

Abstract

Oregano Systems provides two Application Programming Interfaces (APIs) for the syn1588® Software Suite. This application note describes the shared memory API for the syn1588® PTP Stack allowing access to many parameters defined in IEEE 1588-2008 as well as other relevant status data. This API can be used to configure, control, and monitor the syn1588® PTP Stack and thus is ideally suited for all QoS applications.

Please note that the syn1588® shared memory API is designed and specialized in working with the syn1588® PTP Stack only. Other applications than the syn1588® PTP Stack are not supported by this API.

Introduction

There are two APIs available for the Oregano Systems' syn1588® software suite, namely:

- The “shared memory API” (sharedmem_API) for controlling, monitoring and configuring the syn1588® PTP Stack which is described in this document.
- The “syn1588® API” (syn1588_API) for accessing syn1588® hardware like the syn1588® PCIe NIC hardware. It allows to control hardware features like generating frequencies, capturing events, and configuring the packet timestamping units. For more information about the syn1588® API take a look at the respective documentation (doc folder of the syn1588_API package) or contact [Oregano Systems](#).

The shared memory API allows the user to gather status information about the syn1588® PTP Stack as well as current values of all its configuration parameters. For example, the API can be used by any 3rd party application to read the current UTC offset and synchronization status or configure the current UTC offset, time traceability and clock class depending on the GPS input to mention but a few of the possibilities. Figure 1 depicts two use cases of the shared memory communication mechanism.

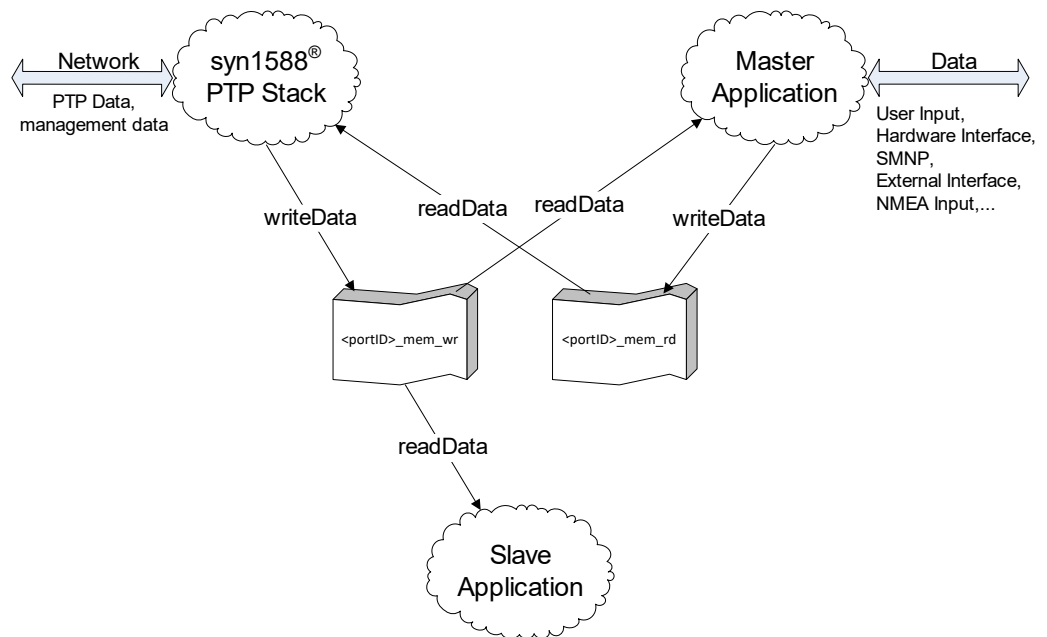


Figure 1 Shared Memory Communication

A “master” application reads data from the shared memory **<portID>_mem_wr** for monitoring purposes. After processing the information it will write back modified data to **<portID>_mem_rd** for (re-)configuration purposes. This updated data will in turn be read and applied by the **syn1588® PTP Stack**. A “slave” application, on the other hand, only reads data and processes it.

NOTE: The terms “master application” and “slave application” are used in this case only in regards to the shared memory and have no connection to a PTP Master or PTP Slave.

Shared Memory Layout

The shared memory includes the following fields

Data Field name	Code reference	Detail
syn1588 [®] PTP Context Data Field	struc Context_t	Always present (some sub-fields are optional)
syn1588 [®] PTP Synchronization Status Data Field	struc BoundaryLf_t	Always present
syn1588 [®] PTP Configuration Data Field [OPTIONAL]	struc PTP_ConfigData	Optional

The syn1588[®] PTP Context is a collection of all PTP data sets as specified by the IEEE 1588 standard including profile specific and PTP optional feature data sets. Please contact Oregano Systems' support if you require more information about the implemented optional feature data sets.

The syn1588[®] PTP synchronization (Sync.) status contains information about the current accuracy of the clock synchronization. This includes a "in-sync" flag and an index representing the current accuracy of the clock synchronization. The limit/boundary that is used as decision for the system being "in-Sync" or not (i.e., re/setting the "in_Sync" flag) currently cannot be set via the shared memory but is controlled via the command line parameter "-b" or the config file parameter "boundary".

The syn1588[®] PTP Configuration provides access to the complete configuration of the syn1588[®] PTP Stack and allows to (re-) configure the syn1588[®] PTP Stack during operation (e.g., reconfigure network interfaces for PTP ports or enable/disable access to the hardware clock). This is an optional feature which is not required for standard PTP operation. Please contact Oregano Systems' support if you require more information about this feature.

General Data Format

The data format used for the shared memory communication is specified in the source file "src/datasets.h". It contains structures of the data sets described in the IEEE 1588-2008 standard chapter 8 as well as other relevant data like profile specific parameters. Additionally, there is a status field to flag the current status of the shared memory as well as an ID field to flag which application accessed the shared memory.

Oregano Systems provides an API for simplified access the shared memory. The API is available in C++ for Windows as well as for Linux facilitating application development for controlling, configuring, and monitoring of the syn1588® PTP Stack. The include file "src/datasets.h" is written in a way that it can be used for C as well as C++ programs.

Please be aware that Oregano Systems reserves the right to update the data format at any time, e.g. with newer revisions of the syn1588® PTP Stack. It's extremely important to make sure that only compatible software versions are used for shared memory communication (e.g. only use identical versions of ptp and eSync).

syn1588® PTP Context Data Field

The Context_t data structure (see "src/datasets.h") is the main block provided by the shared memory. The following lists contain more information about the different data sets that are part of this block.

NOTE: data sets marked [OPTIONAL] may be excluded from a shared memory at compile time to reduce the size of the shared memory.

[CONTENT] Status Field

The following flags are used to identify the memory's status:

Flag	Status	Enum ContextStat_e
0x000	Empty	EEmpty
0x001	New data available (written by the PTP Stack)	ENewDataWr
0x004	New data for the PTP Stack available	ENewDataRd
0x010	Alternate time offset data is valid	EAltTimeOff
0x020	Power profile (C37.238) data is valid	EPowerData
0x040	SMPTE sync metadata is valid	ESmpteData
0x080	Telecom 2 data (G.8257.1) data is valid	ETelecom2Data
0x100	Local time zone data (C37.238 2014) is valid	ELocalTZ
0x200	Path Trace data is valid	EPathTrace
0x400	802.1 as data is valid	E802_1as
0x10000	Message counters are valid	EMessageCounter
0x20000	Unicast connections are valid	EUnicastConnections

[CONTENT] ID Field

The following IDs are used to identify access to the shared memory of different syn1588® utilities.

ID	Application	Enum ContextId_e
0x00	Res	ERes
0x01	syn1588® ptp1 (Version 1 PTP Stack)	EPtpv1
0x02	syn1588® PTP Stack	EPtpv2
0x03	syn1588® eSync	EEsync
0x04	syn1588® ISync	ELsync
0x05-0x0F	Reserved	
0x10	Meinberg	EMbg
0x11-0xFF	Third party applications	

[CONTENT] General IEEE 1588 Data sets

- **defaultDS (IEEE 1588 v2.0 chapter 8.2.1)**

Writeable: all dynamic members (PTP Boundary Clock (BC): processed by port number 1 only) which are:

- Clock Quality (clock class, clock accuracy and variance)
- Priority 1
- Priority 2
- Domain number
- Slave only flag

- **currentDS (IEEE 1588 v2.0 chapter 8.2.2)**

Writeable: n.a., this is always sourced by the syn1588® PTP stack and can only be read via this API.

- **parentDS (IEEE 1588 v2.0 chapter 8.2.3)**

Writeable: n.a., this is always sourced by the syn1588® PTP stack and can only be read via this API.

- **timePropertiesDS (IEEE 1588 v2.0 chapter 8.2.4)**

Writeable: all members (master/passive state only, PTP BC: processed only port number 1)

- **portDS (IEEE 1588 v2.0 chapter 8.2.5)**

Writeable: all configurable members (PTP BC: processed by all ports):

- Announce Interval (master/passive state and unicast slave)
- Announce receipt timeout
- Sync Interval (master/passive state and unicast slave)
- Delay mechanism
- min PDelay Request Interval
- Delay Request Interval (master/passive state and unicast slave)

[CONTENT] IEEE1588 Optional feature and PTP Profile data sets

- [OPTIONAL] **Path Trace List (IEEE 1588 v2.0 chapter 16.2)**
Writeable: n.a., this is always sourced by the syn1588® PTP stack and can only be read via this API.
- [OPTIONAL] **Alternate Time Offset Properties (IEEE 1588 v2.0 chapter 16.3)**
Writeable: all members, three instances (master/passive state only, PTP BC: processed by port number 1 only)
- [OPTIONAL] **C37.238-2011 PTP Profile Data (C37.238-2011 chapter 5.12.2)**
Writeable: power profile activated, master/passive state only, (PTP BC: processed by port number 1 only)
- [OPTIONAL] **C37.238-2017 PTP Profile Data (C37.238-2017 chapter 6.2.1)**
Writeable: power 2 profile activated, master/passive state only, (PTP BC: processed by port number 1 only)
- [OPTIONAL] **SMPTE PTP Profile sync metadata**
Writeable: SMPTE profile activated, master/passive state only, (PTP BC: processed by port number 1 only)
- [OPTIONAL] **G8275.1 PTP Profile Data (G.8275.1/Y.1369.1, Amendment 1, 08/2017, Table A.1 – A.7)**
Writeable: all members
NOTE: The standard Data sets have not been modified as specified in the profile standard, instead the new data set members have been added as separate data set
- [OPTIONAL] **G8275.2 PTP Profile Data (G8275.2/Y.1369.2, Amendment 1, 08/2017, Table A.1 – A.5)**
Writeable: all members
NOTE: SF has not been implemented
NOTE: The standard Data sets have not been modified as specified in the profile standard, instead the new data set members have been added as separate data set
- [OPTIONAL] **802.1AS PTP Profile Data (IEEE 802.1AS-2011)**
Writeable: n.a., this is always sourced by the syn1588® PTP Stack and can only be read via this API.
NOTE: not all 802.1AS data sets are available

Non-standardized data sets

- **extParentData set**
additional information about the current selected PTP master not available via the standard PTP data sets

- [OPTIONAL] **messageCounter**
This data set provides status information for send and received PTP messages
- [OPTIONAL] **unicastInfoData**
This data set provides status information for currently active PTP unicast connections (see also IEEE 1588 v2.0 chapter 16.1 and 17.5).

[ACCESS] Reading Context Data

If an application other than the syn1588® PTP Stack wants to access the shared memory, it shall do the following steps (C++ API syntax):

Open the desired shared memory identified by <portID> using

```
PTP_SharedMemoryAPI::open(AccessMode_e mode, PortId_t &id)
```

in

```
PTP_SharedMemoryAPI::ESlave mode
```

Read the data structure “Context_t” using

```
PTP_SharedMemoryAPI::readPtpContext(Context_t *c)
```

Please see “sm_example” for an example on how to read data.

[ACCESS] Writing Context Data

If an application wants to configure the syn1588® PTP Stack by writing to the shared memory, it shall do the following things (C++ API syntax). Open the desired shared memory identified by <portID> using

```
PTP_SharedMemoryAPI::open(AccessMode_e mode, PortId_t &id)
```

In “PTP_SharedMemoryAPI::EMaster “ mode. Read the data structure “Context_t” using

```
PTP_SharedMemoryAPI::readPtpContext(Context_t *c)
```

Change all desired parameters. It shall indicate the source (“ContextId_e”) and status shall indicate new data available for reading (“ContextStat_e::ENewDataRd”). Write the data structure “Context_t” using

```
PTP_SharedMemoryAPI::writePtpContext(const Context_t *c)
```

Please see the program sm_example for an example of how to write data.

syn1588[®] PTP Synchronization Status Data Field

This data field (BoundaryIf_t in “src/datasets.h”) of the shared memory can be used to:

[CONTENT] Current accuracy

NOTE: This accuracy check is an in-bound check (= self-check of the syn1588[®] PTP Stack) and its accuracy is thereby limited.

Basictypes.h →Boundaries_e	corresponding accuracy
EOver10s	Over 10 [s]
EWithin10s	Under 10 [s]
EWithin1s	Under 1 [s]
EWithin250ms	Under 250 [ms]
EWithin100ms	Under 100 [ms]
EWithin25ms	Under 25 [ms]
EWithin10ms	Under 10 [ms]
EWithin2_5ms	Under 2,5 [ms]
EWithin1ms	Under 1 [ms]
EWithin250us	Under 250 [us]
EWithin100us	Under 100 [us]
EWithin25us	Under 25 [us]
EWithin10us	Under 10 [us]
EWithin2_5us	Under 2,5 [us]
EWithin1us	Under 1 [us]
EWithin250ns	Under 250 [ns]
EWithin100ns	Under 100 [ns]
EWithin25ns	Under 2 [ns]
EWithin10ns	Under 10 [ns]

[CONTENT] Current state

The inSync flag, gives information if the clock is synchronized within a user defined boundary. This boundary can be set via the command line or config file when starting the PTP stack.

[ACCESS] Reading Synchronization Status Data

If an application other than the syn1588® PTP Stack wants to access the shared memory, it shall do the following steps (C++ API syntax):

Open the desired shared memory identified by <portID> using

```
PTP_SharedMemoryAPI::open(AccessMode_e mode, PortId_t &id)
```

in “PTP_SharedMemoryAPI::ESlave” mode. Read the data structure “BoundaryIf_t” using

```
PTP_SharedMemoryAPI::readSyncBoundary(BoundaryIf_t *b)
```

Please see “sm_example” for an example on how to read data.

syn1588® PTP Configuration Data Field [OPTIONAL]

This data set is used to apply non-standardized re-configuration of central structures of an active syn1588® PTP Stack which may include switching network interfaces or enabling/disabling of control of the underlying hardware clock.

This data set can be added to the shared memory if a customer application requires it. Please contact Oregano Systems ‘ support if you require more information about this data field.

syn1588® PTP Stack Operation

This section describes how the syn1588® PTP Stack handles the shared memory internally. The syn1588® PTP Stack will attempt to access the shared memory when one of the following events occurs:

- Heartbeat timeout (once every second)
- Announce Timeout (depends on the configuration, typically every few seconds, with a short announce interval this may be lower)
- PTP management “Set” message is received

During every access, the syn1588® PTP Stack performs the following actions:

1. Lock access to the shared memory (<portID>_mem_rd)
2. Read data (Context_t) from the shared memory (<portID>_mem_rd)
3. Evaluate the data read from the shared memory (Context_t) and apply it to the syn1588® PTP Stack
4. Write the syn1588® PTP Stack internal data as “Context_t” structure to the shared memory (<portID>_mem_wr)
5. Unlock access to the shared memory (<portID>_mem_rd)

When using a boundary clock (BC), all ports of the BC write data to the respective shared memory (<portID>_mem_rd). However, most data sets are only read and applied by the port with port number 1.

NOTE: The syn1588® PTP Stack will only apply a sub set of data sets to its run-time configuration and will apply validity checks before doing so.

Shared Memory API usage

Prerequisites

To start using the shared memory API, the following steps have to be executed:

- Add the library to your build. Please refer to the manual of your build environment to get more information of how to add an external project to your project.
- Include the header file “sharedmem_api.h” to use the functions which allows you to access the shared memory.

Please see the example “sm_example” for an example of how to use the API.

Compiling

Oregano Systems provides C++ header- and source-files for the shared memory library and an example (“sm_example”). These sources are available as CMake build flow. Refer to AN037 for more information.

As described in the chapter [CONTENT] IEEE1588 Optional feature and PTP Profile data sets the PTP Stack can provide optional datasets via the shared memory API.

The presence or absence of these datasets changes the shared memory size and layout and has to be taken into account when building an application that utilizes the shared memory API.

Compiling for a pre-compiled PTP stack

Use the shared memory API sources for the same release version as the pre-compiled PTP Stack.

Compiling for a self-build PTP Stack

If you are compiling the PTP Stack from source you have to apply the same compile-configuration to the PTP Stack as to the application utilizing the shared memory API.

Use the shared memory API sources for the same release version as the pre-compiled PTP Stack and refer to AN037 chapter “Building the syn1588® PTP Stack” for more information.

Example

To run the example make sure that the syn1588® PTP Stack is running. Then change to the directory “bin” and enter the following command

```
./sm_example <clkId> <port>
```

for Linux or

```
sm_example.exe <clkId> <port>
```

for Windows. “<clockId>” (the PTP clockId of the appropriate syn1588® device) and “<port>” (the PTP port of the selected syn1588® device, usually 1) specify which shared memory will be opened.

Remarks

An application using the shared memory to modify the PTP Stack has to use the shared memory in „PTP_SharedMemoryAPI::EMaster” mode. It has to perform the following sequence for accessing the shared memory:

1. Call “readPtpContext()” to get the current PTP Context_t and lock access to the shared memory
2. Modify the PTP Context_t
3. Call “writePtpContext()” to write the modified PTP Context_t back to the shared memory and unlock access to the shared memory.


This sequence will assure a consistent (e.g., racing condition free) access to the shared memory. It is important to reduce the time between locking and unlocking the shared memory (point 1. And 3.) to a minimum.

Please be aware that an application intending to configure the syn1588® PTP Stack should check for new data periodically, using a similar interval as the heartbeat timeout (one second) of the syn1588® PTP Stack

The shared memory will always be opened by an utility, even if there is no syn1588® PTP Stack running. The shared memory will stay open as long as a syn1588® utility and/or the syn1588® PTP Stack is connected to it.

The syn1588® PTP Stack only applies a limited set of range checking when data is read and applied from the shared memory. A master application is responsible for guaranteeing data integrity. If parameters are not within the range according to the appropriate specification, the syn1588® PTP Stack's behavior may be undefined.

Please note that the shared memory API is not designed to analyze or monitor other functionalities than the syn1588® PTP Stack; e.g. do not use it for monitoring or controlling eSync or ISync.

 <p>Franzosengraben 8 A-1030 Vienna Austria http://oregano.at contact@oregano.at</p>	<p>Copyright © 2024 Oregano Systems – Design & Consulting GmbH ALL RIGHTS RESERVED.</p> <p>Oregano Systems does not assume any liability arising out of the application or use of any product described or shown herein nor does it convey any license under its patents, copyrights, or any rights of others.</p> <p>Licenses or any other rights such as, but not limited to, patents, utility models, trademarks or tradenames, are neither granted nor conveyed by this document, nor does this document constitute any obligation of the disclosing party to grant or convey such rights to the receiving party.</p> <p>Oregano Systems reserves the right to make changes, at any time without notice, in order to improve reliability, function or design. Oregano Systems will not assume responsibility for the use of any circuitry described herein.</p> <p>All trademarks used in this document are the property of their respective owners.</p>
---	--