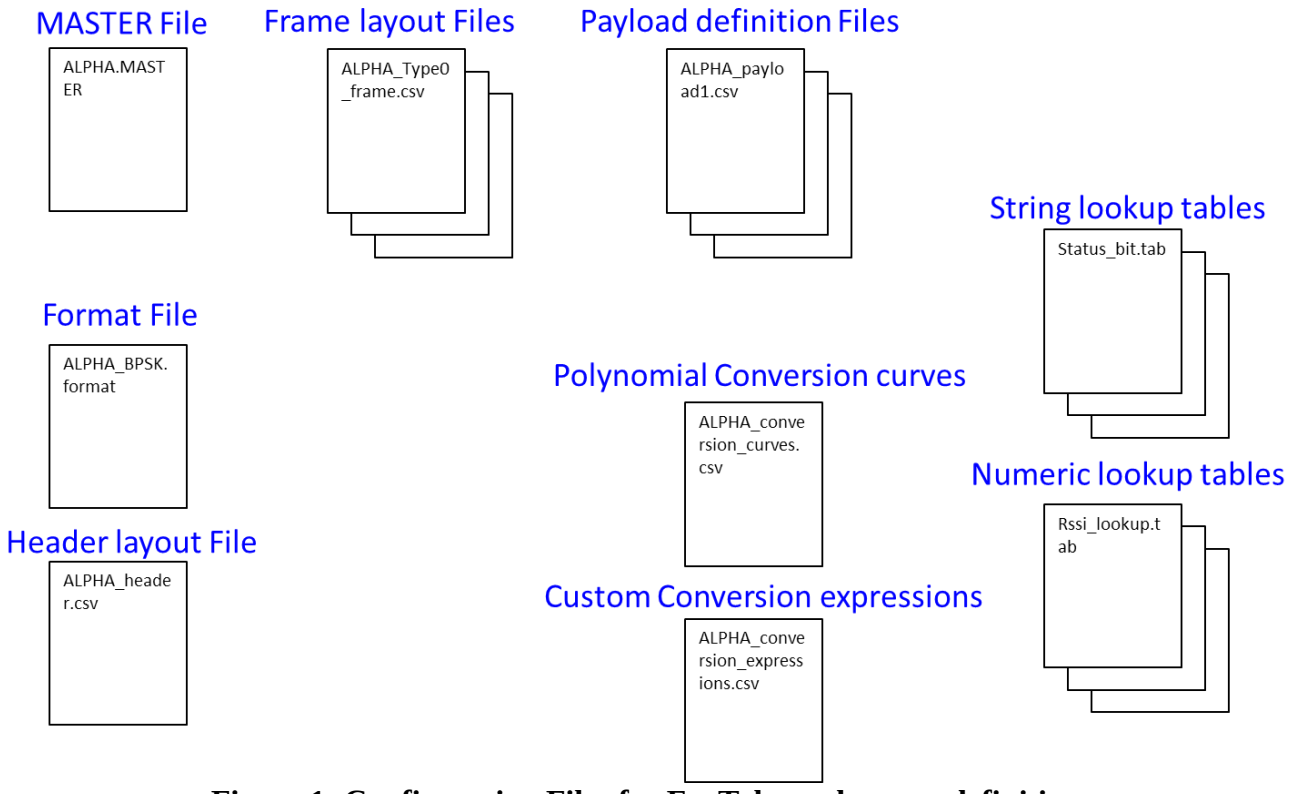


Figure 1: Configuration Files for FoxTelem telemetry definition

The MASTER file (called ALPHA.MASTER in the example above) is the starting point. It defines the key parameters for the spacecraft and references all the other files except the header layout, which is referenced from the Format file.

This document is split into the following sections. Each section will explain the content required in the MASTER file and how the individual files are setup. The final section covers additional fields in the MASTER file:

- **Spacecraft MASTER File**
- **Frame definition** – which defines the overall frame and how it is separated into payloads
- **Payload layout** – to allow the binary format to be parsed into telemetry channels
- **Field Conversions** – to convert raw values into human readable ones
- **Formatting -** for display of the final data on the screen

This approach is experimental. We are sure it can be improved. Please provide feedback and suggestions by email to g0kla@arrl.net or by logging an issue at:
https://github.com/ac2cz/FoxTelem/issues

1.1 License

2 Table of Contents

3 Spacecraft MASTER File

The master file is the key configuration file used to define a spacecraft. The following mandatory key value pairs must be defined (those in blue can be edited by the end user):

Key / Value pair	Description
foxId=X	a number provided by AMSAT to identify the spacecraft. This value is contained in the telemetry frame header
catalogNumber=X	currently not used but refers to the NOAA catalog number
name=KEPS_NAME	where KEPS_NAME is the Three Line Element (TLE) name that should match the name in the TLE file
description=A 1U Cubesat etc..	displayed on the screen where user parameters can be edited
model=0	0 for Engineering Model 1 for Flight Model 2 for Flight Spare
telemetryDownlinkFreqkHz=123456	nominal telemetry frequency in kHz
minFreqBoundkHz=123456	minimum frequency in kHz to scan to when in Fina Signal mode
maxFreqBoundkHz=123456	maximum frequency in kHz to scan to when in Fina Signal mode
numberOfLayouts=X	the number of telemetry payloads. **See section 5** of this document for the other mandatory fields that should be included to define each payload
numberOfSources=X	the number of telemetry formats. e.g., there may be a low speed and high speed downlink. **See section 4** for details.
numberOfLookupTables=X	the number of numeric lookup tables. **See section 6.3** for an explanation of the other mandatory fields needed for each lookup table
user_format=0	the default source format that should be decoded. Typically set to 0, which will be the first source in the list of sources
displayName=GUI_NAME	Where GUI_NAME is the name to show on the GUI which may be the same or different to "name" above, which is used to select the TLEs
IHU_SN=X	the serial number of the onboard computer board. Displayed in the GUI but otherwise not used
BATTERY_CURRENT_ZERO=0	a legacy value used in the FOX Series of Spacecraft. Future spacecraft should use a conversion curve or equation for this type of information.
measurementsFileName= measurements.csv	this defines the format of the measurements file and is supplied with FoxTelem
passMeasurementsFileName= passMeasurements.csv	this defines the format of the pass measurements file and is supplied with FoxTelem
EXPx=Y	a legacy scheme to describe the experiments on the FOX Spacecraft where x is the experiment number and must be 1,2,3 or 4. Y is decoded and displayed on the GUI as follows:

	0 - "Empty", 1 - "Vanderbilt LEP", 2 - "Virginia Tech Camera", 3 - "University of Iowa HERCI", 4 - "Rad FX Sat", 5 - "Virginia Tech Low-res Camera", 6 - "Vanderbilt VUC", 7 - "Vanderbilt REM", 8 - "Vanderbilt LEPF", 9 - "CAN Packet Interface", 10 - "Ragnaroc ADAC", 11 - "L-Band Downshifter"

The following optional key value pairs can be defined:

Key / Value pair	**Description**
useConversionCoeffs=true	enables the use of polynomial conversion curves, math expressions and string lookup tables. **See section 6** for details of the other fields that must be defined if this is set to true
numberOfFrameLayouts=X	the number of frame layouts. There is one layout for each frame type. This is optional because the format was fixed in early Fox spacecraft, but should be used on all future spacecraft. **See section 4** for details of the other fields that must be defined for each frame type
hasFOXDB_V3=true	Always defined as true for new spacecraft. Pre Version 3 databases are only valid for legacy spacecraft
track=false	the default state of the find signal algorithm for this spacecraft. Typically set to false. The end user can change this as they see fit for each spacecraft they are tracking
series=FOX	Used to create the filename for the extracted TLEs and potentially other files needed by FoxTelem
localServer=127.0.0.1	the IP address of a server that will be streamed telemetry. Currently this only supports CAN Packet forwarding for HuskySat
localServerPort=12345	the Port used to stream telemetry
priority=9	the default priority order in which this spacecraft will be tracked. This can be set by the end user
hasCanBus=false	enables parsing of Can Packets from Can Packet frames. Requires a sub directory with a layout for every Can packet format. Currently only supported for HuskySat
hasFrameCrc=true	Set to true if the frame contains a final CRC checksum. The checksum goes after all payloads and before the RS checkbytes This must be a 32-bit CRC as defined in this file: https://github.com/ac2cz/FoxTelem/blob/master/src/decoder/Crc32.java
mpptResistanceError=0	legacy value for the Fox MPPT. Future spacecraft should

	store this type of information in an equation or curve.
mpptSensorOffThreshold=0	legacy value for the Fox MPPT. Future spacecraft should store this type of information in an equation or curve.
hasImprovedCommandReceiver=false	set to true if using the Improved Command Receiver. Only required for Fox-1E
hasImprovedCommandReceiverII= true	set to true if using version 2 of the Improved Command Receiver. Used to parse the command count into two separate values for display
hasModeInHeader=true	true if the mode is supplied in the header of every frame
hasFixedReset=false	only set to true in an emergency if the epoch does not update on the spacecraft
memsRestValueX=X	legacy value that defined the rest point of the MEMS Gyro experiment. Future spacecraft should store this type of information in an equation or curve.
memsRestValueY=X	legacy value that defined the rest point of the MEMS Gyro experiment. Future spacecraft should store this type of information in an equation or curve.
memsRestValueZ=X	legacy value that defined the rest point of the MEMS Gyro experiment. Future spacecraft should store this type of information in an equation or curve.

4 Frame Definition

A frame contains a header, several payloads and a trailer with an optional CRC checksum and the RS check bytes. Every frame must be the same length because we use a fixed length RS Block code. That length is defined for each spacecraft in the format file.

4.1 Format File

The format file contains the following key value pairs:

```
name=GOLF_BPSK
mode=BPSK
bps=1200
data_length=669
header_length=12
header_layout_file=GOLF-T_header.csv
word_length=10
sync_word_length=31
rs_words=3
rs_padding=0,0,0
```

This file tells FoxTelem the total length of the frame, which is calculated in bits by FoxTelem as follows:

*Sync_word_length + (header_length + data_length + rs_words * 32) * word_length*

4.2 Sources

The master file must contain the following key value pair which defines the number of formats that the spacecraft transmits:

```
numberOfSources=X
```

Where X is the number of different telemetry formats.

For each format the following two key value pairs should be in the master file:

```
sourceY.name=telemetry format name
sourceY.formatName=filename
```

Where Y is the format or source number and filename specifies the format file.

In this example we have defined a source format called amsat.golf-t.lihu.bpsk which is stored in the file GOLF_BPSK.format

```
source0.name=amsat.golf-t.lihu.bpsk
source0.formatName=GOLF_BPSK
```

4.3 Frame Layouts

When a frame of data is transmitted, it can have different contents based on the type of the frame. The type is stored in the header in a field defined by the header_layout, which is specified in the format file.

For example, a type 0 frame for GOLF is defined in a frame format file as follows:

```
payload0.name=rttelemetry
payload1.name=radtelemetry
payload2.name=ragtelemetry
payload3.name=mintelemetry
payload4.name=wodtelemetry
payload5.name=wodradtelemetry
payload6.name=wodragtelemetry
```

This means that the frame is made up of 6 payloads. The payload layouts are then each defined in their own layout files, see section 5. When you sum up the lengths of all of the payloads it must be less than the data_length, as defined in the format file. If there is a 32-bit CRC then that must also fit into the data_length

The MASTER file must contain the following key value pair to define the number of frame definitions:

```
numberOfFrameLayouts=X
```

Where X is the number of frame definitions.

Each frame definition must then have the following key value pairs in the MASTER file:

```
frameLayoutY.filename=filename
frameLayoutY.name=name
```

So, in this example:

```
frameLayout0.filename=LTM_Type0_RTMIN.frame
frameLayout0.name=Realtime 1
```

We have defined frame type 0 in the file LTM_Type0_RTMIN.frame. We have given this frame the name Realtime 1.

5 Payload Layout

Each payload has a file which defines the telemetry channels and the number of bits allocated to each. The file must define a set of columns using the CSV format. The first row contains the number of rows that follow and then an optional set of column header names.

The name of each channel is defined in column FIELD and its length is defined in column BITS. The remaining columns are concerned with Field Conversions and Formatting, which are discussed in later sections of this document.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	96	TYPE	FIELD	BITS	UNIT	CONVERSI	MODULE	MODULE_I	MODULE_I	LINE_TYPE	SHORT_NA	DESCRIPTION		
2	0	wod	ReflectedP	8	mW	41	Legacy Radio	1	4	3	Ref Power	Reflected Power from the Transn		
3	1	wod	ICR3VProt	8	V	8_bit_volt	Command Receivers	4	2	3	3V Prot	ICR 3V Proteted		
4	2	wod	ICR2dot5V	8	V	8_bit_volt	Command Receivers	4	3	3	2.5V	ICR 2.5V		
5	3	wod	gTemp	8	C	8_bit_tem	Experiments	5	10	3	Gyro Temp	Gyro temperature		
6	4	wod	Xspin	8	dps	8_bit_spin	Experiments	5	1	3	X Rotation	Angular Velocity around X Axis		
7	5	wod	Yspin	8	dps	8_bit_spin	Experiments	5	2	3	Y Rotation	Angular Veolcity around Y Axis		
8	6	wod	Zspin	8	dps	8_bit_spin	Experiments	5	3	3	Z Rotation	Angular Veolcity around Z Axis		
9	7	wod	Xaccel	8	g	8_bit_acce	Experiments	5	4	3	X Accelerat	Acceleration along X axis		
10	8	wod	Yaccel	8	g	8_bit_acce	Experiments	5	5	3	Y Accelerat	Acceleration along Y axis		
11	9	wod	Zaccel	8	g	8_bit_acce	Experiments	5	6	3	Z Accelerat	Acceleration along Z axis		
12	10	wod	Xmag	8	uT	8_bit_mag	Experiments	5	7	3	X Magneto	Angular Veolcity around X Axis		
13	11	wod	Ymag	8	uT	8_bit_mag	Experiments	5	8	3	Y Magneto	Angular Veolcity around Y Axis		
14	12	wod	Zmag	8	uT	8_bit_mag	Experiments	5	9	3	Z Magneto	Angular Veolcity around Z Axis		

The above example shows a number of 8-bit fields assigned to telemetry channels. These could also be 12-bit, 16-bit or other lengths. Where possible keep things aligned to byte boundaries until you are sure that the C compiler on your spacecraft does not do anything unexpected when packing the data. If in doubt, put the sub byte fields at the end. If the data is packed correctly then FoxTelem can extract it.

Data for each field must be transmitted with the MSBit first. FoxTelem also assumes that multi-byte values are little endian.

The spacecraft MASTER file must contain these two key value pair to define the payloads:

```
numberOfLayouts=X
hasFOXDB_V3=true
```

Where X is the number of payload layouts that you are using. Version 3 of the Fox DB is needed to support all the latest features defined in this document.

Each payload is then defined as follows:

```
layoutY.filename=filename
layoutY.name=name
layoutY.parentLayout=optional_parent_layout_name
layoutY.type=typeKeyword
layoutY.shortTitle=tab name
layoutY.title=title to display above the telemetry modules
```

Where Y is the payload number.

The parentLayout is only defined for secondary layouts. It is optional. These are used to convert experiment payloads into a separate format on the ground. This is often required to unravel another layer of the protocol used to transmit experiment data.

The typeKeyword is one of the following:

Type Keyword	Description
RT	Real Time data
MAX	Maximum readings of real time data
MIN	Minimum readings of real time data
WOD	Whole orbit data
EXP	Experiment data
WOD_EXP	Whole Orbit Experiment data
CAN_EXP	CAN bus experiment data
CAN_WOD_EXP	CAN bus whole orbit data
DBG	A debugging payload or other payload that will not be displayed in the GUI

The type keyword determines what type of telemetry tab the data is displayed on. e.g., MIN and MAX are not displayed on separate tabs. Secondary Payloads are displayed on the same tab as the primary payload.

In addition, there are some type keywords used to store data on the ground. These layouts are not transmitted. They allow FoxTelem to store experiment data converted on the ground into a second format.

Type Keyword	Description
CAN_PKT	Raw CAN packets
WOD_CAN_PKT	Raw Whole Orbit data CAN packets

So, in this example:

```
layout7.filename=LTM_diagnostictelemetry.csv
layout7.name=diagnostictelemetry
layout7.shortTitle=Diagnostics
layout7.title=AMSAT Diagnostics
layout7.type=EXP
```

We have defined layout 7. The details are in the file LTM_diagnostictelemetry.csv. The payload is named diagnostictelemetry. **This is the name that must be used in the Frame definition file**. The tab that will be displayed on the GUI will be called Diagnostics and will have a tile "AMSAT Diagnostics" above the telemetry modules. It is of Type EXP, meaning it will be an experiment data tab.

6 Field Conversions

When a telemetry channel is extracted from the telemetry frame we end up with a raw value. This is a positive integer from 0 to the maximum value supported by the number of bits. Internally FoxTelem stores these channels as 32-bit integers. If a value needs to be stored with more than 32 bits then a custom conversion will be needed. This is best achieved by defining the channel as 2 or more sub-fields with an equation to combine them.

The raw value typically is not in a human readable form. It is perhaps an Analog to Digital Converter (ADC) reading and we want a temperature or rotation rate. We can convert the value into a human readable form by specifying a conversion in the CONVERSION column of the payload layout. The conversion can be one of the following:

- Curve Conversion
- Mathematical equation
- Look up table
- String Lookup table
- A list of conversions in a pipeline
- Legacy conversion

6.1 Curve Conversions

Perhaps the most common conversion is a curve conversion. These are defined in a curve file which is formatted as follows:

	A	B	C	D	E	F	G	H
1	CurveName	a	bx	cx^2	dx^3	ex^4	fx^5	Description
2	com1_tx_fwd_pwr	-13.019	36.436	-13.107	1.7685	0	0	Forward power. Apply to the voltage value from the IHU lookup table
3	com1_tx_ref_pwr	-34.583	0.1727	0	0	0	0	Reflected power. Apply to the voltage value from the IHU lookup table
4	8_bit_temp	-20	0.5	0	0	0	0	gtemp RT1Temp and RT2Temp IHU supplies as 7.1 fixed point with offset of 20
5	golf-t_bus_voltage	0	0.0399	0	0	0	0	0-255 means 0-13.61V
6	8_bit_voltage	0	0.03026	0	0	0	0	3V Prot 2.5V prot VGA Control and others
7	golf-t_tx_current	0	7.557	0	0	0	0	Milliamps of Power Amp Current
8								

Each curve applies a simple equation to convert the value into a human readable form. e.g., the curve 8_bit_temp applies the formula 0.5x – 20, where x is the raw value.

To use conversion curves the spacecraft MASTER file must contain the key value pairs:

```
useConversionCoeffs=true
conversionCurvesFileName=conversion_curves_filename.csv
```

In this example the curves are stored in GOLF-T_conversion_curves.csv. You can use any file name that you like.

```
conversionCurvesFileName=GOLF-T_conversion_curves.csv
```

To apply the curve to a telemetry channel, the CurveName is placed in the CONVERSION column of the Payload Layout.

6.2 Mathematical Equations

A mathematical equation can be defined to convert a telemetry value or to calculate a value that requires multiple telemetry channels. The equations are defined in a Conversion Expressions file with the following format:

	A	B	C
1	ExpressionName	Expression	Description
2	ExpScalarRotation	sqrt(Xspin*Xspin + Yspin*Yspin + Zspin*Zspin)	Test 2
3	ExpMemsXRotationAngle	360 * acos(abs(Xspin / MemsScalarRotation)) / (2 * 3.14159)	
4	ExpMemsYRotationAngle	360 * acos(abs(Yspin / MemsScalarRotation)) / (2 * 3.14159)	
5	ExpMemsZRotationAngle	360 * acos(abs(Zspin / MemsScalarRotation)) / (2 * 3.14159)	
6	tx_pwr2	10 ^ (X/10)	Do the exp require for the power curve

For example, the ExpScalrRotation equation sums the squares each of the spin axis telemetry channels and then takes the squareroot.

Currently the following scientific functions are supported:

Function	Description
Sqrt()	Square root
Sin()	Sine
Cos()	Cosine
Tan()	Tangent
Acos()	Inverse cosine
Asin()	Inverse Sine
Atan()	Inverse Tangent
Abs()	Absolute value

To use conversion curves the spacecraft MASTER file must contain the key value pairs:

```
useConversionCoeffs=true
conversionExpressionsFileName=conversion_expressions_filename.csv
```

In this example the expressions are stored in GOLF-T_conversion_expressions.csv. You can use any file name that you like.

```
conversionExpressionsFileName=GOLF-T_conversion_expressions.csv
```

To apply the equation to a telemetry channel, the ExpressionName is placed in the CONVERSION column of the Payload Layout.

6.3 Lookup tables

A telemetry channel can be converted into a human readable value using a lookup table. This may be required if a value is not easily modelled with an equation. A lookup table is formatted in a file as follows:

	A	B
1	0	0
2	20	0.0304
3	621	1
4	1246	2
5	1876	3
6	2502	4
7	3127	5
8	3253	5.2
9	3372	5.4
10	3753	6
11	3875	6.2
12	3998	6.4
13	4095	6.6

This format defines the raw value and the value that corresponds to it. e.g., if the raw value is 1246 then the converted value will be displayed as 2.

The lookup table uses linear interpolation to return values that are in between the supplied data points.

To use lookup tables the spacecraft MASTER file must contain the key value pair:

```
numberOfLookupTables=X
```

Where X is the number of lookup tables that you are using.

Each lookup table is then defined in the MASTER file with two key value pairs as follows:

```
lookupTableY.filename=filename.tab
lookupTableY=name
```

Where Y is the number of the lookup table you are defining, starting at 0.

So, in this example:

```
lookupTable0.filename=FOX1A_rssiFM.tab
lookupTable0=RSSI
```

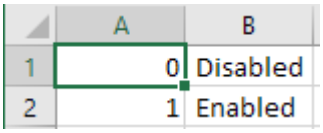
We have defined a lookup table in a file FOX1A_rssiFM.tab with the name RSSI.

To apply the lookup table to a telemetry channel the name is placed in the CONVERSION column of the Payload Layout. In the example above, the name is RSSI.

6.4 String Lookup tables

A telemetry channel can also be converted into a human readable value using a string lookup table. This allows a string value to be displayed from a numeric key. This is often the case with status fields.

A string lookup table is formatted in a file as follows:



	A	B
1	0	Disabled
2	1	Enabled

This table defines a raw value and a string value that will be displayed in its place.

To use lookup tables the spacecraft MASTER file must contain the key value pair:

```
numberOfStringLookupTables=X
```

Where X is the number of lookup tables that you are using.

Each lookup table is then defined in the MASTER file with two key value pairs as follows:

```
stringLookupTableY.filename=filename.tab
stringLookupTableY=name
```

Where Y is the number of the string lookup table you are defining, starting at 0.

So, in this example:

```
stringLookupTable0.filename=status_enabled.tab
stringLookupTable0=STATUS_ENABLED
```

We have defined a lookup table in a file status_enabled.tab with the name STATUS_ENABLED.

To apply the string lookup table to a telemetry channel the name is placed in the CONVERSION column of the Payload Layout. In the example above, the name is STATUS_ENABLED.

6.5 Conversion List

Sometimes one conversion is not enough. We can apply more than one conversion to a telemetry channel using a pipeline. Each conversion in the CONVERSION column of the payload layout is separated by a | symbol. This is the pipe symbol and represents feeding the output of one conversion as the input to the next.

In this way a value can be converted and then turned into a string using a string lookup table. Or two different conversions can be applied. e.g., a conversion may be defined for a 3 volt ADC that measures many telemetry channels. The final channels may use the conversion for the 3 volt ADC and then separate conversions for their curves or equations. This would be put in the conversion column as follows:

```
3_volt_ADC | plus_x_panel_temp
3_volt_ADC | plus_x_spin_value
```

6.6 Legacy Conversions

Historically the Fox spacecraft used conversions that were hard coded into FoxTelem. This made sense initially because the spacecraft were of very similar design and shared 90% of the telemetry channels. These legacy conversions are defined as an integer value. You can still use these integer values, but they are not recommended.

A list of legacy conversions can be found in this java class:
https://github.com/ac2cz/FoxTelem/blob/master/src/telemetry/BitArrayLayout.java
They are implemented in this java class:
https://github.com/ac2cz/FoxTelem/blob/master/src/telemetry/FramePart.java

7 Formatting

Once a telemetry channel is in human readable form we want to display it on the screen or in a graph. The final step is to format it. This defines how it is displayed on the telemetry tab.

7.1 Field Formatting

To format a field, we add one of the following reserved words to the CONVERSION column of the payload layout:

```
INT
FLOATx
BINx
HEXx
```

These reserved words are not case sensitive.

In the above examples x is the number of decimal points for the floating-point number, the number of bits for the binary number or the number of hex digits.

It is often convenient to add these formatting words as the final conversion in a conversion pipeline. e.g., we could put the following in the CONVERSION column:
8_bit_temp | float1

This would apply the 8_bit_temp conversion and then format it with 1 decimal place.

7.2 Timestamp Conversion

To format a field as a timestamp based on an epoch (or reset) and uptime stored in two separate fields, we can use the timestamp conversion function:

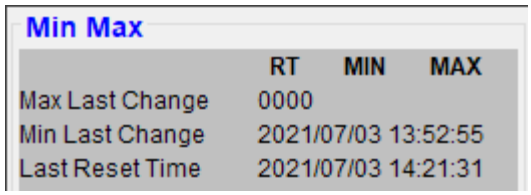
```
TIMESTAMP epochField uptimeField
```

The timestamp reserved word is not case sensitive.

In the above example epochField is the name of the field that contains the epoch number and the uptimeField is the name of the field that contains the uptime in seconds.

e.g., the conversion below can be applied to show the Last time the Min Payload changed. The spacecraft sends this time down in two fields. Epoch is sent down in minTimestampEpoch and uptime is sent down in minTimestampUptime. We can use the below conversion to display it in human readable form:

```
TIMESTAMP minTimestampEpoch minTimestampUptime
```



7.3 Telemetry Tabs

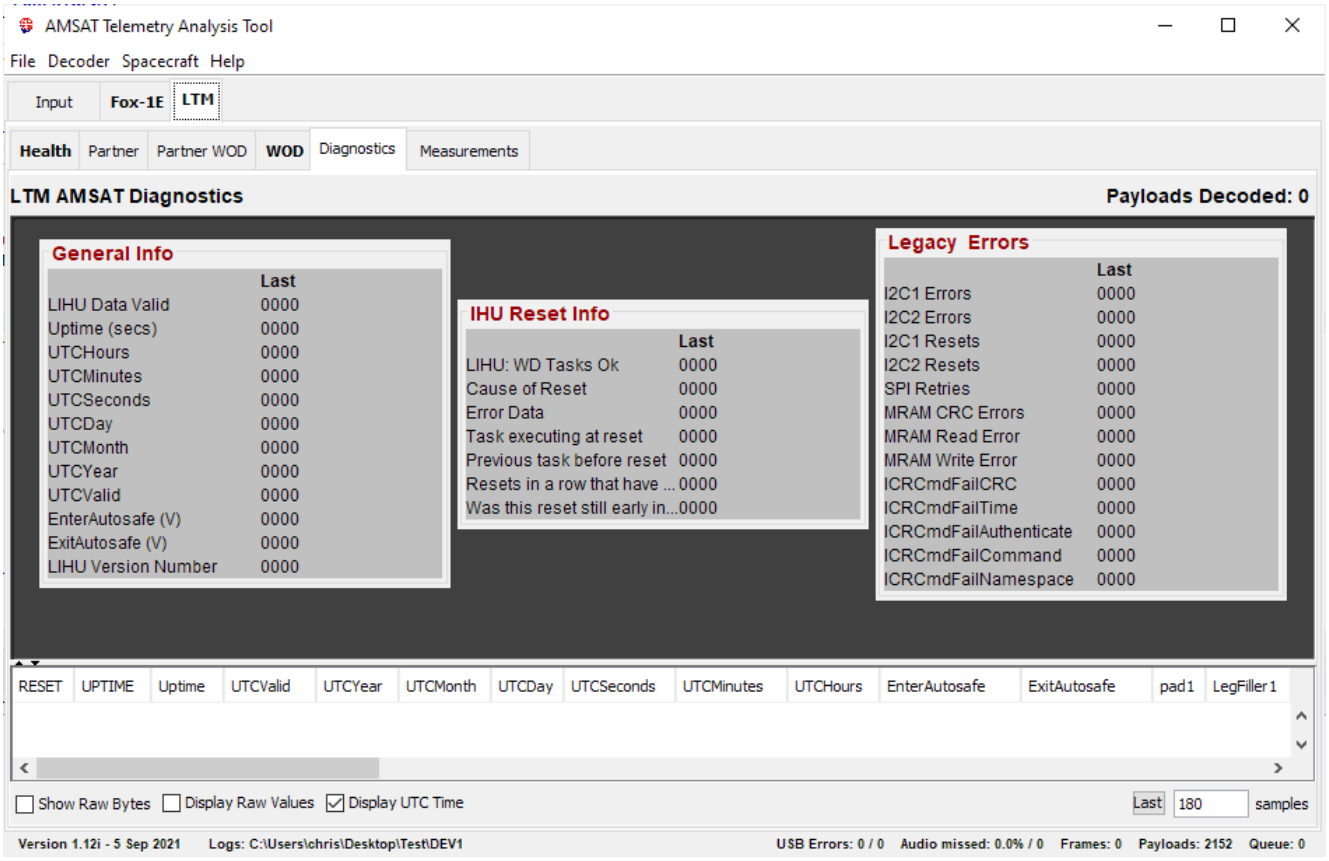
FoxTelem can display some or all the telemetry channels on a telemetry tab. The type of tab to use is defined in the MASTER file for each payload. This is discussed in the Payload Layout section above, where each payload can have the following key value pairs:

```
layoutY.filename=filename
layoutY.name=name
layoutY.parentLayout=optional_parent_layout_name
layoutY.type=typeKeyword
layoutY.shortTitle=tab name
layoutY.title=title to display above the telemetry modules
```

For example, if we define the following:

```
layout7.filename=LTM_diagnostictelemetry.csv
layout7.name=diagnostictelemetry
layout7.shortTitle=Diagnostics
layout7.title=AMSAT Diagnostics
layout7.type=EXP
```

Then we get a tab called Diagnostics with the title LTM AMSAT Diagnostics. This tab is defined as Experiment Data. This is the simplest type of tab. Raw data is simply displayed in the requested modules.



To define a module, such as "General Info" in the screenshot above, we add details to the MODULE columns in the Payload Layout. To define the "General Info" module we put the name "General Info" in the MODULE column for every channel we want to include. We also give General Info the MODULE_NUM (or number) 1. Note that module numbers start from 1 and not from 0.

Each channel is then numbered in the MODULE_LINE column to set the order that we want them to be displayed and is given a name in the SHORT_NAME column and a description in the final column.

	A	B	C	D	E	F	G	H	I	J	K
1	34	TYPE	FIELD	BITS	UNIT	CONVERSION	MODULE	MODULE_NUM	MODULE_LINE	LINE_TYPE	SHORT_NAME
2	0	diagnostic	Uptime	32	secs	INT	General Info	1	2	0	Uptime
3	1	diagnostic	UTCValid	1	-	STATUS_BOOLEAN	General Info	1	9	0	UTCValid
4	2	diagnostic	UTCYear	5	-	INT	General Info	1	8	0	UTCYear
5	3	diagnostic	UTCMonth	4	-	INT	General Info	1	7	0	UTCMonth
6	4	diagnostic	UTCDay	5	-	INT	General Info	1	6	0	UTCDay
7	5	diagnostic	UTCSecon	6	-	INT	General Info	1	5	0	UTCSeconds
8	6	diagnostic	UTCMinut	6	-	INT	General Info	1	4	0	UTCMinutes
9	7	diagnostic	UTCHours	5	-	INT	General Info	1	3	0	UTCHours
10	8	diagnostic	EnterAutc	8	V	golf-t_bus_voltage	General Info	1	10	0	EnterAutosafe
11	9	diagnostic	ExitAutos	8	V	golf-t_bus_voltage	General Info	1	11	0	ExitAutosafe
12	10	diagnostic	pad1	16	-	0	NONE	0	0	0	Fill to 32-bits
13	11	diagnostic	LegFiller1	5	-	0	NONE	0	0	0	pad to 32 bits
14	12	diagnostic	LegValid	1	-	STATUS_BOOLEAN	General Info	1	1	0	LIHU Data Valid

Each channel is also given a LINE_TYPE which is one of the following:

LINE_TYPE	Description
0	Display 1 column of real time data

1	Display the MAX value only
2	Display the MIN value only
3	Display real time, Max and Min
4	Display real time, Max and Min, but swap the Max and Min values
5	Display experiment data
8	Display WOD experiment data
9	Display Measurement data
10	Display Pass Measurement data
16	Display WOD data

There are also several legacy LINE_TYPES which should not be used.

If a description is defined in the final column then this is displayed as a tool tip when the mouse is moved over the field on the GUI.

7.4 Secondary Layouts on Experiment tabs

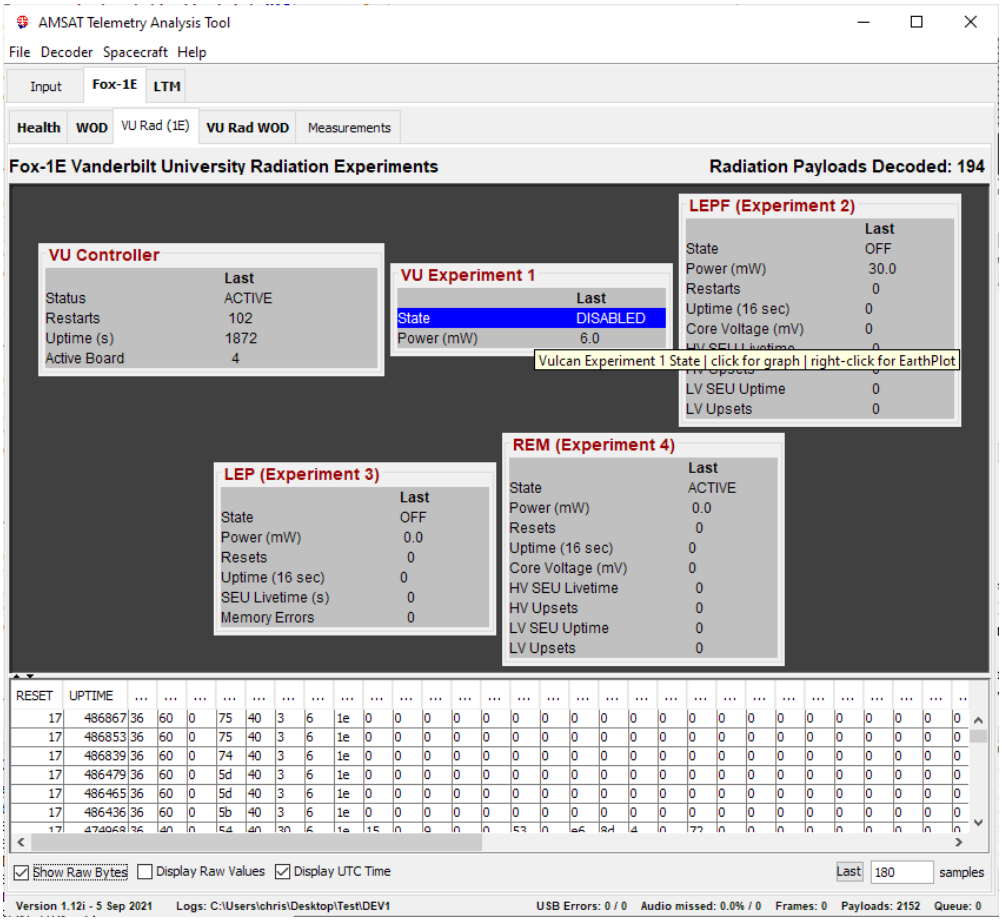
A payload of type EXP can have a secondary payload. This is defined in the MASTER file by giving the secondary payload a pointer to the parent payload. e.g., on FOX-1E we defined the following related payload layouts:

```
layout4.filename=FOX1E_radtelemetry.csv
layout4.name=radtelemetry
layout5.filename=FOX1E_radtelemetry2.csv
layout5.name=radtelemetry2
layout5.parentLayout=radtelemetry
```

Payload 4 is the data for the radiation experiment. It is named radtelemetry. Layout 5 is the secondary layout for that payload. Notice that the parentLayout for layout 5 is radtelemetry.

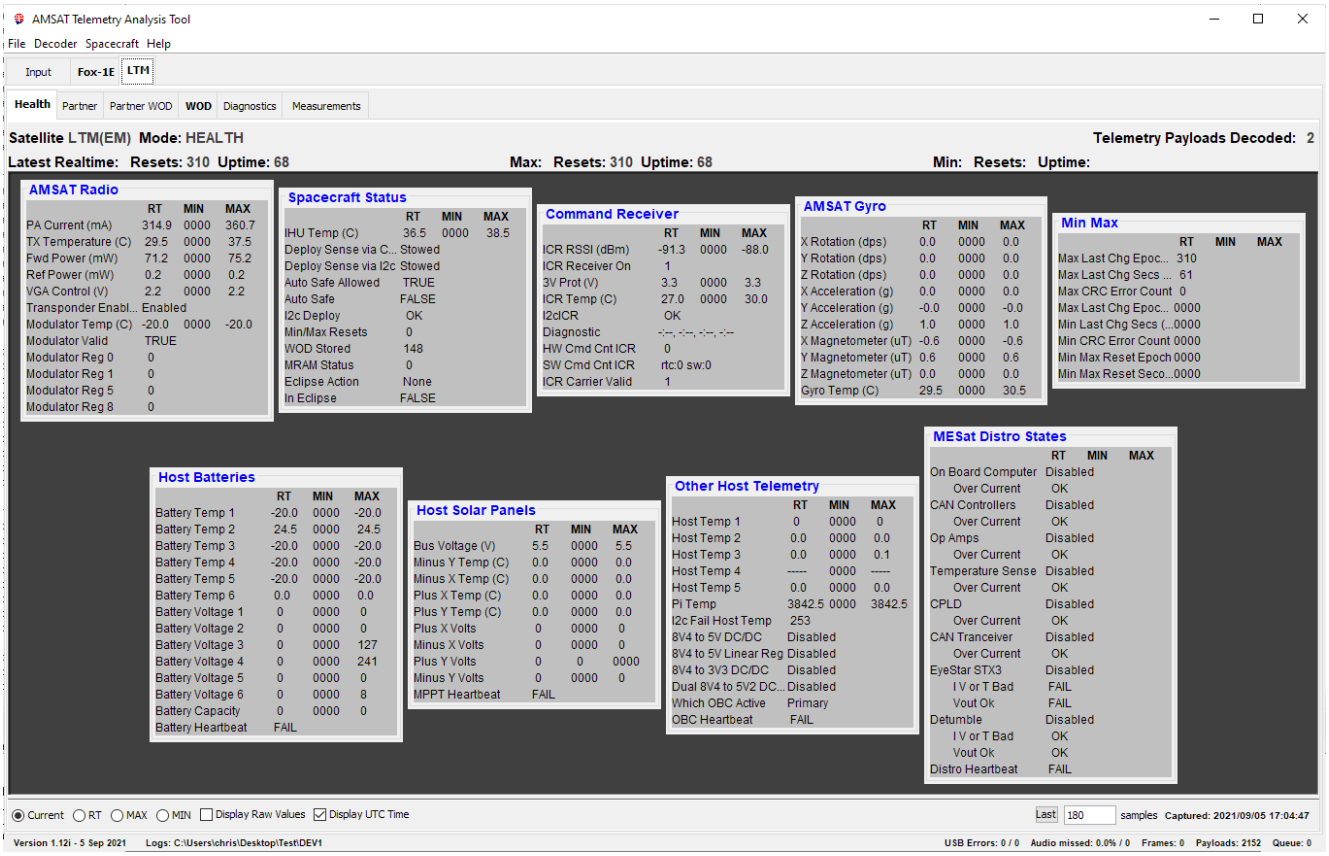
The secondary layout is then displayed on the experiment tab with the data formatted into the secondary payload at the top of the tab, as shown in the screenshot below. The data in the table at the bottom of the tab can be displayed as the parent layout when the “Show Raw Bytes” is checked or as the secondary layout when it is unchecked.

This approach allows experiment data to be received and stored as received, while allowing a secondary level of protocol to be processed so that human readable values can be displayed.



7.5 Health Tabs

At least one payload and tab must be defined as type RT in the master file. There should also be two payloads defined with the payload types of MAX and MIN. This data is then automatically combined and displayed together on the HEALTH tab:

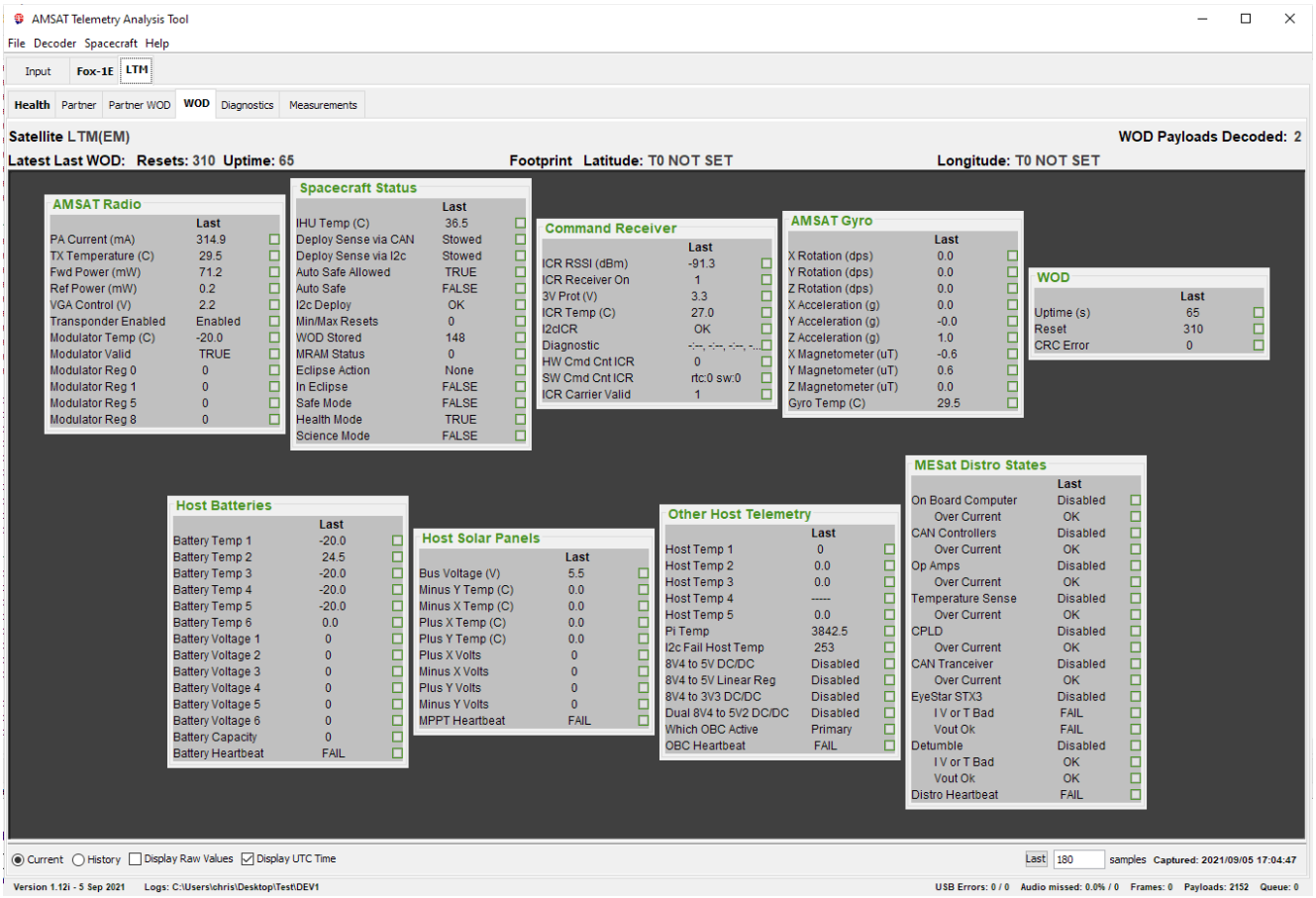


For each telemetry channel that you want to display you can decide to display just the real time value or a combination of real time with max and min. This is achieved by changing the LINE_TYPE as defined in the previous section.

The health tab has a radio button at the bottom that allows exploration of the real time, max and min data. Clicking the radio button opens a table of the latest data.

7.6 WOD Tabs

A whole orbit data tab is similar to the health tab but it does not combine max and min data. Each row only displays one value and contains a button that will open an earth plot. See the FoxTelem user manual for details.



Similar to health data, there is a radio button at the bottom of the screen that allows you to open a table of historical records.

7.7 CAN Tabs

A payload with the type CAN_EXP is automatically parsed into individual can packets. These are extracted and stored in a separate payload layout with the type CAN_PKT. In a similar way payloads with the type CAN_WOD are extracted and saved into the separate layout WOD_CAN_PKT.

The tab automatically displays both payloads. The CAN_EXP payloads are displayed in a table at the bottom of the window. The bytes are shown as they were received. The table at the top of the window shows the can packets (those parsed into the layout CAN_PKT) for the latest payload received or the payload that is highlighted in the table at the bottom. If modules are defined in the CAN_EXP or CAN_WOD layout then any channel data is displayed beside the can packets.