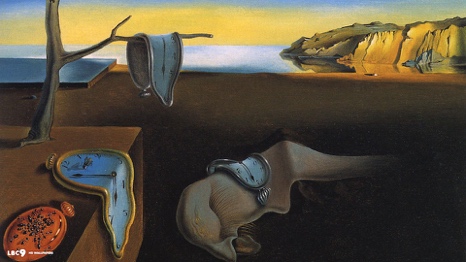
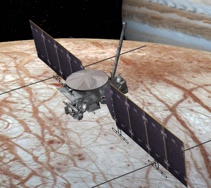
Multi-Mission Time Correlation

*User’s Guide*



August 18, 2023

Rev C for MMTC version 1.3.0

MGSS DOC-002610

Change Log

| **Revision** | **Submission Date** | **Affected Sections or Pages** | **Change Summary** |
| --- | --- | --- | --- |
| 1.0 | April 6, 2020 | All | Initial release |
| 1.1 | May 5, 2020 | Title Page, Table 1, Table 2 | Added document numbers |
| 1.5 | May 1, 2021 | Multiple revisions | Updated configuration parameters, TimeHistoryFile description, added discussion of clock change rates, other. |
| Rev A | June 7, 2021 | Multiple | Accompanies MMTC rel 1.1.0. |
| Rev B for Rel 1.2.1 | December 12, 2022 | Multiple, Table 6 Configuration parameters | For MGSS release 1.2.1 |
| Rev C for Rel. 1.3.0 | August 18, 2023 | Multiple | Updated for MMTC rel 1.3.0.  Corrected various descriptions, including the Consecutive Frame Filter and the logging configuration.  Attempted to clarify wording throughout. |

**Prepared at the Johns Hopkins Applied Physics Laboratory (JHU/APL) by:**

* Michael R. Reid, JHU/APL
* Andrew J. Smith, JHU/APL
* Eric C. Chang, JHU/APL
* Robert M. Seng, JHU/APL

Point of contact for questions or comments on the MMTC or this document:

**Robert Seng**

**Johns Hopkins Applied Physics Laboratory**  
11100 Johns Hopkins Rd.  
Laurel, Maryland 20723

Robert.Seng@jhuapl.edu

Table of Contents

List of Figures vi

1 Document Overview 1

1.1 Identification 1

1.2 Purpose 1

1.3 Overview 1

1.4 Terminology and Notation 1

1.5 References 1

2 Basics of Time Correlation 2

2.1 Determining the Clock Change Rate 3

3 Installing the MMTC 4

3.1 Installing from the RPM 5

3.2 Installing from the .tar.gz distribution archive 6

3.3 Building from Source 7

4 Adapting the MMTC to a Particular Mission 9

4.1 The start-up script 9

4.2 Telemetry Source Plugins 12

4.3 AMPCS-Based Ground Systems 12

4.4 Channelized Time Correlation Parameters 14

5 Running the MMTC 15

5.1 Arguments and Options 15

6 Inputs 18

6.1 Inputs 18

6.2 Environment Variables 19

7 Telemetry Source 21

7.1 AMPCS 21

7.2 Raw Telemetry Table 23

8 Configuration Files 23

8.1 Time Correlation Configuration File 23

8.2 Input SPICE Kernels 34

8.3 TK Packet Description File 36

8.4 SCLK Partition Map File 39

8.5 Ground Stations Map File 39

9 Time Correlation Data Tables 40

9.1 Raw Telemetry Table 40

9.2 Summary Table 40

10 MMTC Output Products 41

10.1 Output SCLK Kernel 41

10.2 SCLK/SCET File 41

10.3 Uplink Command File 42

10.4 Time History File 42

11 Filters 43

11.1 Contact Filter 43

11.2 SCLK Filter 43

11.3 ERT Filter 43

11.4 Data Rate Filters 44

11.5 Stations Filter 44

11.6 Valid Sample Filter 44

11.7 Consecutive Frames Filter 44

11.8 VCID Filter 45

12 Log File 45

13 Operating Environment 45

14 Support Concept 45

15 Appendix 46

A. Description of Time 46

B. Sample SCLK Kernel 46

C. Sample SCLK/SCET File 48

D. Sample Uplink Command File 48

E. Time History File Fields 49

F. The RawTlmTable 53

G. The Summary Table 54

H. Example TK Packet Definition File 57

I. Example Seed SCLK Kernel 58

J. External References 62

K. Acronyms and Glossary 62

List of Tables

[Table 1 Item identification. 1](#_Toc143250670)

[Table 2 Applicable Documents 2](#_Toc143250671)

[Table 3 Received Channelized telemetry values and the configuration parameters that specify their channel IDs. 15](#_Toc143250672)

[**Table 4 The Optional MMTC Command Line Options** 18](#_Toc143250673)

[Table 5 MMTC Environment Variables Overview of the System 19](#_Toc143250674)

[Table 6 Configuration Parameters ($TK\_CONFIG\_PATH/TimeCorrelationConfigProperties.xml) 31](#_Toc143250675)

[Table 7 AMPCS Plugin-Specific Configuration Parameters 33](#_Toc143250676)

[Table 8 The Time History File 52](#_Toc143250677)

[Table 9 The Raw Telemetry Table 53](#_Toc143250678)

[Table 10 The Summary Table 56](#_Toc143250679)

# List of Figures

[Figure 1 The MMTC start up script 9](file:////Users/reidmr1/Library/CloudStorage/Box-Box/Mike%20Shared/AMMOS/MMTC/MMTC_UsersGuide_RevC.docx#_Toc128575362)

[Figure 2 The mmtc start up script. 9](file:////Users/reidmr1/Library/CloudStorage/Box-Box/Mike%20Shared/AMMOS/MMTC/MMTC_UsersGuide_RevC.docx#_Toc128575363)

[Figure 3 High-Level View of the MMTC 17](#_Toc128575364)

[Figure 4 Overview of MMTC processing flow. 18](file:////Users/reidmr1/Library/CloudStorage/Box-Box/Mike%20Shared/AMMOS/MMTC/MMTC_UsersGuide_RevC.docx#_Toc128575365)

[Figure 5 Structure and contents of a TK Packet (from the Europa Clipper mission). 31](#_Toc128575366)

[Figure 6 Sample TK packet definition file. 32](file:////Users/reidmr1/Library/CloudStorage/Box-Box/Mike%20Shared/AMMOS/MMTC/MMTC_UsersGuide_RevC.docx#_Toc128575367)

[Figure 7 Sample SCLK Partition Map file 33](file:////Users/reidmr1/Library/CloudStorage/Box-Box/Mike%20Shared/AMMOS/MMTC/MMTC_UsersGuide_RevC.docx#_Toc128575368)

[Figure 8 The Ground Stations Map File 34](file:////Users/reidmr1/Library/CloudStorage/Box-Box/Mike%20Shared/AMMOS/MMTC/MMTC_UsersGuide_RevC.docx#_Toc128575369)

[Figure 9 Example TK Packet Definition File 50](file:////Users/reidmr1/Library/CloudStorage/Box-Box/Mike%20Shared/AMMOS/MMTC/MMTC_UsersGuide_RevC.docx#_Toc128575370)

# Document Overview

## Identification

| **Property** | **Value** |
| --- | --- |
| Configuration ID (CI) | 635.401 |
| Element | Mission Planning, Sequencing, and Analysis (MPSA) |
| Program Set | Multi-Mission time Correlation (MMTC) |
| Document Number | DOC-002610 |
| Version | 1.3 |

Table 1 Item identification.

## Purpose

The Multi-Mission Time Correlation (MMTC) User’s Guide describes the installation, configuration, and use of the MMTC application. The MMTC is intended for inclusion in space mission ground data systems. It is a mission-independent spacecraft time to ground time correlation system. All space missions need to maintain an accurate, and in many cases, an extremely accurate knowledge of time as measured onboard the spacecraft. The MMTC provides an automated way of maintaining knowledge of time and produces standard and widely-used time correlation products. This application is normally expected to run in a Mission Operations Center (MOC) or mission support area (MSA.)

## Overview

The MMTC is a component of NASA’s Advanced Multi-Mission Operations System (AMMOS). It is intended to run in a MOC or MSA as part of the mission’s Ground Data System (GDS). It functions as a stand-alone system which interfaces to a space mission’s telemetry archive. It does not run in real-time and is not dependent on any other applications.

## Terminology and Notation

See glossary in Appendix K.

## References

| **Title** | **Revision** | **Document Number** |
| --- | --- | --- |
| AMMOS Technical Standards Profile | A | DOC-001101 |
| MGSS Implementation and Maintenance Task Processes | B | DOC-001638 |
| MGSS Implementation and Maintenance Task Requirements (External) | B | DOC-001819 |
| MMTC Architecture Description Document |  | DOC-002607 |
| MMTC Concept of Operations |  | DOC-002608 |
| SCLK – *Reference for the SPICE spacecraft clock subsystem and the Spacecraft Clock Kernel (SCLK)*  https://naif.jpl.nasa.gov/pub/naif/toolkit\_docs/C/req/sclk.html | 5/27/2010 | N/A |
| Generic SCLK versus SCET Correlation File, Software Interface Specification, Rev. E, January 2012. | E | DOC-000574 |
| M. R. Reid and S. B. Cooper, “The Multi-mission Time Correlation System,” 2019 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT), Pasadena, CA, USA, 2019, pp. 41-46. Doi: 10.1109/SMC-IT.2019.00010 | N/A | N/A |
| *Acton, C.H.; “Ancillary Data Services of NASA’s Navigation and Ancillary Information Facility;” Planetary and Space Science, Vol. 44, No. 1, pp. 65-70, 1996.* | N/A | N/A |
| *Charles Acton, Nathaniel Bachman, Boris Semenov, Edward Wright; A look toward the future in the handling of space science mission geometry; Planetary and Space Science (2017);  DOI 10.1016/j.pss.2017.02.013 https://doi.org/10.1016/j.pss.2017.02.013* | N/A | N/A |

Table 2 Applicable Documents

# Basics of Time Correlation

Time correlation is the association of a time measured onboard the spacecraft with an independently computed ground time and a measurement of the rate that the spacecraft clock (SCLK) has drifted away from ground time.

SCLK

Spacecraft clock time (SCLK) is represented as a number of ticks since a defined starting ground time. An SCLK value consists of a coarse and a fine part in the form of unsigned integers. The coarse part of the SCLK is usually approximately one second in length. The fine part is a count of clock oscillations that occur in a cadence of a fraction of second. One can convert the fine part to a decimal fraction of a second by dividing it by the SCLK modulus, which is a fixed value for the clock. The coarse part of the SCLK is often called the “seconds” part and the fine part the “subseconds” part. SCLK can be represented as either a raw floating point number or as an “encoded SCLK.”

The raw form of SCLK is a regular floating point number, given in seconds, and is computed by *SCLKcoarse+(SCLKfine/modulus)*. The *modulus* is the fractional representation of a tick, also called the “SCLK resolution” or the “tick weight.”

Encoded SCLK is a way of representing SCLK defined for use with the NAIF SPICE library that assures an ever-increasing SCLK value regardless of clock rollovers or interrupts. The raw form of SCLK is written to the SCLK/SCET file. The MMTC writes the encoded form to the SCLK Kernel as an unsigned integer.

TDT(G)

Ground time is measured by reading the Earth Receive Time (ERT) attached to a telemetry frame containing an SCLK by the ground station. One-way light time (OWLT) and some smaller factors are then subtracted from the ground time to make it match the SCLK. The ground stations provide ERT as a UTC value; however, it is converted to Terrestrial Dynamical Time (TDT) for time correlation purposes. This can be given either as a calendar string or as a numeric second of epoch. The TDT representation of correlated ground time is represented in the MMTC as TDT(G). TDT(G) is written to the SCLK Kernel in association with an encoded SCLK

SCET

Spacecraft Event Time (SCET), in the MMTC context, is the UTC equivalent of TDT(G). It is computed independently of the SCLK from the ERT-derived TDT(G). This value represents the correlated ground time given as SCET0 in the SCLK/SCET file.

CLKRATE

This clock change rate (CLKRATE) can also be thought of as a measurement of much the length of an SCLK “tick” drifts over time due to imprecisions in the onboard clock. The clock rate compares the differences between the latest computation of TDT(G) and one from a previous run with the difference of the associated SCLKs from the current and previous runs. This value goes into both the SCLK Kernel and the SCLK/SCET file.

The SCLK, in either raw or encoded form; the ground time, as either TDT(G) or SCET; and the CLKRATE are the essential parameters needed to perform time correlation. The MMTC computes these values during each run. See *Reid & Cooper 2019* (Table 2) for a much more detailed and in-depth discussion of time correlation and the MMTC.

## Determining the Clock Change Rate

As stated above, a time correlation record contains three fundamental components: a spacecraft time, a ground time, and a clock change rate. The clock change rate value is part of the basic time correlation record in both the SCLK kernel and the SCLK/SCET file; it measures the differential between time as measured by the spacecraft clock as compared to an “absolute” time measured on the ground. There are no perfect clocks; therefore, time as measured onboard the spacecraft will drift away from that measured on the ground. This measurement of clock change rate is essential for adjusting for drift when converting SCLK to a ground time. The MMTC produces this value in one of four ways, as specified in either configuration parameters or command line options. Command line options supersede configuration parameters.

**Compute Predicted Clock Change Rate**

The MMTC computes the predicted clock change rate in TDT seconds per SCLK second using the following formula:

*CLKRATE = (TDT(G)1 – TDT(G)0)/(SCLK1 – SCLK0)*

Where:

* **TDT(G)1** is the ground time as measured in Terrestrial Dynamical Time (TDT) of the current target sample based on the sample Earth Received Time (ERT) minus one-way light travel time (OWLT) and some other adjustments.
* **TDT(G)0** is the ground time from a previous time correlation measured in TDT obtained from the input SCLK kernel looking back a number of records whose ground times are immediately before a number of hours as indicated in the **compute.tdtG.rate.predicted.lookBackDays** configuration parameter. Typically, this will correspond to several contacts back.
* **SCLK1** is the spacecraft clock value in ticks (SCLK) relative to a determined epoch of the current target sample as obtained either from a TK packet or transfer frame header.
* **SCLK0** is the SCLK value from the same previous SCLK Kernel record from which TDT0 was obtained.

Unless otherwise specified, the MMTC will compute the clock change rate for the current time correlation using this method and will write it to the end of the new SCLK kernel and SCLK/SCET file. This setting is specified by setting the default **compute.clkchgrate.mode** configuration parameter to **compute-predict**.

NOTE: The computation of Predicted CLKRATE depends on there being at least one prior time correlation record in the input SCLK Kernel. The MMTC subtracts the number of lookback days as given in the **compute.tdtG.rate.predicted.lookBackDays** configuration parameters from the time of the current target sample. This is the lookback time. It selects the record from the input SCLK Kernel that matches or immediately precedes the lookback time for use in computing the CLKRATE. If the next record that precedes the lookback time is earlier than the maximum lookback time specified in the **compute.tdtG.rate.predicted.maxLookBackDays**, the MMTC will terminate with an error unless it is running in test mode. If running in test mode, it will log a warning message and continue processing.

**Compute Interpolated Clock Change Rate**

The MMTC computes the interpolated clock change rate using the same formula as for the predicted clock change rate, except that **TDT0** and **SCLK0** are the values from the time correlation immediately prior to the current one (i.e., the last record in the input SCLK kernel). However, this interpolated clock change rate is written to that prior time correlation record, not the current one. The MMTC uses the predicted method to compute the clock change rate for the current time correlation, but then overwrites the clock change rate in the previous time correlation record in the SCLK kernel. The next time that the MMTC runs with interpolation on, it will overwrite this latest clock change rate in the current time correlation record with a new interpolated one. This method has been shown to provide a slightly more accurate value for clock change rate. This value is also written to the SummaryTable and the TimeHistoryFile. Since the MMTC creates the SCLK/SCET file from the SCLK kernel, the SCLK/SCET file will contain the same interpolated values for all but the very last time correlation, but rounded to one less digit of precision. This is the recommended method of computing the clock change rate and is the default if the **compute.clkchgrate.mode** parameter is not provided in the configuration file.

**Assign Clock Change Rate**

The MMTC allows the user to assign the value for the clock change rate that will be written to the SCLK kernel and SCLK/SCET file and the other products. If the **compute.clkchgrate.mode** configuration parameter is set to **assign**, then the value in the accompanying **compute.clkchgrate.assignedValue** configuration parameter will be used. This option should be used when there is a discontinuity in time correlations such as after an oscillator switch, after a radio switch (for mission in which the radio provides the SCLK values), or when an anomaly occurred that affected the spacecraft clock. Comparing SCLK values across time correlations that are based on different clocks is erroneous. In such cases, a time correlation engineer will usually externally estimate the clock change rate by manual computation.

**Set Clock Change Rate to No Drift**

The MMTC allows the user to set the clock change indicate that there was no drift between time correlations by setting the **compute.clkchgrate.mode** configuration parameter to **nodrift**. This would indicate a perfect clock, which does not exist in reality. This option is equivalent to assigning the clock change rate to 1.00000000000. There are circumstances when a time correlation engineer may determine that this is appropriate. The first record in an SCLK kernel, which is usually created by hand, normally sets the clock change rate to 1.00000000000.

# Installing the MMTC

These are the system requirements to run MMTC:

* RedHat Enterprise Linux (RHEL) 8
* Java 8 JRE

## Installing from the RPM

MMTC is distributed as an RPM for RHEL 8. This is the preferred way of installing MMTC.

The MMTC RPM is a relocatable RPM; by default, it will install MMTC into /opt/local/mmtc, but this can be overridden with the *rpm* command’s --prefix flag when installing the RPM.

The RPM produces the following directory structure under the installation path:

**/bin**

* **mmtc** – this is the script that is run to start the application (see section 4.1)

**/conf**

The path to this directory should be given in the environment variable **$TK\_CONFIG\_PATH**.

* **log4j2.properties** – configuration parameters for the message logger (see section 12). This file is specified as an option to the java JRE command itself.
* **properties.dtd** – defines the structure of the XML schema.
* **TimeCorrelationConfigProperties.xsd** – the XML schema that defines the time correlation parameters.
* **TimeCorrelationConfigProperties.xml** – the time correlation parameters file.
* **GroundStationsMap.csv** – the ground station map (see section 8.5). This file is indicated in the configuration parameter **groundStationMap.path** and can thus be named anything and located anywhere in the file system, but it is recommended to place it in this directory.
* **SclkPartitionMap.csv** – the SCLK partition map (see section 8.4). This file is indicated in the configuration parameter **sclkPartitionMap.path** and can thus be named anything and located anywhere in the file system, but it is recommended to place it in this directory.

**/lib**

* **mmtc-core-1.3.0-app.jar** – the Java jar file containing the compiled MMTC application.
* **/naif** – the directory where NAIF SPICE (via JNISpice) is bundled for use by MMTC
* **/plugins** – the directory where telemetry source plugins should be placed.
  + **mmtc-plugin-ampcs-1.3.0.jar** – a plugin for using AMPCS as the telemetry source.

**/log**

* **(mmtc.log)** – the log file created by the MMTC (this file will not exist until the first time the MMTC is run).

**/output**

* This is the directory where output files will be written. This directory is indicated in configuration parameters such as **spice.kernel.sclk.kerneldir** and **product.uplinkCmdFile.outputDir** and can thus be located anywhere in the file system.
* A seed SCLK kernel must also be placed here before the first time the MMTC is run.

Note that the MMTC RPM currently does not have special handling for configuration files. If configuration files already exist in the conf folder inside the destination directory, the RPM installation will overwrite them. Before installing the RPM over an existing directory, be sure to back up any custom, modified configurations files (such as TimeCorrelationConfigProperties.xml and log4j2.properties).

If a user named “mmtc” does not already exist, the RPM installation will also create the mmtc user and group. All files installed by the RPM are world-readable; however, only the mmtc user and members of the mmtc group can modify configuration files and add or create files (such as new log files or additional telemetry source plugins) in the conf, log, and lib/plugins directories.

After installing the RPM, do the following steps as needed:

* Add users to the mmtc group as needed.
* To use a telemetry source other than AMPCS, add an appropriate telemetry source plugin to the lib/plugins folder.
* Update the configuration files, in particular TimeCorrelationConfigProperties.xml, in the conf folder to suit the mission.
* Place a seed SCLK kernel file in the output folder.

MMTC can then be invoked from the command line. Change the working directory to the MMTC installation directory (/opt/local/mmtc by default), and then run:

bin/mmtc [options] <start-time> <stop-time>

See the rest of the User’s Guide for detailed command syntax.

## Installing from the .tar.gz distribution archive

MMTC is also distributed as a .tar.gz archive file. This method of installation can be used when RPM installation is not possible; it also gives more flexibility regarding where MMTC-related folders can be placed.

Extracting the archive file produces the directory structure depicted below. This is a recommended directory structure. Inside the top-level directory are the following:

**/bin**

* **mmtc** – this is the script that is run to start the application (see section 4.1).

**/conf**

* **log4j2.properties** – configuration parameters for the message logger (see section 12). This file is specified as an option to the java JRE command itself.
* **properties.dtd** – defines the structure of the XML schema.
* **TimeCorrelationConfigProperties.xsd** – the XML schema that defines the time correlation parameters.
* **TimeCorrelationConfigProperties.xml** – the time correlation parameters file.
* **GroundStationsMap.csv** – the ground station map (see section 8.5). This file is indicated in the configuration parameter **groundStationMap.path** and can thus be named anything and located anywhere in the file system, but it is recommended to place it in this directory.
* **SclkPartitionMap.csv** – the SCLK partition map (see section 8.4). This file is indicated in the configuration parameter **sclkPartitionMap.path** and can thus be named anything and located anywhere in the file system, but it is recommended to place it in this directory.

**/lib**

* **mmtc-core-1.3.0-app.jar** – the Java jar file containing the compiled MMTC application.
* **/naif** – the directory where NAIF SPICE (via JNISpice) is bundled for use by MMTC
* **/plugins** – the directory where telemetry source plugins should be placed.
  + **mmtc-plugin-ampcs-1.3.0.jar** – a plugin for using AMPCS as the telemetry source.

**/log**

* **(mmtc.log)** – the log file created by the MMTC (this file will not exist until the first time the MMTC is run).

**/output**

* This is the directory where output files will be written. This directory is indicated in configuration parameters such as **spice.kernel.sclk.kerneldir** and **product.uplinkCmdFile.outputDir** and can thus be located anywhere in the file system.
* A seed SCLK kernel must also be placed here before the first time the MMTC is run.

After the archive file is extracted, some folders and files can be moved elsewhere in the filesystem to suit the mission:

* The **conf** folder can be relocated anywhere; in fact, it does not need to be used at all, and most files inside it can be relocated anywhere. The exceptions are TimeCorrelationConfigProperties.xml, TimeCorrelationConfigProperties.xsd, and properties.dtd, all three of which must be placed in the same folder. After relocating the folder or any files, simply modify the following:
  + Ensure that TimeCorrelationConfigProperties.xml, TimeCorrelationConfigProperties.xsd, and properties.dtd are located in the same folder. Set the **$TK\_CONFIG\_PATH** environment variable to the path of that folder.
  + In the **TimeCorrelationConfigProperties.xml** file, set the **groundStationMap.path** and **sclkPartitionMap.path** properties to specify the paths to **GroundStationsMap.csv** and **SclkPartitionMap.csv**, respectively.
  + In the bin/mmtc script, modify the line  
    “-Dlog4j.configurationFile=${MMTC\_HOME}/conf/log4j2.properties \”  
    with the correct path to log4j2.properties.
* The **lib/plugins** folder can be relocated anywhere; simply modify the **TimeCorrelationConfigProperties.xml** file’s **telemetry.source.pluginDirectory** property to specify the path to the folder.
* The **log** folder can be located anywhere; simply modify the **log4j2.properties** file’s **property.basedir** property to specify the path to the folder.
* There is no requirement for all output products to be written to the same **output** folder. Simply ensure that the various output path properties in **TimeCorrelationConfigProperties.xml** point to the desired output paths.

MMTC can then be invoked from the command line. Change the working directory to the folder where the archive file was extracted, and then run:

bin/mmtc [options] <start-time> <stop-time>

See the rest of the User’s Guide for detailed command syntax.

## Building from Source

MMTC can also be built from source. The MMTC release includes a source distribution bundle that contains a Gradle project to build the MMTC application, as well as sample configuration files and test data.

The requirements to build MMTC are:

* JDK 8
* Gradle 8.2.1

To build MMTC:

1. tar xzf mmtc-1.3.0.tar.gz
2. cd mmtc-1.3.0
3. ./gradlew clean build

To build the MMTC RPM, additionally run:

1. ./gradlew mmtcRpm

The build automatically produces a Javadoc jar file as well. To examine the Javadoc, extract the jar contents:

1. mkdir javadoc
2. cd javadoc
3. jar -xf ../target/mmtc-1.3.0-javadoc.jar
4. cd ..

Then browse the resulting Javadoc HTML files in the **javadoc** folder.

In order to run MMTC after it has been built, set up the runtime environment:

1. Untar the JNISpice and CSPICE package:
   1. mkdir -p naif/JNISpice
   2. tar xzf lib/JNISpice-N0066-el7-gcc-4.8.5-Java-1.8.tar.gz -C naif/JNISpice
2. Create a configuration folder and an output folder:
   1. mkdir conf
   2. mkdir output
3. Set environment variables:
   1. Set the **$MMTC\_HOME** environment variable to the path to mmtc-1.3.0 (the same directory that mmtc-1.3.0.tar.gz was extracted to)
   2. Set the **$TK\_CONFIG\_PATH** environment variable to the path of the configuration folder from step (2a); for example, if you created the configuration folder inside of mmtc-1.3.0 as suggested in step (2a), then set the environment variable to **$MMTC\_HOME/conf**
   3. Also see section 6.2 for additional environment variables that may need to be set
4. Copy configuration schema files into the configuration folder:
   1. cp src/main/resources/TimeCorrelationConfigProperties.xsd $TK\_CONFIG\_PATH
   2. cp src/main/resources/properties.dtd $TK\_CONFIG\_PATH
   3. cp src/main/resources/tk\_packet.xsd $TK\_CONFIG\_PATH
5. If in step (2b) you created an output folder with a different name or at a different location, then modify the MMTC configuration file, **conf/TimeCorrelationConfigProperties.xml**, to update affected path values. **NOTE:** These configuration values cannot contain environment variables. Use absolute paths instead.
   1. Change the following output path configurations to the output folder from step (2b):
      1. **spice.kernel.sclk.kerneldir**
      2. **product.sclkScetFile.dir**
      3. **product.uplinkCmdFile.outputDir**

For example, if you created an output folder **/custom/output/folder** in step (2b), then the **spice.kernel.sclk.kerneldir** configuration line should be changed from  
 <entry key=”spice.kernel.sclk.kerneldir”>**./output/sclk**</entry>  
to  
 <entry key=”spice.kernel.sclk.kerneldir”>**/custom/output/folder**</entry>

* 1. Change the following output file configurations, replacing the path prefix with the output folder from step (2b):
     1. **table.rawTelemetryTable.uri**
     2. **table.summaryTable.uri**
     3. **table.timeHistoryFile.uri**

For example, if you created an output folder **/custom/output/folder** in step (2b), then the **table.rawTelemetryTable.uri** configuration line should be changed from

<entry key=”table.rawTelemetryTable.uri”> **file:///opt/local/mmtc/output**/RawTlmTable.csv</entry>  
to

<entry key=”table.rawTelemetryTable.uri”>**file:///custom/output/folder**/RawTlmTable.csv</entry>

1. (Optional) MMTC ships with sample data from New Horizons, allowing users to quickly run MMTC and examine example file formats and results. To use this sample data, make the following additional changes.
   1. Copy the initial kernel file into the output folder:
      1. cp new-horizons-sample-data/kernels/sclk/new-horizons\_1454.tsc **/path/to/outputs**

replacing **/path/to/outputs** with the actual path to the output folder you created in step (2).

* 1. Modify the following values in the configuration file, replacing the path prefix with the path of the mmtc-1.3.0 folder. Again, note that these values cannot use environment variables such as $MMTC\_HOME.
     1. **groundStationMap.path**
     2. **sclkPartitionMap.path**
     3. **telemetry.source.uri**
     4. **spice.kernel.lsk.path**
     5. **spice.kernel.spk.path**
     6. **spice.kernel.fk.path**
     7. **spice.kernel.pck.path**

For example, if your mmtc-1.3.0 folder is located at **/home/myusername/mmtc-1.3.0**, then the groundStationMap.path configuration line should be changed from

<entry key=”groundStationMap.path”> **/opt/local/mmtc/conf**/GroundStationMap.csv</entry>

to

<entry key=”groundStationMap.path”>**/home/myusername/mmtc-1.3.0**/new-horizons-sample-data/mission-data/GroundStationMap.csv</entry>

Once the environment has been configured, MMTC can be invoked from the command line. Change the working directory to the mmtc-1.3.0 folder, then run the following:

bin/mmtc [options] <start-time> <stop-time>

See the rest of the User’s Guide for detailed command syntax.

# Adapting the MMTC to a Particular Mission

The MMTC is highly configurable and every effort was made to address mission-specific items via either configuration data or telemetry source plugins. This section discusses how to adapt MMTC for a particular mission.

## The start-up script

The **./bin/mmtc** bash wrapper script is used to invoke the MMTC application. Along with running the application JAR and passing along command line arguments, it sets certain Java system properties based on environment variables described above, namely the location of the SPICE library and the location of the log4j2.properties file that controls message logging. A user might need to edit this script for their particular site.

#!/bin/bash

MMTC\_BIN\_DIR=$( cd -- "$( dirname -- "${BASH\_SOURCE[0]}" )" &> /dev/null && pwd )

set -e

if [[ -z "${JAVA\_HOME+x}" ]]

then

echo "JAVA\_HOME environment variable not set. Please set it to a valid JRE/JDK 1.8 installation path."

exit 1

fi

if [[ -z "${MMTC\_HOME+x}" ]]

then

curdir=`pwd`

cd $MMTC\_BIN\_DIR/..

export MMTC\_HOME=`pwd`

cd $curdir

echo "MMTC\_HOME environment variable not set; defaulting to ${MMTC\_HOME}"

fi

if [[ -z "${TK\_CONFIG\_PATH+x}" ]]

then

export TK\_CONFIG\_PATH="${MMTC\_HOME}/conf"

echo "TK\_CONFIG\_PATH environment variable not set; defaulting to ${TK\_CONFIG\_PATH}"

fi

exec ${JAVA\_HOME}/bin/java \

-jar \

-Djava.library.path=${MMTC\_HOME}/lib/naif/JNISpice/lib \

-Dlog4j.configurationFile=${MMTC\_HOME}/conf/log4j2.properties \

${MMTC\_HOME}/lib/mmtc-core-1.3.0-app.jar "$@"

Figure 1 The MMTC start up script

Figure 1 gives the mmtc start up bash script. It is recommended to use this script as-is to launch MMTC; however, the script can be modified as needed for a particular installation or for a different shell. Note that the script depends upon the standard **$JAVA\_HOME** environment variable being already set; on the other hand, if the custom **$MMTC\_HOME** and **$TK\_CONFIG\_PATH** environment variables are not already set, it will derive values for them relative to the current working directory. (As a consequence, users should navigate to the MMTC installation directory, not to the MMTC bin directory, before running MMTC.) Also note that the script specifies to look for the Log4j 2 logging configuration file at **$MMTC\_HOME**/conf/log4j2.properties; however, as with the rest of the script, this can be edited as needed.

Figure 2 The mmtc start up script.

## Telemetry Source Plugins

The MMTC uses a plugin architecture. It is expected that the “core” of MMTC can be reused from mission to mission and will generally not need to be modified. Each mission will select or develop a plugin specific to the telemetry source that the mission wishes to use.

To develop a new plugin, a developer needs to implement the ***TelemetrySource*** interface, returning a list of **FrameSample** objects that contain information about the timekeeping samples that match the MMTC core’s query. This must be packaged in a jar file, along with a META-INF/services folder containing a service provider configuration file that identifies the class(es) that implement the ***TelemetrySource***. MMTC ships with an AMPCS telemetry source plugin, whose source code also serves as an example of how to implement a plugin.

To use a telemetry source, the corresponding plugin must be placed in the plugins folder. In MMTC’s TimeCorrelationConfigProperties.xml configuration file, the **telemetry.source.pluginDirectory** configuration option must then specify the path of the plugins folder, the **telemetry.source.pluginJarPrefix** option must specify the plugin jar file’s basename, and the **telemetry.source.name** option must specify the name of the class that implements ***TelemetrySource***. For example, to use the AMPCS plugin, these options might be set as follows:

<entry key="telemetry.source.name">AmpcsTlmArchive</entry>

<entry key="telemetry.source.pluginDirectory">/opt/local/mmtc/lib/plugins/</entry>

<entry key="telemetry.source.pluginJarPrefix">mmtc-plugin-ampcs-1.3.0</entry>

If the telemetry source plugin requires any configuration options of its own, these are also set in MMTC’s TimeCorrelationConfigProperties.xml file. By convention, plugin-related options all have names that begin with “telemetry.source.plugin.”

## AMPCS-Based Ground Systems

The MMTC ships with a plugin to use AMPCS as the telemetry source. It integrates with the AMPCS telemetry archive using AMPCS’s *chill\_get\_packets* and *chill\_get\_chanvals* interfaces. It assumes that time correlation (TK) packets are in the archive. It queries for all of the TK packets based on APID within the interval specified by the user and captures the essential metadata associated with each packet (e.g., ERT, VCID, VCFC, etc.). It then reads and decommutates the binary files containing the individual TK packets, extracting the SCLK and associated information from each. The resulting data is input to the time correlation computations.

The MMTC queries separately for the temperature of the active oscillator, as this is important in assessing the health of the oscillator. It also queries for the current SCLK, ground time in Terrestrial Dynamical Time (TDT), and clock change rate values that the onboard software uses for computing time. These values are originally uploaded to the spacecraft and are already known on the ground, but it is important to assure that they have not been corrupted.

### Code Customization

Class **AmpcsTlmArchive** andClass **AmpcsTlmWithFrames**

For missions based on the AMPCS core T&C system, telemetry interface is abstracted and contained within the **AmpcsTlmArchive** and the **AmpcsTlmWithFrames** classes, which derive from the ***AmpcsTelemetrySource*** abstract class, which implements the ***TelemetrySource*** interface. A mission will use either the **AmpcsTlmArchive** or the **AmpcsTelemetrySource** class, never both. The class used is specified in the **telemetry.source.name** configuration parameter. The *AmpcsTlmArchive.getSamplesInRange*() function reads the TK packet data, parses it, and inserts it into a list of FrameSample class objects. This function may need to be customized for a particular mission. The rest of the MMTC has no knowledge of the source of telemetry.

The *AmpcsTlmArchive.getGncTkParms*() function gets the SCLK, TDT, and clock change rate associations that the flight system uses to compute absolute time onboard. It also queries for the latest SCLK/TDT association computed. The MMTC assumes that each of the values are channelized. The Channel IDs are specified in configuration parameters. If a given mission provides the data differently, this function will require modification, resulting in the customization of code.

The *AmpcsTlmArchive.getOscillatorTemperature*() function gets the latest temperature of the active oscillator. The MMTC assumes that this value is available in channelized telemetry and is associated with a specific Channel ID which can be specified in configuration parameters. The configuration parameter **telemetry.channelId.oscillatorTemperature** provides the Channel Id(s) of the temperature for each oscillator. The value of this parameter is a comma-separated list. If there are multiple oscillators, the channel IDs corresponding to each oscillator are provided in this parameter in order. In other words, the channel ID of Oscillator 1 will be the first given, followed by that of Oscillator 2, etc. If a given mission provides the values through any method other than in single channels, the function will require modification, resulting in the customization of code.

The *AmpcsTlmArchive.getActiveOscillator*() function returns the identifier (e.g., “1” or “2” or “A” or “B”) of the oscillator onboard the spacecraft that is currently providing SCLK values. For spacecraft with only one oscillator, the value is always “1”. For spacecraft with two or more oscillators, this function will require specific implementation, since the means by which the current active oscillator may be determined is highly mission-specific. By default, the MMTC assumes two oscillators and assumes that TK packets received on VCID 0 are oscillator 1 and those on VCID 32 are oscillator 2. Obviously, this will have to be adapted to specific missions. The function *AmpcsTlmArchive.getActiveOscillator*() can be overridden.

The *AmpcsTlmArchive.getRadioId*() function returns the identifier of the currently active radio (e.g., 1 or 2). This is important in the case of missions (e.g., Europa Clipper) where the radio plays a role in time correlation. This is nominally indicated in the **spacecraft.activeRadioId** configuration parameter. It is assumed that the operations personnel know which radio is active prior to running the MMTC and will set this parameter whenever that changes. The *AmpcsTlmArchive.getRadioId*() function can be overridden. This information is used only in the TimeHistoryFile.

### Assumptions

Time Correlation Packets

The spacecraft flight system creates Time Correlation (TK) packets that contain the SCLK at which a preceding frame was created and radiated. (We refer to these as “TK” packets rather than “TC” so that they will not be confused with “telecommand” packets.) The TK packets are stored in the AMPCS telemetry archive and can be queried using the *chill\_get\_packets* command. The MMTC requires that there be at least one more TK packet in the contact interval than the size of the sample set given in the configuration parameter **telemetry.samplesPerSet**. In other words, if **telemetry.samplesPerSet**=10, there must be at least 11 consecutive TK packets in the interval. If there are not a sufficient number of TK packets within the specified query interval to form a sample set, the MMTC will terminate with an error.

Active Oscillator

Some spacecraft have multiple oscillators (i.e., clocks). Only one of these oscillators is active at a given time and is the one that is supplying SCLK values to the TK packets or the frame headers. The TimeHistoryFIle indicates which oscillator is currently active in the **Oscillator** field. This value is mission-specific and is usually a ‘1’ or ‘2’ or ‘A’ or ‘B’. When the active oscillator changes, it is advisable to set the clock change rate method to ‘Assign’ with an externally computed value or to ‘No Drift’, since computing the clock change rate between oscillators is not meaningful.

Oscillator Temperature

The temperature of the active oscillator (i.e., the spacecraft oscillator that is reporting SCLK counts) is downlinked in channelized telemetry (**telemetry.source.plugin.ampcs.channelId.  
oscillatorTemperature**) and can be queried by ERT using the *chill\_get\_chanvals* command. This information can be included in any channelized packet, so long as it has an assigned Channel ID defined in the flight software dictionary. The data is included in the Time History File. If the data item is not available in a given query interval, the MMTC will continue without it and will write a dash (‘-‘) indicating null value to its field in the Time History File.

Switching Oscillators

Some missions will have two or possibly more oscillators. Others will have only one. When there is an oscillator switch, the clock change rate compute method should be set to either ‘Assign’ or ‘No Drift’ for the next MMTC run. A meaningful clock change rate cannot be computed from data from different oscillators. Depending on the thresholds set, the new change rate might fail the Contact Filter [11.1]. If so, the user should turn off that filter in the configuration parameters for that run or possibly for the next several depending on the setting of configuration parameter **compute.tdtG.rate.predicted.lookBackDays.** The clock change rate method can be reset to Predicted or Interpolate for subsequent runs.

Onboard Time Correlation Data

The SCLK, TDT(S), and clock change rate values that the flight system uses to compute absolute time from SCLK values are available in channelized telemetry and can be queried by ERT using the *chill\_get\_chanvals* command. This information can be included in any channelized packet, so long as they have assigned Channel IDs defined in the flight software dictionary. The data is included in the Time History File. If the data is not available in a given query interval, the MMTC will continue without them and will write null values to the Time History File. The associated Channel IDs are included in Table 3.

### The Time Correlation Packet

A set of Time Correlation (TK) packets received from spacecraft telemetry is the primary input to the MMTC and contains the fundamental data needed to perform time correlation. At minimum, each packet contains the coarse SCLK and fine SCLK of a previous (target) frame and the VCID and VCFC of the target frame. The frame carrying the TK packet from which SCLK is read is called the supplemental frame. The SCLK is actually associated with a previous frame called the target frame. The supplemental frame usually, but not always, immediately follows the target frame in the telemetry sequence. In other words, its VCFC is one greater than that of the target frame. However, since this is not always the case, the number of frames that separate the target and supplemental frames may be specified in configuration parameter **telemetry.supplementalSampleOffset**.

As in the case of the Europa Clipper mission, the TK packet may also contain a validity flag, the frame encoding method, and the downlink data rate.

The structure of the TK packet is defined in the TK Packet Description file. This is an XML file that conforms to the **tk\_packet.xsd** schema (see section 8.3). This packet defines the bit offsets from the beginning of the packet of each field and their lengths in bits. This file must be customized for each mission. The name and location of this file is given in **telemetry.source.plugin.ampcs.tkpacket.tkPacketDescriptionFile.uri** within the configuration parameters.

## Channelized Time Correlation Parameters

As described above, the MMTC expects certain ancillary information to be available in channelized telemetry where each data item is associated with a Channel ID also called a “Telemetry Point” defined in the flight software telemetry dictionary. The Channel IDs are set in the configuration parameters. These items are given in Table 3 below. It is expected that the SCLK1, TDT1, and CLKCHG1 values will be downlinked in the same packet, as the values are tightly associated with one another.

Table 3 Received Channelized telemetry values and the configuration parameters that specify their channel IDs.

| **Item** | **Data Type** | **Config Parm (all are prefixed by “telemetry.source. plugin.ampcs.channelId”** | **Description** |
| --- | --- | --- | --- |
| SCLK1 | Float | **sclk1** | The current SCLK that the flight software onboard the spacecraft uses to derive ground time in TDT from SCLK values. This value is periodically uploaded to the spacecraft from the ground. |
| TDT1 | Float | **tdt1** | The ground time in TDT corresponding to TDT1. This value is periodically uploaded to the spacecraft from the ground. |
| CLKCHG1 | Float | **tdtChgRate** | The clock change rate in SCLK seconds/TDT seconds that corresponds to SCLK1 and TDT1. |
| GNC SCLK | Float | **gncsclk** | The latest SCLK from which an onboard time correlation was computed within the query interval. |
| TDT(S) | Float | **tdts** | The ground time as computed onboard the spacecraft in TDT corresponding to GNC SCLK. |
| Oscillators Temperature | Float | **oscillatorTemperature** | The temperature of the oscillator(s). There can be more than one, so this is a list of channel Ids. |

# Running the MMTC

The MMTC is invoked by running the startup script described in section 4.1. There is no GUI and it does not interact with the user after starting. Once started, its operation is controlled by the configuration parameters.

## Arguments and Options

The MMTC takes two required arguments, a start time and a stop time in Earth Received Time (ERT) for the data to query from its telemetry source. These are calendar string values in UTC and in ISO

*yyyy*-mm-*dd***T***hh*:*mm*:*ss*.*nnn* or *yyyy*-*doy***T***hh*:*mm*:*ss*.*nnn*

where *yyyy* is the year, mm the month, *dd* the day of the month, *hh* the hour of the day, *mm* the minutes, *ss*, the seconds, and *n* the fraction of second. All times are given in the 24-hour clock. *Doy* is the day of the year (1-366).

Besides the two required arguments, the MMTC also accepts additional optional arguments described in **Table 4**. The MMTC is started as follows:

**bin/mmtc** <*start-time*> <*stop-time*> [options]

or

**bin/mmtc** [options] <*start-time*> <*stop-time*>

For example:

**bin/mmtc 2017-04-03T00:00:00.0 2017-04-04T00:00:00.0 --clkchgrate-compute I --generate-cmd-file -K 11**

The above example queries AMPCS session ID 11 for 24 hours of telemetry over the entirety of April 4, 2017. It will compute a predicted clock change rate for the newest time correlation and replace the previous time correlation with an interpolated clock change rate. It will generate an uplink command file from the newly computed time correlation.

**bin/mmtc 2019-183T02:14:00.0 2019-183T04:16:00.0 --clkchgrate-compute p -K 16**

The above example queries 2 hours and 2 minutes of telemetry beginning on July 2, 2019 (doy 183, non-leap year) at 02:14 UTC from AMPCS session ID 16. It will compute the clock change rate for the new time correlation, but will not overwrite the previous one with a new interpolated value. It will not create an uplink command file.

**bin/mmtc 2019-183T02:14:00.0 2019-183T04:16:00.0 --clkchgrate-assign 1.00000000000**

The above example queries all AMPCS sessions for 2 hours and 2 minutes of telemetry beginning on July 2, 2019 (doy 183) at 02:14 UTC. It will assign the clock change rate of the new time correlation to **1.00000000000**. The particular value in this example indicates no drift and is equivalent to providing **--clkchgrate-nodrift** with no value.

**NOTE**: If you get an error message suggesting that SPICE kernels are missing, it could be due to not all required kernels having been provided, but it could also be due to the spacecraft ephemeris SPK kernel not covering the start and stop times provided to the MMTC. If this is the case, the MMTC cannot compute the essential one-way light travel time (OWLT) and will fail.

| Option | Full Option Name | Value(s) | Description |
| --- | --- | --- | --- |
| -c | --generate-cmd-file | N/A | Generate an Uplink Command File. Overrides the corresponding configuration parameter. This file will be generated only if this option is provided. |
|  | --clkchgrate-assign | A floating-point number (e.g., 1.00000001234). | In a testing environment and in certain operational scenarios, it can be necessary to have the MMTC assign the clock change rate of the new time correlation to a value specified by the user rather than computing it. Also, if this option is selected, the MMTC will not replace the existing clock change rate in the SCLK kernel with an interpreted value. This value will appear in the latest time correlation record in the SCLK kernel and SCLK/SCET file. This option and **–clkchgrate-compute** are mutually exclusive. |
|  | --clkchgrate-compute | **I** or **p** | **I** : Directs the MMTC to compute a predicted clock drift rate AND overwrite the previous time correlation record with an interpolated clock change rate. This is the default MMTC behavior.  **P** : Directs the MMTC to compute a predicted clock drift rate, but does NOT overwrite the previous time correlation record.  This option and **–clkchgrate-assign** are mutually exclusive. |
|  | --clkchgrate-nodrift | N/A | In a testing environment and in certain operational scenarios, it can be necessary to have the MMTC assign the clock change rate of the new time correlation to 1.0, indicating no clock drift, rather than computing it. Also, if this option is selected, the MMTC will not replace the existing clock change rate in the SCLK kernel with an interpreted value. The value 1.00000000000 will appear in the latest time correlation record in the SCLK kernel and SCLK/SCET file. This option and **–clkchgrate-compute** or **–clkchgrate-assign** are mutually exclusive. |
| -F | --disable-contact-filter | N/A | Turns off the Contact Filter. Overrides the Contact Filter enabled feature in the configuration parameters (filter.contact.enabled). If the Contact Filter is not enabled in the configuration parameters, this option does nothing. It may be desirable to turn off the Contact Filter for a single run in test environments and in some operational scenarios such as when there is a discontinuity of telemetry data, a switch between onboard oscillators, or a new clock partition is declared. |
| -T | --test-mode-owlt | A floating point value of units seconds (e.g., 12.413) | This option is for use ONLY in I&T environments. It directs the MMTC to use the provided value for the One-way Light Time (OWLT) rather than compute it from ephemeris. It also allows processing to continue if there is insufficient previous data in the input SCLK Kernel to compute the Predicted CLKRATE. This option is strictly for testing and should NEVER be used operationally. |

| -K | --ampcs-session-id | Positive integer | The AMPCS session ID to query packet and frame data from. This option is only applicable when using the AMPCS telemetry source plugin. |
| --- | --- | --- | --- |
| -n | --connection-parms | Free-format String | Any values, such as AMPCS database connection parameters, that need to be specified in AMPCS queries. This is unparsed and added to the chill\_get\_\* command calls. This option is only applicable when using the AMPCS telemetry source plugin. |
| -h | --help | N/A | Prints out a short summary of the command line usage and these options. If this option is specified, the MMTC will not perform time correlation. |

**Table 4 The Optional MMTC Command Line Options**

# Inputs

Though SPICE SCLK kernel files are one of the primary output products of the MMTC (see section 10 for details about outputs), MMTC also requires a seed SPICE SCLK kernel file or other previously generated SPICE SCLK kernel file as an input in order to run. It requires other inputs and writes to some other files as well. Upon start-up, the MMTC loads a set of configuration parameters read from a configuration file. It then loads a set of SPICE kernels, which are specified in the configuration parameters. In the course of its processing, it reads and writes to a cumulative Summary Table, and writes to a cumulative Raw Telemetry Table and a cumulative Time History File.

## Inputs

### Configuration Parameters

Upon start-up, the MMTC reads an XML-formatted file containing a large set of configuration parameters (**$TK\_CONFIG\_PATH/TimeCorrelationConfigProperties.xml**). These parameters tell the application where its inputs and outputs are, and direct most of its processing. The configuration parameters are described in detail in section 8. They must be set for each mission and each instance of the MMTC.

The MMTC also reads a file that configures message logging (see section 12). If using the AMPCS telemetry plugin, the MMTC also reads a file that describes the format of the Time Correlation (TK) packet (see section 8.3).

### SPICE Kernels

The MMTC uses the NAIF SPICE library to do many of its internal computations. SPICE “kernels” are actually specially-formatted parameter and data files that an application loads into SPICE. The kernels that one must load depend on the calculations to be performed. The MMTC uses SPICE to convert between types of time measurement, to compute one-way light travel time (OWLT), distances between the spacecraft and Earth, distances between the spacecraft and the Sun, the Earth-Sun distance, the velocity of the spacecraft relative to the Earth, the velocity of the spacecraft relative to the Solar System barycenter, and the velocity of the Earth relative to the Solar System barycenter. The SPICE kernels needed are discussed in detail in section 8.2. In particular, see section 8.2.4 for a discussion of the input SCLK kernels and the incrementing of versions.

### Telemetry Archive

The Telemetry Archive is the data store maintained by the core commanding and telemetry system for the mission. It contains a store of all telemetry frames and packets received from the spacecraft. The TK packets that are the primary input to the MMTC reside in the archive. The MMTC queries data from the telemetry archive by ERT in order to obtain time correlation information. The AMPCS telemetry archive is an example. A Raw Telemetry Table generated from a previous run of the MMTC can also be used as a telemetry source if there is a desire to reprocess previously computed time correlations. Telemetry source configuration is explained in detail in section 7.

## Environment Variables

MMTC may use the following environment variables defined in the host environment.

|  |  |  |
| --- | --- | --- |
| **Variable** | **Required?** | **Description** |
| **JAVA\_HOME** | YES | This must be set to the path of the JRE. It is required in order to launch MMTC. |
| **CHILL\_GDS** | CONDITIONAL | The standard AMPCS environment variable that gives the location of the AMPCS *chill*\* interface programs. Required if and only if using an AMPCS telemetry source. |
| **MMTC\_HOME** | NO | This should be set to the path where MMTC is installed. If it is not set, it defaults to the base installation path of the currently-executing MMTC program. |
| **TK\_CONFIG\_PATH** | NO | This should be set to the directory path containing the **TimeCorrelationConfigProperties.xml** configuration file. If it is not set, it defaults to $MMTC\_HOME/conf. |

Table 5 MMTC Environment Variables Overview of the System



Figure 3 High-Level View of the MMTC

Ground knowledge of the correlation between the spacecraft clock, which we call SCLK, and terrestrial time is determined using the timing of downlink telemetry frames. In particular, for each run of the MMTC module, a single “target” frame is identified as the vehicle for providing the information needed for that determination. Figure 3 provides a high-level view of the application with a mission adaptation for Europa Clipper as an example. Figure 4 shows this processing in terms of data flow.

The SCLK for the target frame may be included in the target frame itself or may be included in a later frame, which we refer to as the “supplemental” frame. If included in the target frame, the SCLK is typically packaged in a telemetry packet that is included in the data portion of that frame and we usually refer to that as a Time Correlation Packet or “TK” Packet.

If the SCLK for the target frame is included in a later frame, nominally the next frame downlinked, the SCLK may be packaged in a Time Correlation Packet or in the downlink frame secondary header itself. Which option is chosen depends on the format of the downlink frame.

Each time MMTC is run, a collection of output products is updated. MMTC is run over a specified time interval. For some missions, MMTC will be run once a day over all the data accumulated for the previous day. For other missions, MMTC will run following each downlink contact between the spacecraft and a ground station to process all the data obtained during that contact. MMTC processes the newest (most recent) data first. If that data set does not provide a valid correlation between the spacecraft time and terrestrial time, the software notes that in a log file and looks at an earlier set of data within the specified interval.

That process continues until a “good” set of data is identified by a set of filters. That good set is used to determine the correlation between SCLK and Earth time. All the data used in that determination are assembled into a text record that is appended to the Summary Table. Each successful run of MMTC results is captured in a single new time record in the Summary Table. The MMTC completes when a good set of data is identified and all the products are produced or updated.

The receiving ground station reports the “Earth Received Time” or ERT of the received downlink frame, typically in terms of UTC. MMTC combines the SCLK of the supplemental frame with the ERT of the target frame and other data, such as the estimated one-way light time (OWLT) for the frame to travel from the spacecraft to the ground station, to establish the correlation between SCLK and a standard terrestrial time scale. The standard terrestrial time scale used by MMTC is called Terrestrial Dynamical Time or TDT. See Appendix A for a description of TDT. Figure 4 illustrates the data flows involved in time correlation.



Figure 4 Overview of MMTC processing flow.

# Telemetry Sources

The MMTC does not run in “real-time.” It interfaces with received telemetry repositories. It accepts telemetry from a Raw Telemetry Table that it itself created or that is of the same format that it creates, an AMPCS telemetry archive using the AMPCS telemetry source plugin that it ships with, or another telemetry source using another telemetry source plugin. (Taking input from Raw Telemetry Tables is generally not intended for operational use; it is intended to be used in I&T environments or when there is a desire to reprocess time correlations from previous runs.) The MMTC is invoked the same way regardless of the telemetry source. The configuration parameters **telemetry.source.name**, **telemetry.source.pluginDirectory**, and **telemetry.source.pluginJarPrefix** determine which telemetry source is used: if **telemetry.source.name** is set to rawTlmTable, then the MMTC expects a Raw Telemetry Table; otherwise, it looks for a plugin jar file with the basename specified by **telemetry.source.pluginJarPrefix** inside the plugin directory specified by **telemetry.source.pluginDirectory**, and then it looks for the class specified by **telemetry.source.name** within the jar file.

To run the MMTC with an AMPCS telemetry source, use the AMPCS plugin by setting **telemetry.source.pluginJarPrefix** to mmtc-plugin-ampcs-1.3.0 and **telemetry.source.name** to either AmpcsTlmArchive or AmpcsTlmWithFrames, as described in section 7.1.

Plugins that don’t ship with MMTC and are obtained from other sources should include instructions on what value to use for **telemetry.source.name**.

## AMPCS

A functioning AMPCS must be installed on the host system that the MMTC runs on and its telemetry archive must contain TK packets that can be queried by ERT. It must also contain the ancillary data described in section 4.3 that can be queried by ERT and Channel IDs. The AMPCS *chill*\* interface must be available, and the AMPC standard environment variable **$CHILL\_GDS** must be defined in the environment. The AMPCS plugin calls *chill*\* commands by spawning system calls and capturing the output.

### How It Works

When querying telemetry from the AMPCS, the MMTC uses the **chill\_get\_packets** and **chill\_get\_chanvals** interfaces. To get the time correlation data, it evokes

chill\_get\_packets -m -K <*sessionId*> --packetApid <*TKPKTAPID*> --timeType ERT –beginTime <*begin time*> --endTime <*end time*> --report –filename <*pktFilespec*>

This command queries AMPCS for the time correlation packets in the *begin time* to *end time* ERT interval, as specified by *TKPKTAPID* which is the APID of the time correlation packet read from configuration parameters. *sessionID* is the AMPCS session ID number. A file containing the binary packets is written to the file /*tmp*/*pktFilespec* in the system temporary directory. *pktFilespec* is a contrived unique filename with the file suffix **.tkpkt**. The MMTC then reads this file and decommutates the packets inside extracting the time correlation information. Once it has done this, the packet file is no longer needed and will be deleted by the operating system whenever it routinely clears out its /tmp directory. The MMTC also captures the metadata associated with each of the packets and extracts from it some essential information, such as ERT.

Some of the data that goes into the TimeHistoryFile is read from channels that are specified in the configuration parameters. These include the oscillator temperature and the time correlations computed onboard the spacecraft (TDT(S), onboard SCLK, and onboard clock change rate). The channel IDs of all of these parameters are specified in configuration parameters (TimeCorrelationConfigProperties.xml).

chill\_get\_chanvals -m -K sessionId –timeType ERT –-channelIds *GNCSCLK,TDTS,SCLK1,TDT1,CLKGCHRATE1* –beginTime <*begin time*> --endTime <*end time*>

The APIDs used in the queries above are obtained from the following configuration parameters:

*TKPKTAPID telemetry.source.plugin.ampcs.tkpacket.apid*

*GNCSCLK telemetry.source.plugin.ampcs.channelId.gncsclk*

*TDTS telemetry.source.plugin.ampcs.channelId.tdts*

*SCLK1 telemetry.source.plugin.ampcs.channelId.sclk1*

*TDT1 telemetry.source.plugin.ampcs.channelId.tdt1*

*CLKGCHRATE1 telemetry.source.plugin.ampcs.channelId.tdtChgRate*

The oscillator temperature value or values, if there are two (only one is active at a time), are queried from AMPCS telemetry with APIDs specified in the following configuration parameter (this parameter can be a list of APIDs if there are multiple sources for the temperature value):

TKOSCTEMP telemetry.source.plugin.ampcs.channelId.oscillatorTemperature

The AMPCS telemetry query has two modes, as indicated in the **telemetry.source.name** configuration parameter.

AmpcsTlmArchive

If **telemetry.source.name** is set to **AmpcsTlmArchive**, the MMTC will assume that a set of consecutive TK packets are available in the query interval from which it can construct a sample set. It will then select one of these as the source of the time correlation data. It will assume that one of the preceding TK packets, usually the immediate predecessor as indicated by VCFC, comes from the target frame and will use its ERT for computations. Again, there must be a set of TK packets delivered in consecutive frames because it will look back a fixed number of packets to find the one with the associated ERT. By “consecutive,” we mean that their frame VCFCs, as provided in the AMPCS metadata, are exactly 1 greater than the preceding TK packet. The target TK packet needs to always precede the one selected from the sample by a fixed number. This is usually 1, but can be any fixed number as specified in the **telemetry.supplementalSampleOffset** configuration parameter.

AmpcsTlmWithFrames

If set to **AmpcsTlmWithFrames**, the MMTC will not try and associate the SCLK values given in the selected TK packet with a frame by looking back a fixed number of packets. Instead, it will query for it using *chill\_get\_frames* and the VCID and VCFC provided in the data portion of the selected TK packet. This way, it will not need packets from consecutive frames in order to process. Note that, as a result, the ERT, SCLK, Consecutive Frames, and VCID filters are not applicable and cannot be used. Always set the **telemetry.samplesPerSet** configuration parameter to **1** and the **filter.ert.enabled**, **filter.sclk.enabled**, **filter.consecutiveFrames.enabled**, and **filter.vcid.enabled** parameters to **false** when running in this mode. When running in this mode, the time correlation packets MUST contain a downlink data rate. The downlink data rate cannot be computed if frames or packets are non-consecutive.

## Raw Telemetry Table

The MMTC creates and writes to a cumulative output file called the Raw Telemetry Table. The Raw Telemetry Table stores a record of the raw time correlation data that were used to perform all previous time correlations. MMTC appends to it each time it runs. This file can also serve as an input, thus allowing time correlation computations to be rerun. This is useful in I&T environments where TK packets are not available or when there is a need to reprocess time correlations. This need can arise due to errors having occurred in the previous runs or when an improved spacecraft ephemeris is available that will provide for more accuracy.

When the **telemetry.source.name** configuration parameter is set to **rawTlmTable**, the MMTC will read its input from the Raw Telemetry Table indicated in the **telemetry.source.plugin.rawTlmTable.tableFile.uri** configuration parameter. The fields in the Time History File that come from telemetry will be null.

# Configuration Files

## Time Correlation Configuration File

The MMTC Configuration file is an XML-format file that conforms to the schema **TimeCorrelationConfigProperties**.**xsd** provided with the distribution. The full path to this file is **$TK\_CONFIG\_PATH/TimeCorrelationConfigProperties.xml**. Table 6 describes the MMTC configuration parameters.

This file is a Java XML properties file and contains a set of key/value pairs. Some of these parameters in this file are always required and others only in certain configurations. Note that not all of the parameters are applicable to all missions.

Telemetry source plugins may also require additional configuration parameters, and these parameters should be included in **$TK\_CONFIG\_PATH/TimeCorrelationConfigProperties.xml**. By convention, plugin-specific parameters have names that begin with “telemetry.source.plugin.<plugin-identifier>.”. Table 7 describes the configuration parameters that are specific to the AMPCS telemetry source plugin that ships with the MMTC. If a different telemetry source is used, the provider of the telemetry source plugin should include instructions about relevant configuration parameters.

| Parameter Name | Required? | Type | Description |
| --- | --- | --- | --- |
| missionName | REQUIRED | STR | The name of the mission, e.g., “Europa Clipper”. |
| missionId | REQUIRED | INT | The Mission ID to be included in SCLK/SCET files. |
| spacecraftName | REQUIRED | STR | The name of the spacecraft, e.g., “Europa Clipper”. |
| spacecraft.id | REQUIRED | INT | The DSN-assigned Spacecraft ID (SCID). (Not to be confused with the CCSDS/SANA ID). |
| compute.clkchgrate.mode | OPTIONAL | STR | Specifies the default method by which the MMTC will compute the clock change rate. Options are: **compute-interpolate**, **compute-predict**, **assign**, or **nodrift**. See section 2.1 for a description of these options. This parameter can be overridden from the command line (see Table 4). This parameter is optional; if it is not specified, the MMTC will default to **compute-interpolate** unless overridden from the command line. |
| compute.clkchgrate.assignedValue | CONDITIONAL | DBL | Specifies the fixed value to use as the clock change rate. This parameter is only applicable if compute.clkchgrate.mode is set to “assign”, and it can be overridden from the command line. |
| spacecraft.timeDelaySec | REQUIRED | DBL | TDSC: Time Delay-Spacecraft. The number of seconds (usually a fraction of a second) delay in the spacecraft systems that affects a frame radiation SCLK. This is the time delay from the SCLK reported by the spacecraft to when the related downlink frame leaves the spacecraft antenna. With careful spacecraft design, this can be ~ 0 at all downlink bit rates. TDSC is useful during Mission Simulations prior to launch to bias TDT(G) to a future time, but that is an “off-label” application of this parameter that is not done in flight. |
| spacecraft.frameErtBitOffsetError | REQUIRED | DBL | Frame offset from ERT stamp. An adjustment to be used in the bit-rate dependent time error due to a mismatch between frame times reported by the spacecraft and the receiving ground station; TDBE = 0 when there is no mismatch. This value will often be 0. This value divided by the measured downlink data rate in bps will be subtracted from the computation of ground time (TDT(G)). |
| spacecraft.sclkModulusOverride | OPTIONAL | INT | Overrides the modulus of the SCLK fine time (i.e., subseconds) given in the second field of the SCLK Kernel variable SCLK01\_MODULI\_\*. If this parameter is not provided, the MMTC uses the value in the SCLK Kernel to convert SCLK fine time to decimal fraction of a second. If provided, it uses the value in this parameter. This is necessary for missions where the SCLK in the TK packets or frame header is coming from a system other than the GNC or avionics (e.g., a radio) that has a different fine time modulus. |
| spacecraft.activeRadioId | OPTIONAL | STR | A single character that indicates the active Radio. This parameter is optional and applies only to missions that use a radio clock to set the SCLK values for time correlation and may be omitted for those that do not. If not provided, the Radio ID will either be set per mission-specific adaptation or default to “-“. If the active radio changes, it is incumbent upon a user to change this parameter accordingly. |
| spacecraft.activeOscillator | OPTIONAL | STR | Specifies the active oscillator that is providing the SCLK values if it is not to be determined at runtime. This value is typically ‘1’ or ‘2’ or ‘A’ or ‘B’. This parameter is optional and mutually exclusive with parameters spacecraft.oscillator1Vcid and spacecraft.oscillator2Vcid. If these two other parameters are also provided, they are ignored and the value in spacecraft.activeOscillator is used. |
| spacecraft.oscillator1Vcid | OPTIONAL | STR | [To be implemented] Specifies the VCID of downlink time correlation telemetry associated with oscillator 1 or oscillator A. Only applicable for missions in which the active oscillator can be determined from VCID. If this parameter is provided, spacecraft.oscillator2Vcid must be as well. It is mutually exclusive with spacecraft.activeOscillator. |
| spacecraft.oscillator2Vcid | OPTIONAL | STR | [To be implemented] Specifies the VCID of downlink time correlation telemetry associated with oscillator 2 or oscillator B. Only applicable for missions in which the active oscillator can be determined from VCID. If this parameter is provided, spacecraft.oscillator1Vcid must be as well. It is mutually exclusive with spacecraft.activeOscillator. |
| groundStationMap.path | REQUIRED | STR | The fully qualified path to the Ground Stations Map file. |
| sclkPartitionMap.path | REQUIRED | STR | The fully qualified path to the SCLK Partition Map file. |
| telemetry.samplesPerSet | REQUIRED | INT | The number of consecutive samples (e.g., frames or TK packets) needed to perform time correlation.  Set this value to 1 if using the AMPCS telemetry source plugin’s AmpcsTlmWithFrames mode. |
| telemetry.supplementalSampleOffset | REQUIRED | INT | The number of frames that separate the target and supplemental frame (i.e., the frame containing the SCLK value). This will be 1 if the supplemental frame immediately follows the target frame based on VCFC.  Set this value to 1 if using the AMPCS telemetry source plugin’s AmpcsTlmWithFrames mode. |
| telemetry.frameSizeBits | REQUIRED | INT | The size of a mission telemetry frame containing time correlation data in bits. This is the entire size of the transfer frame, including headers. |
| telemetry.source.name | REQUIRED | STR | Indicates which telemetry source to use. A value of “rawTlmTable” causes the MMTC to use the Raw Telemetry Table specified by telemetry.source.plugin. rawTlmTable.tableFile.uri; this would generally only be useful in test venues or if a previous run needs to be reprocessed. Otherwise, valid values depend on the telemetry source plugin specified by telemetry.source.pluginDirectory and telemetry. source.pluginJarPrefix; consult the plugin provider for instructions.  When using the AMPCS plugin that ships with MMTC, valid values for telemetry.source.name are “AmpcsTlmArchive” and “AmpcsTlmWithFrames”:   * **AmpcsTlmArchive**: Query all information from TK packets from the AMPCS TLM archive. Use if consecutive TK packets are available to build a sample set. * **AmpcsTlmWithFrames**: Query supplemental sample information from a TK packet. Query the target sample information from the frame referenced from within the TK packet. Use if consecutive TK packets are not available. NOTE: In this mode, the TK packets must contain the downlink data rate in bps. |
| telemetry.source.plugin.rawTlmTable. tableFile.uri | CONDITIONAL | STR | The fully qualified path to the input Raw Telemetry Table file. This parameter is ignored if telemetry. source.name is not set to rawTlmTable. |
| telemetry.source.pluginDirectory | CONDITIONAL | STR | The path to the directory containing the telemetry source plugin that should be used. This parameter is ignored if telemetry.source.name is rawTlmTable. |
| telemetry.source.pluginJarPrefix | CONDITIONAL | STR | The basename of the telemetry source plugin jar file that should be used. This parameter is ignored if telemetry.source.name is rawTlmTable. |
| telemetry.tkSampleWindow | RESERVED | INT | The number of seconds +/- at which an ERT query is to be made for a TK packet. This value will be subtracted from a begin time and added to an end time. This value is reserved for future use. Always set to 0. |
| telemetry.tkOscTempWindow | REQUIRED | INT | The number of seconds +/- at which the oscillator temperature is to be queried by ERT. This allows for the ERT of the queried channel to be approximate. This value will be subtracted from a begin time and added to an end time. |
| telemetry.tkParmWindow | REQUIRED | INT | The number of seconds +/- at which the GNC packet containing onboard SCLK/TDT(S) values is to be queried by ERT. This allows for the ERT of the queried channel to be approximate. This value will be subtracted from a begin time and added to an end time. |
| compute.tdtG.rate.predicted.lookBackDays | REQUIRED | FLT | The number of days to look back into previous contacts to get SCLK and TDT values when computing the predicted clock change rate. This value is a float type and will be converted to truncated integer whole hours internally. If there is no time correlation record that corresponds to this time between this time and the maxLookBackDays parameter below, it is an error. |
| compute.tdtG.rate.predicted.maxLookBackDays | REQUIRED | FLT | The maximum number of days to look back into previous contacts to get SCLK and TDT values when computing the predicted clock change rate. This value is a float type and will be converted to truncated integer whole hours internally. If there is no time correlation record in the input SCLK Kernel within this amount of time prior to the current sample time, the MMTC will terminate with an error, unless it is running in test mode. If in test mode, it will log a warning and continue. |
| compute.tdtS.threshold.errorMsec | REQUIRED | DBL | The threshold in milliseconds above which a computed error in TDT(S) calculations will indicate a warning as recorded in the Time History File. |
| compute.tdtS.threshold.alarmMsec | REQUIRED | DBL | The threshold in milliseconds above which a computed error in TDT(S) calculations will indicate an alarm as recorded in the Time History File. |
| **The following parameters relate to SPICE kernels** | | | |
| spice.naifSpacecraftId | ALL | INT | The spacecraft ID assigned by NAIF. Normally, this is the negative of the DSN SCID. |
| spice.kernel.sclk.kerneldir | REQUIRED | STR | The directory in which the SCLK kernels reside. The MMTC will search this directory for the SCLK kernel with the highest version number. Since this kernel will be input to the next run, this directory is also the directory to which the new SCLK Kernel will be written. |
| spice.kernel.sclk.inputPathOverride | Testing Only | STR | The fully qualified path of an SCLK kernel that the MMTC is to use as the input SCLK. Overrides the default behavior of searching the sclk.kerneldir for the latest SCLK kernel. This is intended for use in testing only and should not be used operationally. This can cause failures in succeeding runs due to duplicate filenames. |
| spice.kernel.sclk.baseName | REQUIRED | STR | The basename of the SCLK kernel. For example, given “europaclipper\_00000.tsc”, “europaclipper” is the basename. |
| spice.kernel.sclk.separator | REQUIRED | STR | The separator between the basename and the version number of the SCLK kernel. For example, given “europaclipper\_00000.tsc”, “\_” (underscore) is the separator. |
| spice.kernel.mk.path | OPTIONAL | STR | The fully qualified path to the metakernel to load. This parameter is optional. |
| spice.kernel.lsk.path | OPTIONAL | STR | The fully qualified path to the leap seconds kernel (LSK) to load. This parameter is optional and should not be provided if the LSK is included in a metakernel. |
| spice.kernel.spk.path | OPTIONAL | STR[] | The fully qualified path(s) to the ephemeris kernels (SPK) to load. This parameter is optional and should only be provided for SPKs that are not included in a metakernel.  Multiple values are separated by commas. |
| spice.kernel.fk.path | REQUIRED | STR[] | The fully qualified path(s) to the frame kernels (FK) to load. This parameter is optional and should only be provided for FKs that are not included in a metakernel.  Multiple values are separated by commas. |
| spice.kernel.pck.path | REQUIRED | STR[] | The fully qualified path(s) to the planetary constants kernels (PCK) to load. This parameter is optional and should only be provided for PCKs that are not included in a metakernel.  Multiple values are separated by commas. |
| **The following parameters relate to data sample filters** | | | |
| filter.contact.enabled | REQUIRED | BOOL | Set this value to true to turn the contact filter on, or false to turn it off.  Overridden by the -F command line option, if present. |
| filter.contact.deltaLowerThreshold | CONDITIONAL | DBL | The lower threshold in milliseconds per day above which a data sample will fail the Contact Filter. This parameter is ignored if filter.contact.enabled is false. |
| filter.contact.deltaUpperThreshold | CONDITIONAL | DBL | The upper threshold in milliseconds per day below which a data sample will fail the Contact Filter. This parameter is ignored if filter.contact.enabled is false. |
| filter.ert.enabled | REQUIRED | BOOL | Set this value to true to turn the ERT filter on, or false to turn it off.  Always set this value to false if using the AMPCS telemetry source plugin’s AmpcsTlmWithFrames mode. |
| filter.ert.deltaThresholdSec | CONDITIONAL | DBL | The threshold in seconds above which the ERTs separating two consecutive time correlation data samples will fail the ERT filter. This parameter is ignored if filter.ert.enabled is false. |
| filter.sclk.enabled | REQUIRED | BOOL | Set this value to true to turn the SCLK filter on, or false to turn it off.  Always set this value to false if using the AMPCS telemetry source plugin’s AmpcsTlmWithFrames mode. |
| filter.sclk.deltaThresholdSec | CONDITIONAL | DBL | The threshold in seconds above which the SCLKs separating two consecutive time correlation data samples will fail the SCLK filter. This parameter is ignored if filter.sclk.enabled is false. |
| filter.groundStation.enabled | REQUIRED | BOOL | Set this value to true to turn the stations filter on, or false to turn it off. |
| filter.groundStation.pathIds | CONDITIONAL | INT | A comma-separated list of ground station path IDs that the MMTC can receive data from. This parameter is ignored if filter.groundStation.enabled is false. |
| filter.minDataRate.enabled | REQUIRED | BOOL | Set this value to true to turn the minimum data rate filter on, or false to turn it off. |
| filter.dataRate.minDataRateBps | CONDITIONAL | INT | The downlink data rate in bits per second below which time correlation sample data will be rejected. This parameter is ignored if filter.minDataRate.enabled is false. |
| filter.maxDataRate.enabled | REQUIRED | BOOL | Set this value to true to turn the maximum data rate filter on, or false to turn it off. |
| filter.dataRate.maxDataRateBps | CONDITIONAL | INT | The downlink data rate in bits per second above which time correlation sample data will be rejected. This parameter is ignored if filter.maxDataRate. enabled is false. |
| filter.validFlag.enabled | REQUIRED | BOOL | Set this value to true to turn the valid sample filter on, or false to turn it off. |
| filter.consecutiveFrames.enabled | REQUIRED | BOOL | Set this value to true to turn the consecutive frames filter on, or false to turn it off.  Always set this value to false if using the AMPCS telemetry source plugin’s AmpcsTlmWithFrames mode. |
| filter.vcid.enabled | REQUIRED | BOOL | Set this value to true to turn the VCID filter on, or false to turn it off.  Always set this value to false if using the AMPCS telemetry source plugin’s AmpcsTlmWithFrames mode. |
| filter.vcid.validVcids | CONDITIONAL | INT[] | Lists the comma-separated VCIDs that indicate which Virtual Channels may supply data for time correlation. This only affects the time correlation TK packets or frames. It does not affect the ancillary information (e.g., oscillator temperature, TDT(S), etc.) This parameter is ignored if filter.vcid.enabled is false. |
| **The following parameters relate to the location and contents of the output products** | | | |
| table.rawTelemetryTable.uri | REQUIRED | STR | The desired URI of the output Raw Telemetry Table file. This file and the Summary Table are commonly co-located. |
| table.rawTelemetryTable.dateTimePattern | REQUIRED | STR | The pattern that is used to parse ERTs from query metadata and to write ERT to the RawTlmTable in calendar string date/time form (e.g., ISO DOY format: “yyyy-DDD’T’HH:mm:ss.SSSSSS”). |
| table.summaryTable.uri | REQUIRED | STR | The desired URI of the output Summary Table file. This file and the Raw Telemetry Table are commonly co-located. |
| table.timeHistoryFile.uri | REQUIRED | STR | The desired URI of the output Time History File. |
| table.timeHistoryFile.excludeColumns | OPTIONAL | STR[] | A comma-separated list of the names of columns that should be excluded from the Time History File. Valid values consist of any subset of the strings listed below. This parameter is optional; if not specified, all columns will be included.   * Enc\_SCLK * Int\_SCLK * TDT(G) * TDT(G)\_Calendar * TDB(G) * ClkChgRate(s/s) * SCLK\_Drift(ms/day) * Oscillator * Osc\_Temp(degC) * ClkChgRateMode * ChrRateInterv(days) * Ep(ms) * |Ep/dt|(ms/day) * Interval(days) * SCLK\_Partition * RF\_Encoding * GroundStationID * OWLT(sec) * SpacecraftSunDist(AU) * SpacecraftSunDist(km) * SpacecraftEarthDist(km) * EarthSunDist(AU) * EarthSunDist(km) * SpacecraftVelocity(SSB\_km/s) * EarthVelocity(SSB\_km/s) * SpacecraftVelocity(Earth\_km/s) * The Active Radio * DataRate(bps) * SCET * TDT(S) * TDT(S)\_Calendar * TDT(S)\_Error(ms) * SCLK\_for\_TDT(S) * TDT(S)ErrorThreshold(ms) * TDT(S)AlarmThreshold(ms) * SCLK1 * TDT1 * TDT1\_Calendar * ClockChangeRateForTDT(S) * Warning * Alarm |
| product.sclkScetFile.create | REQUIRED | BOOL | Activates creation of SCLK/SCET files. Set to true if a SCLK/SCET file is to be created. Valid values are “true” or “false”. |
| product.sclkScetFile.dir | REQUIRED | STR | The directory to which the SCLK/SCET files are written. |
| product.sclkScetFile.baseName | REQUIRED | STR | The basename of the SCLK/SCET file. For example, given “europaclipper\_00000.coeff”, “europaclipper” is the basename. Applicable only if product.sclkScetFile.create=true. |
| product.sclkScetFile.separator | REQUIRED | STR | The separator between the basename and the version number of the SCLK/SCET file. For example, given “europaclipper\_00000.coeff”, “\_” (underscore) is the separator. Applicable only if product.sclkScetFile.create=true. |
| product.sclkScetFile.suffix | REQUIRED | STR | The file suffix of the SCLK/SCET file. For example, given “europaclipper\_00000.coeff”, “coeff” is the suffix. Applicable only if product.sclkScetFile.create=true. |
| product.sclkScetFile.utcPrecision | REQUIRED | INT | The number of digits to which the SCET fraction of second is to be written in an SCLK/SCET file. Applicable only if product.sclkScetFile.create=true. |
| product.sclkScetFile.datasetId | REQUIRED | STR | The value for the DATA\_SET\_ID field in the header of the SCLK/SCET file. Applicable only if product.sclkScetFile.create=true. |
| product.sclkScetFile.producerId | REQUIRED | STR | The value for the PRODUCER\_ID field in the header of the SCLK/SCET file. Applicable only if product.sclkScetFile.create=true. |
| product.uplinkCmdFile.create | REQUIRED | STR | Activates creation of Uplink Command Files. Set to true if an Uplink Command File is to be created. Valid values are “true” or “false”. |
| product.uplinkCmdFile.outputDir | REQUIRED | STR | The path to the directory where Uplink Command Files are to be written. |
| product.uplinkCmdFile.baseName | REQUIRED | STR | The basename of the Uplink Command File. For example, given the filename “uplinkCmd1577130458.csv”, “uplinkCmd” is the basename. Applicable only if product.uplinkCmdFile.create=true. |

Table 6 Configuration Parameters ($TK\_CONFIG\_PATH/TimeCorrelationConfigProperties.xml)

| Parameter Name | Required? | Type | Description |
| --- | --- | --- | --- |
| telemetry.source.plugin.ampcs.chillTimeoutSec | REQUIRED | INT | The number of seconds to wait for each AMPCS *chill\_get\_*\* query to complete. If a query takes longer, MMTC will terminate and log an error message. |
| telemetry.source.plugin.ampcs.tkpacket.apid | REQUIRED | INT | The APID of the TK packet. |
| telemetry.source.plugin.ampcs.tkpacket. tkPacketSizeBytes | REQUIRED | INT | The size in bytes of the TK packet, excluding the primary packet header minus 1. In other words, the size of the data portion of the packet, which includes the secondary header, minus 1. |
| telemetry.source.plugin.ampcs.tkpacket. tkPacketFileOutputDir | REQUIRED | STR | The path to a directory where the binary packet file produced by *chill\_get\_packets* can be written/stored. |
| telemetry.source.plugin.ampcs.tkpacket. tkPacketDescriptionFile.uri | REQUIRED | STR | The path to the XML file that describes the TK packet. |
| telemetry.source.plugin.ampcs.frame.ertFieldName | CONDITIONAL | STR | The name of the CSV field output by *chill\_get\_frames* that contains the target frame ERT. Applicable only it telemetry.source.name is set to AmpcsTlmWithFrames. |
| telemetry.source.plugin.ampcs.frame. dssIdFieldName | CONDITIONAL | STR | The name of the CSV field output by *chill\_get\_frames* that contains the target frame ground station identifier. Applicable only it telemetry.source.name is set to AmpcsTlmWithFrames. |
| telemetry.source.plugin.ampcs.frame.vcfcFieldName | CONDITIONAL | STR | The name of the CSV field output by *chill\_get\_frames* that contains the target frame Virtual Channel Frame Count (VCFC). Applicable only it telemetry.source.name is set to AmpcsTlmWithFrames. |
| telemetry.source.plugin.ampcs.frame.vcidFieldName | CONDITIONAL | STR | The name of the CSV field output by *chill\_get\_frames* that contains the target frame Virtual Channel ID (VCID). Applicable only it telemetry.source.name is set to AmpcsTlmWithFrames. |
| telemetry.source.plugin.ampcs.frame. maxTkpacketFrameSeparation | CONDITIONAL | STR | When searching for the target frame corresponding to a timekeeping packet, the maximum number of seconds that a frame is allowed to precede the packet and still be considered a candidate. Applicable only it telemetry.source.name is set to AmpcsTlmWithFrames. |
| telemetry.source.plugin.ampcs.tkpacket. apidFieldName | REQUIRED | STR | The name of the CSV field output by *chill\_get\_packets* that contains the APID. This value is currently always set to “apid”. |
| telemetry.source.plugin.ampcs.tkpacket. scetFieldName | REQUIRED | STR | The name of the CSV field output by *chill\_get\_packets* that contains the SCET. This value is currently always set to “scet”. |
| telemetry.source.plugin.ampcs.tkpacket. ertFieldName | REQUIRED | STR | The name of the CSV field output by *chill\_get\_packets* that contains the ERT. This value is currently always set to “ert”. |
| telemetry.source.plugin.ampcs.tkpacket. sclkFieldName | REQUIRED | STR | The name of the CSV field output by *chill\_get\_packets* that contains the SCLK. This value is currently always set to “sclk”. |
| telemetry.source.plugin.ampcs.tkpacket. vcidFieldName | REQUIRED | STR | The name of the CSV field output by *chill\_get\_packets* that contains the VCID. This value is currently always set to “vcid”. |
| telemetry.source.plugin.ampcs.tkpacket. vcfcFieldName | REQUIRED | STR | The name of the CSV field output by *chill\_get\_packets* that contains the VCFC. This value is currently always set to “sourceVcfcs”. |
| telemetry.source.plugin.ampcs.tkpacket. dssIdFieldName | REQUIRED | STR | The name of the CSV field output by *chill\_get\_packets* that contains the DSS ID. This value is currently always set to “dssId”. |
| telemetry.source.plugin.ampcs.tkpacket. pktLengthFieldName | REQUIRED | STR | The name of the CSV field output by *chill\_get\_packets* that contains the packet length. |
| telemetry.source.plugin.ampcs.channelId. channelIdFieldName | REQUIRED | STR | The name of the field in the CSV metadata output from a chill\_get\* command that contains the identifier of the channel (Channel ID) that was returned from a query. |
| telemetry.source.plugin.ampcs.channelId. channelErtFieldName | REQUIRED | STR | The name of the field in the CSV metadata output from a chill\_get\* command that contains the Earth Receive Time (ERT) associated with a channel returned by a query. |
| telemetry.source.plugin.ampcs.channelId. dnFieldName | REQUIRED | STR | The name of the field in the CSV metadata output from a chill\_get\* command that contains the raw value provided by a channel. This is the content of the channel returned from a query and is provided as a digital number (DN) to be converted to the type specified in the telemetry dictionary. |
| telemetry.source.plugin.ampcs.channelId.sclk1 | REQUIRED | STR | The channel ID of the SCLK1 value. |
| telemetry.source.plugin.ampcs.channelId.tdt1 | REQUIRED | STR | The channel ID of the TDT1 value. |
| telemetry.source.plugin.ampcs.channelId.gncsclk | REQUIRED | STR | The channel ID of the SCLK used to compute TDT(S) onboard the spacecraft. |
| telemetry.source.plugin.ampcs.channelId.tdts | REQUIRED | STR | The channel ID of the TDT(S) computed onboard the spacecraft. |
| telemetry.source.plugin.ampcs.channelId. tdtChgRate | REQUIRED | STR | The channel ID of the clock change rate computed onboard the spacecraft and associated with TDT(S). |
| telemetry.source.plugin.ampcs.channelId. oscillatorTemperature | REQUIRED | STR[] | The channel IDs of the sensors that provide the temperature of oscillators on the spacecraft. If there are more than one, they are listed as a comma-separated string with the first assumed to correspond to Oscillator 1 or ‘A’. |

Table 7 AMPCS Plugin-Specific Configuration Parameters

## Input SPICE Kernels

MMTC uses the JPL NAIF “SPICE” toolkit, which requires a set of SPICE “kernels” to operate. These include generic NAIF SPICE kernels, such as planetary ephemerides, as well as mission spacecraft ephemeris kernels. The specific SPICE kernels that the MMTC needs will vary greatly depending on the mission and even the phase of the mission. Basically, the MMTC needs all of the kernels that SPICE needs in order to compute OWLT between the spacecraft and any receiving ground station, and that it needs to compute distance and velocity of the spacecraft relative to the Earth and Sun.

Consult the NAIF documentation (https://naif.jpl.nasa.gov/naif/tutorials.html) and the mission navigation team to determine which kernels are needed for a specific mission.

### Generic Kernels

Generic SPICE kernels are provided by the NASA Navigation and Ancillary Information Facility (NAIF) located at JPL. These are used on multiple missions and other astrometric applications. These are available from https://naif.jpl.nasa.gov/pub/naif/generic\_kernels.

Leap Seconds Kernel (LSK)

This kernel is specified in **spice.kernel.lsk.path** in the configuration parameters.

* naif00<*nn>*.tls

Frame Kernels (FK)

The FK kernels are provided in the configuration parameters as a list of comma-separated entries within the **spice.kernel.fk.path** configuration parameter.

* earth\_topo\_<*nnnnnn*>.tf
* dss\_<*?*>.tf – various station frame kernels depending on the mission

Planetary Constants Kernels (PCK)

The PCK kernels are provided in the configuration parameters as a list of comma-separated entries within the **spice.kernel.pck.path** configuration parameter.

* pck00<*nnn*>.tpc
* earth\_<*nnnnnn*>\_<*nnnnnn*>\_predict.bpc

Ephemeris Kernels (SPK)

The SPK kernels are provided in the configuration parameters as a list of comma-separated entries within the **spice.kernel.spk.path** configuration parameter.

* earthstns\_fx\_<*nnnnnn*>.bsp
* dss<*?*>.bsp – various station ephemeris kernels depending on the mission
* de<*nnn*>.bsp – a planetary ephemeris that may or may not be needed depending on the mission

### Mission Kernels

The mission kernels are specific to the given mission. The MMTC will require, at minimum, a spacecraft ephemeris. The spacecraft ephemeris kernels used by MMTC are the same short-term predicted spacecraft ephemeris kernels that are typically uploaded to the spacecraft to enable the spacecraft to know its location in space. The ephemeris kernels enable MMTC to compute the one-way light time (OWLT) for a downlink frame to travel from the spacecraft to the receiving ground station, and to compute distances and velocities for the Time History File. SPKs containing ephemerides of other bodies might be required as well, depending on the mission. The mission SPKs should be included in the list within the **spice.kernel.spk.path** configuration parameter. Mission-specific FK and PCK kernels should be included within the **spice.kernel.fk.path** and **spice.kernel.pck.path** configuration parameters respectively.

### Metakernel

Metakernels (MK) are a special type of SPICE kernel that contains a list of other SPICE kernels. Instead of listing the kernels individually in configuration parameters, as described above, one may simply list them in a metakernel specified in **spice.kernel.mk.path**. Only the SCLK kernel needs to be listed separately. One can include all or some of the other required kernels in the metakernel. If for any reason one does not wish to list all of the kernels in the metakernel (e.g., some ephemeris files get updated frequently), one can just list some of them in the metakernel and include the remainder in the individual configuration parameters described above. Using metakernels is convenient and highly recommended. Be aware that the order in which kernels are loaded into SPICE matters. If there is overlap between them, such as two SPKs with overlapping ephemeris for the same planetary body or spacecraft, the latter loaded kernel values will override those of the prior. As stated above, DO NOT include the SCLK kernel in the metakernel. See below.

### Input SCLK Kernel

The Input SCLK kernel is the SCLK kernel that the MMTC created the previous time that it ran. Each time the MMTC runs, it creates a new SCLK kernel that contains all of the contents of the previous one with a new time correlation record added to it. This new SCLK kernel is the input SCLK kernel for the next run. When the MMTC runs, it looks for SCLK kernels in the directory indicated in the **spice.kernel.sclk.kerneldir** configuration parameter that match the naming patterns given in the **spice.kernel.sclk.baseName**, **spice.kernel.sclk.separator**, and **spice.kernel.sclk.suffix** configuration parameters (<*basename*><*separator*><*nnnnn*>.<*suffix*>) where *nnnnn* is an integer version number. By SPICE conventions, SCLK kernel filenames should have a suffix of “.tsc”. The MMTC selects the matching kernel file with the highest version number. It assumes that this is the one from the previous run and loads it. The new kernel that it produces will have this name, but with the version number incremented.

Because the SCLK kernel is created with every run and the input SCLK kernel is different for every run, this kernel should not be included in the metakernel.

A manually created “seed” SCLK kernel with at least one time correlation record must exist the very first time that the MMTC runs. See the NAIF document SCLK – *Reference for the SPICE spacecraft clock subsystem and the Spacecraft Clock Kernel (SCLK)* (Table 2) for what this file contains and how it must be formatted. The MMTC currently requires that the ground time of the time correlation record be given in TDT time and as a calendar string. Numeric seconds of epoch TDT values, although allowed by SPICE, are not currently supported in the SCLK kernel. The time correlation record will typically define SCLK=0. The “seed” kernel will also contain standard information about the spacecraft clock and the first clock partition. See Appendix I for an example of a seed SCLK Kernel.

Since the SCLK kernel is both an input and an output of the MMTC, the input SCLK kernel must be in the same directory to which the new SCLK kernel will be written, which is the location indicated in the **spice.kernel.sclk.kerneldir** configuration parameter. The MMTC reads the latest SCLK kernel upon startup and creates a new one at the end of its run with one new time correlation record appended to the end and an incremented version number in the filename. The time correlation records in an SCLK kernel must always move forward in time. In other words, the ground time in each time correlation record must be later than that in the one before it. If one runs the MMTC with a start time that is earlier than the ground time in the last time correlation record in the SCLK kernel, it will terminate with an error (e.g., *ERROR: TDT(G) of the last record in SCLK kernel is later than the input data sample.*).

If running in a test environment where the same input data is being run over and over (e.g., the same start/stop times) it will be necessary to remove the SCLK kernels from previous test runs whose time correlation records are later than the start time of the test. For example, if the MMTC is run with the sole SCLK kernel *PSYC255SCLK\_00000.tsc* in the directory indicated in **spice.kernel.sclk.kerneldir**, it will produce a new SCLK kernel *PSYC255SCLK\_00001.tsc* and possibly more kernels depending on how many successive time intervals are run. If the MMTC is to be run again with the same start/stop time, it will be necessary to first remove *PSYC255SCLK\_00001.tsc* and any later SCLK kernels from the output directory. This situation should occur only in test venues, never operationally, unless data are being rerun to correct for an anomaly.

## TK Packet Description File

For AMPCS-based missions, the AMPCS telemetry source plugin reads the SCLK values and other related information from the time correlation (TK) packets received from the spacecraft and archived in the telemetry archive.

At minimum, the TK packet must contain a coarse SCLK value and a fine SCLK value. It might also contain the VCID and VCFC of the target frame, the target frame encoding method (e.g., turbo encoding), and the downlink data rate. The coarse and fine SCLK values are assumed to be in raw counts. These and all of the other values in the TK packet are assumed to be unsigned integers.

The user is reminded that the SCLK Coarse and SCLK Fine values are associated with the target frame, NOT the frame containing the TK packet. The target frame is usually, but not always, the frame that immediately precedes the supplemental frame (i.e., the one containing the TK packet) in the telemetry sequence. In other words, the VCFC of the target frame is a set number of counts (usually just 1) less than that of the supplemental frame. The number of frames that the supplemental frame follows the target frame is specified in the **telemetry.supplementalSampleOffset** configuration parameter.

Figure 5 gives the contents of a TK packet and the fields it might contain.

| **Field** | **Length (bits)** |
| --- | --- |
| CCSDS Packet Primary Header | 48 |
| CCSDS Packet Secondary Header | 48 |
| SCLK Coarse (seconds) | 32 |
| SCLK Fine (subseconds) | 16 |
| Spare | 15 |
| Target Frame SCLK/MET Missed Flag (validity flag) | 1 |
| Reserved | 2 |
| Target Frame VCID | 6 |
| Target Frame VCFC | 24 |
| Target Frame Encoding Mode | 8 |
| Target Frame Downlink Rate | 32 |

Figure 5 Structure and contents of a TK Packet (from the Europa Clipper mission).

The structure of TK packets is mission-specific. Each mission must describe its TK packets using a TK Packet Description XML file that implements the **tk\_packet.xsd** schema provided with the MMTC distribution. The full path to the TK Packet Description file is indicated in the **telemetry.source.plugin.ampcs.tkpacket.tkPacketDescriptionFile.uri** configuration parameter. Not all of the fields are applicable for all missions; however all missions must provide at minimum the **SCLK Coarse** and **SCLK Fine** fields in order to process. See Appendix H for an example of a TK Packet Description file.

The primary purpose of the definition file is to specify the locations and bit lengths of these fields within the packet so that the MMTC can decom them. Each field given in the definition file is identified by a name, the offset from the beginning of the packet (normally the first bit of the primary packet header), and the length of the field in bits. Figure 6 below provides an example of a TK Packet Description file that captures the TK packet definition given in Figure 5.

Figure 6 Sample TK packet definition file.

<?xml version="1.0"?>

<Packet\_Definition

xmlns="mmtc"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="mmtc tk\_packet.xsd">

<!-- Packet CCSDS Primary Header (not used) -->

<TelemetryPoint>

<name>"PrimaryPacketHeader"</name>

<offset>0</offset>

<length>48</length>

</TelemetryPoint>

<!-- Packet CCSDS Secondary Header (not used) -->

<TelemetryPoint>

<name>"SecondaryPacketHeader"</name>

<offset>48</offset>

<length>48</length>

</TelemetryPoint>

<!-- Bit 0 Timestamp (seconds), Coarse part of SCLK associated with the target frame -->

<TelemetryPoint>

<name>"SclkCoarse"</name>

<offset>96</offset>

<length>32</length>

</TelemetryPoint>

<!-- Bit 0 Timestamp (subseconds), Fine part of SCLK associated with the target frame -->

<TelemetryPoint>

<name>"SclkFine"</name>

<offset>128</offset>

<length>16</length>

</TelemetryPoint>

<!-- Last SCLK Missed Flag (Indicates invalid data if bit is set) -->

<TelemetryPoint>

<name>"ValidFlag"</name>

<offset>144</offset>

<length>1</length>

</TelemetryPoint>

<!-- Last Frame VCID (Virtual Channel ID of the target frame) -->

<TelemetryPoint>

<name>"TargetFrameVcid"</name>

<offset>162</offset>

<length>6</length>

</TelemetryPoint>

<!-- Last Frame VCFC (Virtual Channel Frame Count of the target frame) -->

<TelemetryPoint>

<name>"TargetFrameVcfc"</name>

<offset>168</offset>

<length>24</length>

</TelemetryPoint>

<!-- Last Frame Encoding Mode (Encoding method of the target frame) -->

<TelemetryPoint>

<name>"EncodingMethod"</name>

<offset>192</offset>

<length>8</length>

</TelemetryPoint>

<!-- Last Frame Downlink Data Rate in bits per second (Downlink rate of the target frame) -->

<TelemetryPoint>

<name>"DownlinkDataRate"</name>

<offset>200</offset>

<length>32</length>

</TelemetryPoint>

</Packet\_Definition>

This file must be customized for every mission based on the structure of its TK packets. All field offsets are given in bits from the beginning of the packet and all field lengths are in bits.

## SCLK Partition Map File

The SCLK Partition Map File indicates to the MMTC when SCLK partitions have occurred. These correspond to partitions provided in the SCLK kernels. Consult the SPICE documentation (https://naif.jpl.nasa.gov/pub/naif/toolkit\_docs/C/req/sclk.html) for an explanation of spacecraft clock partitions. This file is a simple CSV formatted text file that contains two items per line, a number and a date/time in UTC. The numbers begin at 1 and increment with each line for however many partitions are defined. These correspond to the **SCLK\_PARTITION\_START\_**<*scid*> fields in the SCLK Kernel. The sample SCLK Partition Map file given in Figure 7 contains three clock partitions. The first begins at the start of the epoch used for SCLK=0 for the given mission, in this case January 01, 2010 at 00:00h. There were two clock jumps or resets in July and November of 2010. This file is created and updated manually.

Clock partitions are mainly an artifact from the time when onboard spacecraft clock registers were small and would numerically roll over after a certain length of time producing an invalid SCLK count. With most modern spacecraft having larger clock registers, this is usually no longer an issue. Occasionally, there can be clock resets due to an anomaly requiring the definition of a new SCLK partition. The SPICE library handles SCLK clock partitions as defined in the SCLK Kernel. Many missions may complete their entire “lives” with only a single clock partition. Other missions might go for years before a new partition is needed. Still others might have many clock partitions. The path to this file is specified in the **sclkPartitionMap.path** configuration parameter. Raw SPICE SCLK values can be written <*partition*>/<*coarse*>:<*fine*>. Where the first value is the partition number, the second value the SCLK coarse time (e.g., seconds), and the third value the SCLK fine time (e.g., subseconds). For example: “1/876545321:3456”. The MMTC uses the partition map file to associate a ground time with the beginning of a partition.

Partition Number, Date

1,2010-01-01T00:00:00Z

2,2010-07-02T20:00:00Z

3,2010-11-01T00:00:00Z

Figure 7 Sample SCLK Partition Map file

## Ground Stations Map File

The Ground Stations Map File contains a mapping of ground station antenna identifiers that are indicated in frame ground receipt headers to corresponding ground station identifiers known to SPICE through the NAIF generic ground stations ephemeris kernel (earthstns\_fx\_<*nnnnnn*>.bsp). Every DSN ground station antenna has a NAIF ID. The MMTC uses the information in this file to obtain the NAIF ID of a receiving ground station when computing OWLT.

This file is a CSV formatted text file. Each row contains three fields, a Path ID, a Station ID, and a Station Name. The **Path ID** is provided by the DSN or ground simulators with each received telemetry frame. It is the value given in the **dssId** field in AMPCS metadata. The **Station ID** is the DSN identifier of the station that can be passed to a SPICE function to get the SPICE identifier for the station. The **Station Name** is a more human-readable name for the station. The A sample file is provided with the MMTC distribution and will only need to be edited if the DSN adds or changes a station or for simulation equipment. The path to this file is indicated in the **groundStationMap.path** configuration parameter. Figure 8 gives a fragment of the Ground Stations Map file as an example. This is not the full file.

Path ID, Station ID, Station Name

14,dss-14,Goldstone\_70M

15,dss-15,Goldstone\_34HEF

24,dss-24,Goldstone\_34B1

25,dss-25,Goldstone\_34B2

26,dss-26,Goldstone\_34B3

34,dss-34,Canberra\_34B1

35,dss-35,Canberra\_34B2

36,dss-36,Canberra\_34B3

43,dss-43,Canberra\_70M

45,dss-45,Canberra\_34HEF

46,dss-46,Canberra\_26M

54,dss-54,Madrid\_34B1

55,dss-55,Madrid\_34B2

63,dss-63,Madrid\_70M

65,dss-65,Madrid\_34HEF

Figure 8 The Ground Stations Map File

# Time Correlation Data Tables

The MMTC creates the data tables the first time it runs and appends to them each time it runs thereafter, so they must be preserved. If over the course of a mission, they become inconveniently large, the records created from runs from more than ten contacts prior or so can be deleted. They can also be deleted and the MMTC will recreate them; however, after this is done the Contact Filter must be turned off before the next MMTC run. It is recommended that a copy of the intact original files be stored somewhere for future analysis.

## Raw Telemetry Table

The Raw Telemetry Table contains the unprocessed data extracted from telemetry used in time correlation. It contains the basic SCLK, ERT, and virtual channel information. The data contained in this table, along with the associated SPICE kernels and configuration parameters, contain the information needed to run the Time Correlation application and, when necessary, to reproduce the results of a previous time correlation and its output products (SCLK kernel, SCLK/SCET file, command uplink file) without having to connect to the telemetry archive again. This is a comma separated value (CSV) formatted text file. It is useful for analysis and for later reprocessing if needed. It contains records of all frames/TK Packets in the sample set and is appended to each time the application runs. The fields in the table are described in Table 8 in the Appendix.

## Summary Table

The SCLK Summary Table contains record of the processed time correlation, containing the computed values and inputs. The telemetry data that are extracted from the telemetry archive passes through several filtering steps within MMTC to select a suitable set of telemetry to use for time correlation. The set of telemetry data that is selected for time correlation is saved in the SCLK Summary Tables. As the data is processed, all computed values are saved in the SCLK Summary Tables. It is needed for the Contact Filter as well as for analysis and is a CSV-formatted text file. One record is appended to this table each time the MMTC runs and this record contains information on the Target and Supplemental frame/packet used in the time correlation computations. If this file is deleted, the application will generate a new one and run although the Contact Filter should be turned off in the configuration file for the next few runs. This table is described in Appendix G.

# MMTC Output Products

## Output SCLK Kernel

Each time the MMTC runs, it creates a new SCLK kernel. SCLK kernels follow the NAIF specifications (SCLK – *Reference for the SPICE spacecraft clock subsystem and the Spacecraft Clock Kernel (SCLK)*) and is a primary output product. The new SCLK kernel contains the entire contents of the previous SCLK kernel that it read as input, but has one new time correlation record appended to the end. In addition, if the user has selected interpolated clock change rate computation (the default), the clock change rate of the last record of the previous kernel will be overwritten with a new and more accurate value. The new kernel will have the same name as the preceding one, except that the version number in the name will have been incremented.

The MMTC names the SCLK kernel files by <*basename*><*sep*><*nnnn*>.<*suffix*>, where:

*basename* is indicated in the **spice.kernel.sclk.baseName** parameter in configuration data. This is often the mission name.

*sep* is indicated in the **spice.kernel.sclk.separator** parameter in configuration data. This is typically a hyphen or underscore.

*Nnnn* is the integer version number that is incremented with each run.

*Suffix* is indicated in the **spice.kernel.sclk.suffix** parameter in configuration data. Normally, this value will be “tsc” per NAIF conventions.

For example, setting *basename* to “europaclipper”, *sep* to “\_”, and *suffix* to “tsc” in the configuration parameters would produce the SCLK kernel “**europaclipper\_00123.tsc**”, if the previous version was 122.

It writes the SCLK kernels to the directory indicated in **spice.kernel.sclk.kerneldir**. The existing input SCLK Kernel must reside in this directory and the new output SCLK kernel will be written to it.

### Input SCLK Kernel Override

The MMTC provides a configuration parameter **spice.kernel.sclk.inputPathOverride** that directs it to read the source (current) SCLK Kernel not from the normal **spice.kernel.sclk.kerneldir** from which it normally derives its input SCLK kernel, but from an alternate SCLK Kernel. This is intended for use only in test venues. On occasion, it is convenient when testing to read an SCLK kernel other than the one resident in the directory that the new one will be written to. This can cause versioning problems and duplicate SCLK Kernel filename errors in successive runs. Therefore, this option is only for certain test situations and should never be used operationally. To use this, set **spice.kernel.sclk.inputPathOverride** to the fully qualified path of the alternate SCLK Kernel. Remove it before running again. Successive runs will likely fail if this parameter is not removed after the first run.

## SCLK/SCET File

Each time the MMTC runs, it optionally creates a new SCLK/SCET file. SCLK/SCET files follow the NAIF specifications (Generic SCLK versus SCET Correlation File, Software Interface Specification, Rev. E, January 2012). This is a primary output product. The new SCLK/SCET file contains the entire contents of the previous SCLK/SCET file, but has one new time correlation record appended to the end. The version number of the file is incremented each time a new one is produced. The MMTC creates the new SCLK/SCET file from the new SCLK kernel; however, in order to conform with specifications and convention, it adds two records at every point in time where a leap second was added to world timekeeping. The SCET0 fields of the two added leap second records match, but the SCLKRATE of the first record is set to 0.00100000000 and that of the second record to 1.0000000000. Additionally, the SCLK0 field of the second record is incremented by 1 second. This brackets the leap second insertion in world time. The fraction of second part of the SCET0 fields in the added leap second records is set to the fraction of second value of the first record in the SCLK/SCET file with 1 ms subtracted from that of the first record, if its fractional seconds part is non-zero. Again, these modifications are only done to the inserted leap second records, not to those created by time correlation calculations.

The new SCLK/SCET file will have the same name as the preceding one, except that the version number in the name will have been incremented.

The MMTC will create SCLK/SCET files if the **product.sclkScetFile.create** configuration parameter is set to *true*. It writes them to the directory specified in **product.sclkScetFile.dir**.

The MMTC names the SCLK/SCET files by <*basename*><*sep*><*nnnn*>.<*suffix*>, where:

*basename* is indicated in the **product.sclkScetFile.baseName** parameter in configuration data. This is often the mission name.

*sep* is indicated in the **product.sclkScetFile.separator** parameter in configuration data. This is typically a hyphen or underscore.

*Nnnn* is the integer version number that is incremented with each run.

*Suffix* is indicated in the **product.sclkScetFile.suffix** parameter in configuration data.

For example, setting *basename* to “europaclipper”, *sep* to “\_”, and *suffix* to “coeff” in the configuration parameters would produce the SCLK kernel “**europaclipper\_00123.coeff**”, if the previous version was 122.

## Uplink Command File

The Uplink Command File is an optional operations product that can be used to uplink time correlation data to the spacecraft. It is a simple CSV formatted text file with a single line in it. It contains five values: SCLK coarse, ground time in ephemeris time in seconds of epoch (ET also called TDB), ground time in TDT in seconds of epoch, the identical ground time TDT as a calendar string, and the clock change rate. The fine time SCLK (i.e., subseconds) is not included in this file because the corresponding ground time is adjusted, by an amount equal to the fine time, to a whole tick SCLK coarse time; in other words, the subseconds are subtracted from the TDB and TDT ground times. The SCLK1, TDT1, and CLKCHGRATE1 values which the flight software uses to compute ground time from SCLK derive from these. It is left to other elements of the space mission ground system to convert the values in this file to the necessary format and load them into a command for uplink. For missions that use this product, it is left to the mission’s GDS to take this file and form its contents into a command that can be uplinked. See Appendix D for a sample of this file.

This file is produced if the **product.uplinkCmdFile.create** configuration parameter is set to *true*, or if the **–generate-cmd-file** or **-c** option is provided at the command line. If the command line option is provided, it will override the configuration parameter.

## Time History File

The Time History File is a cumulative product that is updated with a single record appended to its end each time the MMTC runs. It is a CSV format text file that contains information useful in assessing the state of the onboard clock. This file is intended for analysis purposes. Table 7in section 15 Appendix E describes the fields in this file.

# Filters

The MMTC implements a set of data filters that ensure the quality of processed time correlation data. They are intended to protect the integrity of the cumulative SCLK Kernel and SCLK/SCET file by rejecting TK data from packets or frames that are corrupted or otherwise should not be used for time correlation computations. These filters can be turned on or off in the configuration parameters. When a data sample fails any one of these filters, it will be rejected. In I&T environments, it will often be necessary to turn off some or all of the filters, since the input time correlation data might not be of sufficient quality to pass them.

## Contact Filter

The Contact Filter compares the clock drift rate from the previous contact with that computed for the current contact in ∆SCLK/∆ERT seconds. It computes a drift rate using the formula:

*driftRate = (∆SCLK/∆TDT(G) – 1) \* milliseconds\_per\_day*

If the drift rate falls outside the bounds specified in the configuration parameters, it throws a fatal exception and processing terminates. Note that this behavior is different than that of the other data quality filters. If any of them fail, the application attempts to create another sample set from data earlier in the contact and if it is able to, processing continues with the new sample set. The reason that the Contact Filter is different in this regard is because it needs to compute TDT(G) for the new target sample and this is not done until after the target sample is selected. Also, if the drift rate is outside the specified thresholds, human investigation is probably called for.

In order to continue processing, the user can rerun the MMTC on the contact, but setting the Stop Time to a time before the beginning of the sample set from the previous run. This will ensure that the problem sample is not included in the new sample set.

The Contact Filter is turned on or off by setting the **filter.contact.enabled** configuration parameter to **true** or **false** and the upper and lower bounding failure thresholds are set in floating point units of ms/day in the **filter.contact.deltaLowerThreshold** and **filter.contact.deltaUpperThreshold** configuration parameters.

## SCLK Filter

The SCLK Filter compares the SCLKs of two consecutive supplemental frames in a sample set. If they differ by more than the specified number of seconds, the filter fails and the sample set is rejected. This filter is turned on or off by the **filter.sclk.enabled** configuration parameter and the failure threshold is set in the **filter.sclk.deltaThresholdSec** configuration parameter. The threshold value is in seconds.

NOTE: This filter is not applicable and should be turned off when using the AMPCS telemetry source plugin’s AmpcsTlmWithFrames mode. This is because the target and supplemental samples do not follow each other in a regular order.

## ERT Filter

The ERT Filter compares the ERTs of two consecutive target frames in a sample set. If they differ by more than the specified number of seconds, the filter fails and the sample set is rejected. This filter is turned on or off by the **filter.ert.enabled** configuration parameter and the failure threshold is set in the **filter.ert.deltaThresholdSec** configuration parameter. The threshold value is in seconds.

NOTE: This filter is not applicable and should be turned off when using the AMPCS telemetry source plugin’s AmpcsTlmWithFrames mode. This is because the target and supplemental samples do not follow each other in a regular order.

## Data Rate Filters

Very slow data rates can adversely affect the accuracy of ERT values attached to received frames. In the case of some missions, time correlation accuracy can also be affected by excessively high data rates. The Minimum Data Rate Filter and Maximum Data Rate Filter set downlink data rate thresholds within which time correlation data will be processed. The filters are turned on or off by setting the **filter.minDataRate.enabled** and **filter.maxDataRate.enabled** configuration parameters to **true** or **false**. (Note that each filter can be turned on or off independently of the other.) The failure thresholds are set in the **filter.dataRate.minDataRateBps** and **filter.dataRate.maxDataRateBps** configuration parameters. The threshold values are specified in integer units of bits per second and are *exclusive*; for example, if the Maximum Data Rate Filter is enabled, a sample whose downlink data rate exactly equals **filter.dataRate.maxDataRateBps** will fail the filter and be rejected.

## Stations Filter

The Stations Filter specifies the ground stations from which the MMTC may use time correlation data. This filter can be turned on or off by the **filter.groundStation.enabled** configuration parameter. A list of ground station Path IDs is provided in the **filter.groundStation.pathIds** configuration parameter. If this filter is enabled and a received time correlation sample was received by a ground station or specific antenna (Path) not in this list, it will be rejected. For various reasons, one might want to receive data only from certain ground stations.

## Valid Sample Filter

The Valid Sample Filter applies only to missions where time correlation data are being obtained from TK packets and a validity or quality flag is provided in the packets. It checks a validity flag that might be in the packet based on the Packet Definition File (sec. 8.3). If this filter is enabled by setting the **filter.validFlag.enabled** configuration parameter to true, the MMTC will reject samples where the valid flag value is set to 1 indicating that the packet is bad. Missions that do not use TK packets or whose packets do not contain such a quality flag should disable this filter.

## Consecutive Frames Filter

The Consecutive Frames Filter performs the 5 checks described below. If any check fails, the filter rejects the samples.

* The filter checks that all samples in the set have the same VCID. If not, the check fails.
* The filter checks that all samples in the sample set have consecutive VCFCs. If not, the filter logs a warning message, but this is not considered a failed check. In other words, despite any warning message, this check never causes the filter to reject samples.
* If supplemental sample VCIDs are available, the filter checks that each sample’s VCID matches its supllemental sample’s VCID. If not, the check fails. If supplemental sample VCIDs are not available, the filter logs a warning message, but it is not considered a failed check.
* If sample and supplemental sample VCFCs are available, the filter checks that each sample’s VCFC is behind (meaning it is less than) its supplemental sample’s VCFC by exactly the number specified in the **telemetry.supplementalSampleOffset** configuration parameter. If not, the check fails. If sample and supplemental sample VCFCs are not available, the filter logs a warning message, but it is not considered a failed check.
* If sample VCFCs are available, the filter checks that each sample’s VCID and VCFC matches its supplemental sample’s “TK” or target frame VCID and VCFC, respectively. If not, the check fails. If sample VCFCs are not available, the filter logs a warning, and it still checks sample VCIDs and supplemental sample TK VCIDs.

This filter can be turned on or off using the **filter.consecutiveFrames.enabled** configuration parameter. In operational venues, this filter should be enabled.

NOTE: This filter is not applicable and should be turned off when using the AMPCS telemetry source plugin’s AmpcsTlmWithFrames mode. This is because the target and supplemental samples do not follow each other in a regular order.

## VCID Filter

The Virtual Channel ID Filter verifies that time correlation data are received only from specified virtual channels. For example, a particular mission might specify that only time correlation data contained on frames received from VCID 0 or VCID 32 may be used for time correlation. If enabled, the MMTC will reject any sample data not received from any of those VCIDs. The valid time correlation virtual channels are listed in configuration parameter **filter.vcid.validVcids** as a comma-separated list. This filter may be turned on or off by the **filter.vcid.enabled** parameter set to *true* or *false*.

NOTE: This filter is not applicable and should be turned off when using the AMPCS telemetry source plugin’s AmpcsTlmWithFrames mode. This is because the target and supplemental samples do not follow each other in a regular order.

# Log File

The MMTC writes processing status messages to a log file named **mmtc.log**. Logging is implemented using the Apache Java **log4j2** **LogManager**, which recognizes six levels of logging: FATAL, ERROR, WARNING, INFO, DEBUG, and TRACE. A FATAL or ERROR message indicates that something is seriously wrong. This will usually cause the MMTC to terminate abnormally. A WARNING message does not necessarily indicate a problem, but something unusual happened and an operator should investigate. INFO messages are normal processing messages. DEBUG and TRACE messages are used for debugging.

The **log4j2.properties** file which is specified in an option to the JRE (see section 4.1) controls how logging is done. The MMTC ships with a **log4j2.properties** file that sets the log level to DEBUG; in normal operational venues, missions may wish to change this by setting the **rootLogger.level** property to “info” or higher since the DEBUG log level may produce excessive output. The **log4j2.properties** file that MMTC ships with also causes log messages to be appended to **mmtc.log** each time the MMTC is executed, to “roll over” the log file to a new file when it reaches 100MB in size or when a week has elapsed since the log file was started, and to delete the oldest files when there are 5 or more rollover files. For more information about how to customize this rollover behavior and other aspects of logging, consult the Apache [Log4j 2 documentation (https://logging.apache.org/log4j/2.x/)](https://logging.apache.org/log4j/2.x/).

Note that log messages may contain information that missions or users consider sensitive, such as hostnames, filesystem paths, even database credentials passed through to telemetry sources using the --connection-parms command line option, etc. If log files are being submitted for MMTC troubleshooting, be sure to first remove any sensitive information.

# Operating Environment

The MMTC is tested and validated using a standard Java JRE 1.8 running on the commercial Red Hat Linux 8 operating system hosted on Intel-based hardware; however, it is designed for platform-independence and should work or at least be easily ported to any other modern computing platform running a current Java JRE. It does not rely on any specialized hardware or software.

# Support Concept

The Johns Hopkins Applied Physics Laboratory, Space Exploration Sector (SES) owns and maintains the MMTC and provides it to the AMMOS catalog. APL can provide support based upon a supporting contract.

# Appendix

1. Description of Time

**Terrestrial Dynamical Time**

The MMTC uses the terrestrial time scale Terrestrial Dynamical Time (TDT), which is in common use for space mission timekeeping including within JPL/NAIF SPICE kernels. TDT is an atomic time based on the International System (SI) second and referenced to the Earth. This time form is also called “Terrestrial Time” or “TT,” but that acronym is ambiguous as it sometimes refers to another time scale, so we use the term “TDT.” MMTC correlates the spacecraft clock (SCLK) with TDT exclusively.

Most people are familiar with UTC, which is a practical time scale that includes leap seconds and estimates of UTC are distributed throughout the world. The relationship between UTC and TDT is given below.

*UTC = TDT – leap seconds – 32.184 seconds*

Both UTC and TDT are defined relative to another nominally atomic time scale called International Atomic Time (TAI).

By definition, *UTC = TAI – leap* *seconds*, exactly.

At the instant 1977 January 1, 0h 0m 0s TAI, TDT = TAI + 32.184 seconds exactly, but TAI and TDT have diverged since by a small amount due to imperfections in atomic time standards. The 32.184 seconds offset of TDT from TAI is a constant defined by the International Astronomical Union (IAU).

TDT(G) is the TDT computed on the ground.

TDT(S) is the TDT computed onboard the spacecraft.

**Ephemeris Time**

Ephemeris Time (ET) is the same as Barycentric Dynamical Time (TDB by its French acronym). This is similar to TDT, but is a theoretical value referenced to the Solar System Barycenter. In other words, if one imagines a hypothetical atomic clock located at the Solar System Barycenter, this is the time it would measure. ET and TDT will differ slightly due to relativistic effects based upon where the Earth is in its orbit around the Sun. However, ET and TDT are so close together that TDT may be used in place of ET for ephemeris mapping. The MMTC uses ET internally, but its time correlations are given in TDT and UTC.

1. Sample SCLK Kernel

Snippet of SCLK Kernel “triplets” \*

**Encoded SCLK Earth Time (TDT) Clock Rate\*\***

18727147700000 @02-DEC-2017-17:51:44.940590 1.00000001423

18727326900000 @02-DEC-2017-18:51:28.940641 1.00000001129

18734707450000 @04-DEC-2017-11:51:39.942308 1.00000001132

18739028600000 @05-DEC-2017-11:52:02.943286 1.00000001191

18748746550000 @07-DEC-2017-17:51:21.945601 1.00000001106

18752526200000 @08-DEC-2017-14:51:14.946437 1.00000001153

18757387200000 @09-DEC-2017-17:51:34.947558 1.00000001119

18761704650000 @10-DEC-2017-17:50:43.948524 1.00000001113

18764764600000 @11-DEC-2017-10:50:42.949205 1.00000001138

18769084150000 @12-DEC-2017-10:50:33.950188 1.00000001136

18774672100000 @13-DEC-2017-17:53:12.951458 1.00000001079

18777182700000 @14-DEC-2017-07:50:04.952000 1.00000001132

18782043000000 @15-DEC-2017-10:50:10.953100 1.00000001127

18787622700000 @16-DEC-2017-17:50:04.954358 1.00000001119

18791942050000 @17-DEC-2017-17:49:51.955325 1.00000001122

18796261750000 @18-DEC-2017-17:49:45.956294 1.00000001055

18798422900000 @19-DEC-2017-05:50:08.956750 1.00000001121

\* Example from New Horizons. An SCLK kernel also contains information describing the spacecraft clock and time partitions.

\*\* Clock Rate is in TDT seconds per SCLK second

1. Sample SCLK/SCET File

Section of SCLKvsSCET File \*

CCSD3ZS00001$$sclk$$NJPL3KS0L015$$scet$$

MISSION\_NAME=new\_horizons;

SPACECRAFT\_NAME=new\_horizons;

DATA\_SET\_ID=SCLK\_SCET;

FILE\_NAME=/project/new\_horizons/spice/kernels/current//../coeff/new-horizons\_1544.coeff;

PRODUCT\_CREATION\_TIME=2018-018T18:50:36;

PRODUCT\_VERSION\_ID=1544;

PRODUCER\_ID=sclk\_to\_scet\_2;

APPLICABLE\_START\_TIME=2006-019T18:08:00;

APPLICABLE\_STOP\_TIME=2018-181T23:59:59;

MISSION\_ID=98;

SPACECRAFT\_ID=98;

CCSD3RE00000$$scet$$NJPL3IS00613$$data$$

\*\_\_\_\_SCLK0\_\_\_\_\_\_\_ \_\_\_\_SCET0\_\_\_\_\_\_\_\_\_\_\_\_ *DUT*\_ *SCLKRATE*\_\_

000000000000.000 2006-019T18:08:00.000 65.184 1.0000000000

000000055325.000 2006-020T09:30:07.299 65.184 1.0000000091

000000103927.000 2006-020T23:00:09.300 65.184 1.0000000093

000000192133.000 2006-021T23:30:15.301 65.184 1.0000000094

(…)

000375838841.000 2017-351T17:48:42.771 69.184 1.0000000112

000375925235.000 2017-352T17:48:36.772 69.184 1.0000000106

000375968458.000 2017-353T05:48:59.773 69.184 1.0000000112

CCSD3RE00000$$data$$CCSD3RE00000$$sclk$$

\* Example from New Horizons.

1. Sample Uplink Command File

An example of an Uplink Command File as described in section 10.3 created from New Horizons data. The values in order are: (1) SCLK coarse, (2) corresponding ephemeris time (TDB), (3) corresponding ground time in TDT, (4) corresponding ground time in TDT calendar string, (5) clock change rate in TDT seconds/SCLK second. This file will contain only a single record.

398916214,714492283.860548,714492283.861766,23-AUG-2022-02:04:43.861766,1.00000000922

1. Time History File Fields

The default fields contained in the Time History File are described in Table 7 below.

| Name | Typical Value | Remarks |
| --- | --- | --- |
| Enc\_SCLK | Positive integer – the number of SCLK ticks since the epoch for SCLK = 0 | Encoded SCLK. |
| Int\_SCLK | positive integer < 232 seconds = 4,294,967,296 seconds | Integer SCLK. |
| TDT(G) | X.Y, in units of 10-6 second | Terrestrial Dynamical Time – Ground. Numeric value in seconds of epoch. The ground time corresponding to the SCLK in TDT. |
| TDT(G)\_Calendar | 10-MAR-2018-05:56:23.825936 (example) | Terrestrial Dynamical Time – Ground. Calendar string representation used in SCLK Kernels. The ground time corresponding to the SCLK in TDT. |
| TDB(G) | X.Y, in units of 10-6 second | Barycentric Dynamical Time (a.k.a. Ephemeris Time) – Ground equivalent. Numeric value in seconds of epoch. Time corresponding to the SCLK measured in Barycentric Dynamical Time (TDB). This is similar to TDT except that it is measured from the Solar System barycenter rather than from Earth.  NOTE: In the MMTC code, this value is given as “ET” in order to match how it is used in SPICE. The term “ET” is otherwise ambiguous. For the MMTC, ET is always a synonym for TDB. |
| ClkChgRate(s/s) | X.Y, where X is “0” or “1” and Y is in units of second/second | Predicted clock Change Rate in TDT seconds/SCLK seconds. No change would be 1.000. |
| Osc\_Temp(degC) | X.Y in units of ℃ | Active Oscillator temperature. |
| Ep(ms) | X.Y in units of ms | Error in Prediction. |
| |Ep/dt|(ms/day) | X.Y in units of ms/day | Error in Prediction normalized over a day. |
| Interval(days) | X.Y in units of days | Number of days between the current and last time correlation. |
| ChgRateMode | “P”, “I”, “N”, or “A” | Clock change rate mode.   * P : Compute predicted * I : Compute interpolate (recompute previous run value) * N : No drift (set to 1.000000000) * A : Assigned (set by config parm or user option) |
| ChgRateInterv(days) | X.Y in units of days | Clock change rate interval (i.e., the number of days to look back into previous time correlations to get a previous TDT(G) and SCLK for clock change rate computations. This is the value from the **compute.tdtG.rate.predicted.lookBackDays** configuration parameter converted to days. |
| SCLK\_Drift(ms/day) | X.Y in units of ms/day | Similar to clock change rate, but given in drift in SCLK per day in milliseconds/day. No drift would be 0.0 ms/day. |
| Oscillator | Typically 1 or 2, etc. | Active Oscillator. Applicable when there are two or more timekeeping (“precision”) oscillators. |
| SCLK\_Partition | Positive Integer | SCLK clock partition defined in SCLK kernel. |
| RF\_Encoding | “T” or “C” | The frame encoding method. Typically “T” for turbo or “C” for convolutional encoding; other encodings may be deployed in the future. |
| GroundStation | DSN-xx if a DSN station, where xx is the DSN station number | Receiving ground station. Often, but not always a DSN station. |
| OWLT(sec) | X.Y in units of seconds | One-Way Light (travel) Time. |
| SpaccraftSunDistance(AU) | X.Y in units of AU | Spacecraft to Sun distance at TDT(G) in Astronomical Units. |
| SpaccraftSunDistance(km) | X.Y in units of km | Spacecraft to Sun distance at TDT(G) in kilometers. |
| SpacecraftEarthDistance(km) | X.Y in units of km | Spacecraft to Earth distance at TDT(G) in kilometers. |
| EarthSunDistance(AU) | X.Y in units AU | The Earth-Sun distance at TDT(G) in Astronomical Units. |
| EarthSunDistance(km) | X.Y in units of km | The Earth-Sun distance at TDT(G) in kilometers. |
| SpacecraftVelocity(SSB\_km/s) | X.Y in units of km/sec | Velocity of spacecraft with respect to the Solar System Barycenter at TDT(G). |
| EarthVelocity(SSB\_km/s) | X.Y in units of km/sec | Velocity of Earth with respect to the Solar System Barycenter at TDT(G). |
| SpacecraftVelocity(Earth\_km/s) | X.Y in units of km/sec | Velocity of spacecraft with respect to the Earth at TDT(G). |
| The Active Radio | Typically 1 or 2, etc. | The active radio. Applicable only when an oscillator on a radio sets the SCLK values used for time correlation. Otherwise, defaults to “-”. |
| DataRate(bps) | Positive integer in units of bps | Transfer frame downlink rate in bits per second. |
| SCET | Spacecraft Event Time in UTC | UTC ISO calendar string *YYYY-DOY***T***hh:mm:ss.sss.* This is the correlated UTC time equivalent of TDT(G). This is the value that is written to the SCET0 field in the SCLK/SCET file. |
| TDT(S) | X.Y, with precision of 10-6 seconds | Terrestrial Dynamical Time as known onboard the spacecraft in numeric seconds of epoch format. |
| TDT(S)\_Calendar | 10-MAR-2018-05:56:23.825936 (example) | Terrestrial Dynamical Time as known onboard the spacecraft in calendar string format. |
| TDT(S)\_Error(ms) | X.Y with precision of of 10-3 ms | Error in TDT(S) measured in milliseconds. |
| SCLK\_for\_TDT(S) | positive integer < 232 seconds | SCLK corresponding to TDT(S). |
| ClockChangeRateForTDT(S) | Floating point number | The clock change rate used onboard the spacecraft for computing ground time from SCLK in seconds/second. |
| TDT(S)ErrorThreshold(ms) | positive integer in units of ms | Error threshold in TDT(S) read from configuration parameter in milliseconds. |
| TDT(S)AlarmThreshold(ms) | positive integer in units of ms | Alarm threshold in milliseconds on the magnitude of the TDT(S) error read from configuration parameter. When exceeded, the time correlation onboard the spacecraft needs to be updated. |
| SCLK1 | X.Y, with precision of 10-6 second | SCLK1 from spacecraft used to compute TDT(S). |
| TDT1 | X.Y, with precision of 10-6 second | TDT1 from spacecraft used to compute TDT(S). |
| TDT1\_Calendar | The TDT1 as a calendar string | TDT1 from spacecraft used to compute TDT(S) in a calendar string form dd-mon-yyyy-hh:mm:ss.ssssss. |
| RunTime | 2018-086T19:29:48 (example) | System time in UTC in ISO format when MMTC was run. |
| Warning | Plain text (no spaces between words). | Typically blank. Contains a message if the TDT(S) Error threshold is exceeded. |
| Alarm | Plain text (no spaces between words). | Typically blank. Contains a message if the TDT(S) Alarm threshold is exceeded. |

Table 8 The Time History File

1. The RawTlmTable

| Name | Typical Value | Remarks |
| --- | --- | --- |
| Run Time | ISO Calendar Time String | The system time in UTC when the MMTC was run. |
| Path ID | Integer | Code for the ground station or testbed front end from which the TLM were received. |
| Target Frame ERT | CDS Time Code | The ERT of the target frame. Used in computing TDT(G). |
| Target Secondary Hdr SCLK Coarse | Unsigned Integer | The SCLK coarse value contained within either the Target Frame secondary header or the target frame packet secondary header. This value is not used in time correlation calculations and is NOT to be confused with the Supplemental Frame/Packet SCLK which is. |
| Target Secondary Hdr SCLK Fine | Unsigned Integer | The SCLK fine value contained within either the Target Frame secondary header or the target frame packet secondary header. This value is not used in time correlation calculations and is NOT to be confused with the Supplemental Frame/Packet SCLK which is. |
| Supp SCLK Coarse | Unsigned Integer | Time SCLK coarse value used for time correlation. Depending on the mission, this is either the SCLK contained in the supplemental frame secondary header or in the selected time correlation packet. |
| Supp SCLK Fine | Unsigned Integer | Time SCLK fine value used for time correlation. Depending on the mission, this is either the SCLK contained in the Supplemental Frame secondary header or in the selected time correlation packet. |
| Supp Frame ERT | CDS Time Code | The ERT of the Supplemental Frame or the frame containing the selected time correlation packet. This value is not used in any calculations. |
| Encoding Type | Unsigned Integer | The mission-specific code for the type of encoding used for the frames. This is usually Turbo, Convolutional, or No Encoding. |
| MCFC | Unsigned Integer | The Master Channel Frame Count of the Target Frame. This is used only on missions which uses traditional transfer frames to do time correlation. Can be null “-“. |
| VCID | Unsigned Integer | The Virtual Channel ID of the Target Frame. |
| VCFC | Unsigned Integer | The Virtual Channel Frame Count of the Target Frame. |
| Target ERT String | DOY Time String | The ERT in day-of-year calendar string form of the Target Frame. |
| Data Rate BPS | Unsigned Integer | The downlink data at which the Target Frame was received in bits per second. |

Table 9 The Raw Telemetry Table

1. The Summary Table

| Name | Typical Value | Remarks |
| --- | --- | --- |
| Run Time | ISO Calendar Time String | The system time in UTC when the MMTC was run. |
| Ground Station | String | The ground station at which the Target Frame was received. |
| Target Frame ERT | CDS Time Code | The ERT of the Target Frame. Used in computing TDT(G). |
| Target Frame UTC | ISO Calendar Time String | The ERT of the Target Frame as a calendar string. Used in computing TDT(G). |
| Target Frame VCID | Unsigned Integer | The Virtual Channel ID of the Target Frame. |
| Target Frame VCFC | Unsigned Integer | The Virtual Channel Frame Count of the Target Frame. |
| TK Encoded SCLK | Unsigned Integer | The SCLK coarse value in SPICE encoded SCLK used in time correlation computations. It is obtained from the body of the time correlation packet and is associated with the preceding target frame. |
| SCLK Partition | Unsigned Integer | The SCLK partition currently in use. |
| Predicted Clk Change Rate | Double | The Predicted CLKRATE. This value is written to the latest record of the SCLK Kernel and SCLK/SCET files. If the next run is in Interpolate mode, this record will be overwritten in the next SCLK Kernel and SCLK/SCET file with an updated value. |
| Interpolated Clk Change Rate | Double | The Interpolated CLKRATE. This value is written to the last record of the SCLK Kernel and SCLK/SCET from the previous run. If the current run is in Interpolate mode. This value replaces the CLKRATE from the previous run. The SCLK Kernel and SCLK/SCET files from the previous run are not changed. The second to the last record in the new files are. |
| OWLT | Double | The one-way light travel time from the spacecraft to the receiving ground station. |
| TDT(G) | Double | The ground time in TDT written to the SCLK Kernel and used to compute the CLKRATE. |
| TD SC (sec) | Double | The spacecraft time delay in seconds. This is a constant value set in configuration parameters (**spacecraft.timeDelaySec**) that is subtracted from the computation of TDT(G). It is the time delay from the SCLK reported by the spacecraft to when the related downlink frame leaves the spacecraft antenna. With careful spacecraft design, this value can be effectively 0 at all downlink bit rates. TD SC can be useful during mission simulations prior to launch to bias TDT(G) to a future time. This value is usually only a fraction of a second and is often set to 0. |
| TD BE (sec) | Double | The bitrate error. This is an adjustment for the bitrate-dependent time error due to a mismatch between frame times reported by the spacecraft and the receiving ground station. This value is divided by the downlink data rate and subtracted from the TDT(G) computation. It is typically only a fraction of a second and is often set to zero. |
| TF Offset | Double | This value adjusts the TDT(G) value to align with the whole second SCLK value. It is computed by dividing the SCLK fine time from the supplemental frame or selected TK packet by the subseconds modulus to convert it to decimal fraction of a second and then subtracting it from the TDT(G). This simplifies time correlations with no loss in accuracy. The subseconds modulus is read from the configuration parameter **spacecraft.sclkModulusOverride** if provided. If it is not provided, the value is read from the second stage of the **SCLK01\_MODULI\_<***nnn*> field in the SCLK Kernel. |
| Data Rate | Unsigned Integer | The downlink data rate at which the Target Frame was received in bits per second. |
| Frame size (bits) | Unsigned Integer | The size in bits of a transfer frame used for time correlation or of a frame which can contain a TK packet. This is a constant set in configuration parameters (**telemetry.frameSizeBits**). |

Table 10 The Summary Table

1. Example TK Packet Definition File

The packet given below is from the Europa Clipper mission.

Figure 9 Example TK Packet Definition File

<!-- Last SCLK Missed Flag (Indicates invalid data if bit is set) -->

<TelemetryPoint>

<name>ValidFlag</name>

<offset>159</offset>

<length>1</length>

</TelemetryPoint>

<!-- Last Frame VCID (Virtual Channel ID of the target frame) -->

<TelemetryPoint>

<name>TargetFrameVcid</name>

<offset>162</offset>

<length>6</length>

</TelemetryPoint>

<!-- Last Frame VCFC (Virtual Channel Frame Count of the target frame) -->

<TelemetryPoint>

<name>TargetFrameVcfc</name>

<offset>168</offset>

<length>24</length>

</TelemetryPoint>

<!-- Last Frame Encoding Mode (Encoding method of the target frame) -->

<TelemetryPoint>

<name>EncodingMethod</name>

<offset>192</offset>

<length>8</length>

</TelemetryPoint>

<!-- Last Frame Downlink Data Rate in bits per second (Downlink rate of the target frame) -->

<TelemetryPoint>

<name>DownlinkDataRate</name>

<offset>200</offset>

<length>32</length>

</TelemetryPoint>

</Packet\_Definition>

<?xml version="1.0"?>

<!--

Definition of the Europa Clipper Time Correlation Packet.

This definition is based upon that given in the "Europa Clipper Flight Ground ICD", 05 November 2019

(JPL D-56521).

All offsets are in bits from the beginning (bit 0) of the packet. All lengths are in bits. All data

types are unsigned integers.

-->

<Packet\_Definition

xmlns="mmtc"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="mmtc tk\_packet.xsd">

<!-- Bit 0 Timestamp (seconds), Coarse part of SCLK associated with the target frame, but contained in the current frame -->

<TelemetryPoint>

<name>SclkCoarse</name>

<offset>96</offset>

<length>32</length>

</TelemetryPoint>

<!-- Bit 0 Timestamp (subseconds), Fine part of SCLK associated with the target frame, but contained in the current frame -->

<TelemetryPoint>

<name>SclkFine</name>

<offset>128</offset>

<length>16</length>

</TelemetryPoint>

1. Example Seed SCLK Kernel

Below is the seed kernel for the Europa Clipper mission. The first time that the MMTC runs, it copies the entire contents of this file into its new SCLK Kernel. It changes the FILENAME and CREATION\_DATE fields. After computing time correlations, it appends its new time correlation record to the end of its new SCLK kernel.

KPL/SCLK

FILENAME = "europaclipper\_00000.tsc"

CREATION\_DATE = "05-Jul-2018"

Europa Clipper Spacecraft Clock Kernel (SCLK)

=================================================================

This file is a SPICE spacecraft clock (SCLK) kernel

containing information required for time conversions

involving the on-board Europa Clipper spacecraft clock ('SCLK').

Version

--------------------------------------------------------

Version History of Europa Clipper Seed Kernel:

Version 1.0 -- July 05, 2018 -- Stan Cooper

Initial (seed) time coefficients triplet set for SCLK = 0

at 2010-001-00:00:00 UTC for Europa Clipper.

Note that this was set using the value of 34 leap seconds.

Note also that this is the SCLK = 0 UTC epoch expected to

be used for Observatory SI&T as well as for Europa Clipper launch.

The sub-seconds is set to a maximum of 65,536 ticks in the

"SCLK01\_MODULI\_159" parameter. That was chosen for this SCLK

kernel because all downlink packet secondary headers will

include a 16-bit sub-seconds field with least significant bit

resolution of 2^-16 second, so a tick is assumed to represent

an interval of 2^-16 second.

The end of the first partition is currently defined as the value

in ticks of the maximum value of the spacecraft clock.

Usage

--------------------------------------------------------

This file is used by the SPICE system as follows: programs that

make use of this SCLK kernel must 'load' the kernel, normally

during program initialization. Loading the kernel associates

the data items with their names in a data structure called the

'kernel pool'. The SPICELIB routine FURNSH loads text kernel

files, such as this one, into the pool as shown below:

FORTRAN:

CALL FURNSH ( SCLK\_kernel\_name )

C:

furnsh\_c ( SCLK\_kernel\_name );

Once loaded, the SCLK time conversion routines will be able to

access the necessary data located in this kernel for their

designed purposes.

References

--------------------------------------------------------

1. "SCLK Required Reading"

Inquiries

--------------------------------------------------------

If you have any questions regarding this file or its usage,

contact:

Mike Reid

(240) 228-4537

Mike.Reid@jhuapl.edu

Kernel Data

--------------------------------------------------------

The Europa Clipper spacecraft clock is represented by the SPICE

type 1 SCLK kernel. It uses TDT, Terrestrial Dynamical Time,

as its parallel time system.

\begindata

SCLK\_KERNEL\_ID = ( @2018-07-05T20:00:00 )

SCLK\_DATA\_TYPE\_159 = ( 1 )

SCLK01\_TIME\_SYSTEM\_159 = ( 2 )

\begintext

In a particular partition of the Europa Clipper spacecraft clock,

the clock read-out consists of two separate stages:

1/18424652:24251

The first stage, an unsigned 32 bit field, represents the spacecraft

clock seconds count. The second, an unsigned 16 bit field, represents

counts of 2^-16 second increments of the spacecraft clock.

The following keywords and their values establish this structure:

\begindata

SCLK01\_N\_FIELDS\_159 = ( 2 )

SCLK01\_MODULI\_159 = ( 4294967296 65536 )

SCLK01\_OFFSETS\_159 = ( 0 0 )

SCLK01\_OUTPUT\_DELIM\_159 = ( 2 )

\begintext

This concludes the invariant portion of the SCLK kernel data.

The remaining sections of the kernel may require updates as the clock

correlation coefficients evolve in time. The first section below

establishes the clock partitions. The data in this section consists

of two parallel arrays, which denote the start and end values in ticks

of each partition of the spacecraft clock.

SPICE utilizes these two arrays to map from spacecraft clock ticks,

determined with the usual modulo arithmetic, to encoded SCLK--the

internal, monotonically increasing sequence used to tag various

data sources with spacecraft clock.

\begindata

SCLK\_PARTITION\_START\_159 = ( 0.00000000000000e+00 )

SCLK\_PARTITION\_END\_159 = ( 2.81474976710655e+14 )

\begintext

The remaining section of the SCLK kernel defines the clock correlation

coefficients. Each line contains a 'coefficient triple':

Encoded SCLK at which Rate is introduced.

Corresponding TDT Epoch at which Rate is introduced.

Rate in TDT (seconds) / most significant clock count (~seconds).

SPICE uses linear extrapolation to convert between the parallel

time scale (TDT) and encoded SCLK. The triples are stored in the array

defined below.

The first time triplet below was entered manually and represents

the approximate time (in TDT) at which SCLK = zero. Note that the

conversion from UTC to TDT used 34 leap seconds.

\begindata

SCLK01\_COEFFICIENTS\_159 = (

0 @01-JAN-2010-00:01:06.184000 1.00000000000

)

\begintext

1. External References
2. S. B. Cooper, “*From Mercury to Pluto: A common approach to timekeeping*”, IEEE Aerospace and Electronic Systems Magazine, vol. 21, issue 10, October 2006.
3. S.B. Cooper, J. R. Jensen, and G. L. Weaver, “*MESSENGER onboard timekeeping accuracy during the first year in orbit at Mercury*,” Proceedings of the 44th Annual Precise Time and Interval (PTTI) Systems and Applications Meeting, Reston, Virginia, November 2012, pp. 361-370.
4. M. R. Reid & S. B. Cooper, “*The multi-mission time correlation system*”, Proceedings of IEEE Space Mission Challenges in Information Technology (SMC-IT 2019), Caltech, Pasadena, California, July 2019.
5. Acronyms and Glossary

|  |  |
| --- | --- |
| 1 PPS | “1-pulse-per-second” signal occurring at 1 Hz rate |
| AMMOS | Advanced Multi-Mission Operations System |
| AMPCS | AMMOS Mission Data Processing and Control System |
| APL | (Johns Hopkins) Applied Physics Laboratory (also “JHU/APL”) |
| AU | Astronomical Unit, roughly 150,000,000 kilometers |
| Avionics | C&DH electronics or, sometimes, C&DH electronics and software |
| C&DH | Command and Data Handling Subsystem |
| CDS | CCSDS Day Segmented Time (days::ms of day::microsecond time code used by the DSN) |
| CLKRATE | See “clock change rate” – This can refer to the (predicted) CLKRATE used by the onboard G&C processor, to the predicted CLKRATE in the SCLK kernel, or to the interpolated CLKRATE in the SCLK kernel (measured in TDT seconds/SCLK second) |
| Clock change rate | The number of UTC or TDT seconds per SCLK second |
| DSN | NASA’s Deep Space Network |
| encoded SCLK | Continuous mission timeline in the SCLK kernel, mapped from possibly discontinuous SCLK |
| ERT | Earth Received Time, the time at which a downlink frame reference edge is received at a DSN station |
| ET | Ephemeris Time, equivalent to Barycentric Dynamical Time (TDB) |
| FGICD | Flight Ground Interface Control Document |
| G&C | Guidance and Control Subsystem |
| GDS | Ground Data System |
| GNC | Guidance, Navigation, and Control |
| I&T | Integration and Test |
| IAU | International Astronomical Union |
| int-SCLK | Integer seconds component of SCLK |
| iSCLK | Although not used in this document, this sometimes refers to an instrument clock |
| JDK | Java Development Kit |
| JHU/APL | Johns Hopkins Applied Physics Laboratory |
| JPL | Jet Propulsion Laboratory |
| JRE | Java Runtime Environment |
| Kbps | Thousands of bits (not bytes) per second |
| Mbps | Millions of bits (not bytes) per second |
| MGSS | Multi-mission Ground Systems and Support (Program Office) |
| MMTC | Multi-Mission Time Correlation |
| MOC | Mission Operations Center (also called an MSA) |
| MSA | Mission Support Area (also called a MOC) |
| NAIF | Navigation and Ancillary Information Facility (at JPL) |
| OWLT | One-way light time, used in this memo to refer to the time for a frame to travel from a spacecraft to a receiving ground station; SPICE provides this in a solar barycentric frame of reference |
| PPS | Same as 1 PPS |
| RHEL | Red Hat Enterprise Linux |
| SC ID | Spacecraft Identifier (assigned by the DSN) |
| SCLK | The spacecraft clock; typically refers to a hardware clock |
| SCLK kernel | Spacecraft clock data file containing correlations between encoded SCLK and TDT(G) |
| SPICE | “Spacecraft Planet Instrument C-Matrix Events” system of software tools developed by the JPL NAIF |
| T&C | Telemetry and Commanding |
| TDB | Terrestrial Barycentric Time (acronym is in French), an absolute time measured from the Solar System Barycenter reference frame, equivalent to ephemeris Time. For MMTC, this is the same as ET. |
| TDT | Terrestrial Dynamical Time (also called Terrestrial Time or TT), measured from an Earth reference frame |
| TDT(G) | Ground estimate of the TDT of the 1 PPS reference edge |
| TDT(S) | Onboard estimate of the TDT of the 1 PPS reference edge |
| TK | Time Correlation (“TK” so as not to be confused with Telecommand.) |
| TLM | Telemetry |
| URI | Uniform Resource Identifier |
| UTC | Coordinated Universal Time |
| v-SCLK | “Vernier” or sub-seconds component of SCLK |