HDF5 telemetry log format (TLMC)

This file describes the content of the so-called `tlmc` format. `tlmc` stands for compressed telemetry: it is simply a standard HDF5 file with compression enabled, that can be opened with any HDF5 reader. This document specifies the organization of data in this file. The examples in this document are made using the Python `h5py` library ;

The telemetry of the robot outputs two different types of object: constants and variables. They must have a unique name and can have various basic types. Constants are (key, value) pairs and variables consists of two time series: one for time, one for values. Unlike constants, variables can have metadata associated to them.

The `tlmc` will be organized as follows:

- The root group shall contain an attribute 'VERSION', which stores an int specifying the version of the `tlmc` standard use. This document describes `VERSION=1`.
- The root group shall contain an attribute 'START_TIME', which stores a long specifying the absolute start time of the log, in second relative to the UNIX epoch.
- A group `constants` will store the telemetry constants in its attribute dictionary for scalar data of any type or as independent `constantName` 0D datasets without filter nor compression for null-padded byte arrays exclusively. The former only supports data of up to 64K bytes as per the HDF5 standard, while the latter as no size limitation and is convenient for pre-serialized data.
- A second group `variables` will store the variables.
  - Each subgroup `variableName` represents a variable, originally named `variableName`. Each variable group contains:
    - A `value` 1D dataset representing the variable's values through time.
    - A `time` 1D dataset representing the time instants relative to the 'START_TIME' file constant. This dataset will contain an attribute `unit` specifying the ratio to SI unit (i.e. 1 second). For instance when using nanoseconds, `file["variables/myvariable/time"].attrs["unit"]` evaluates to `1.0e-9`.
    - Variable-specific metadata stored in the group's attribute.

For storage efficiency, all datasets for variables will be stored using the 'gzip' filter with compression level of 4, and the 'shuffle' filter. The chunk size is equal to the number of timestamps to maxing-out reading performances. These are enabled in `h5py` using the following flags:

```
f.create_dataset(name,
                 data=data_array,
                 compression='gzip',
                 shuffle=True,
                 chunks=(len(data_array),))
```

Examples

Here is a (simplified) view of a `tlmc` file using the `h5dump -p`

```
HDF5 "data/20200921T101310Z_LogFile.tlmc" {
GROUP "/" {
   ATTRIBUTE "START_TIME" {
      DATATYPE  H5T_IEEE_F64LE
      DATASPACE  SCALAR
      DATA {
      (0): 1607002673
      }
   }
   ATTRIBUTE "VERSION" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SCALAR
      DATA {
      (0): 1
      }
   }
   GROUP "constants" {
      ATTRIBUTE "NumIntEntries" {
         DATATYPE  H5T_STD_I32LE
         DATASPACE  SCALAR
         DATA {
         (0): 1
         }
      }
      DATASET "HighLevelController.controlOffsetTimestamp" {
         DATATYPE  H5T_STRING {
            STRSIZE 8;
            STRPAD H5T_STR_NULLPAD;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         }
         DATASPACE  SCALAR
         DATA {
         (0): "1.680000"
         }
      }
      ...
   }
   GROUP "variables" {
      GROUP "HighLevelController.currentPositionLeftSagittalHip" {
         DATASET "time" {
            DATATYPE  H5T_STD_I64LE
            DATASPACE  SIMPLE { ( 338623 ) / ( 338623 ) }
            ATTRIBUTE "unit" {
               DATATYPE  H5T_IEEE_F64LE
               DATASPACE  SCALAR
               DATA {
               (0): 1e-09
               }
            }
         }
         DATASET "value" {
            DATATYPE  H5T_IEEE_F64LE
            DATASPACE  SIMPLE { ( 338623 ) / ( 338623 ) }
            STORAGE_LAYOUT {
               CHUNKED ( 338623 )
               SIZE 2708984 (1.000:1 COMPRESSION)
            }
            FILTERS {
               PREPROCESSING SHUFFLE
               COMPRESSION DEFLATE { LEVEL 4 }
            }
         }
      }
      ...
   }
}
}
```

Here is a python snippet for parsing a generic `tlmc` file:

```python
import h5py

file = h5py.File('my_file.tlmc', 'r')

print(file.attrs['VERSION']) # Prints 1
print("The log contains the following constants:")
for k, v in file['constants'].attrs.items():
    print(k, v)
for k, dataset in tlmc['constants'].items():
   v = bytes(dataset[()])
   v += b'\0' * (dataset.nbytes - len(v))  # Append missing null char '\0' at end
   print(k, v)
print(f"Log start time: {file.attrs['START_TIME']}")
print("The log contains the following variables:")
for variable_name in file['variables']:
    print(variable_name)
```