Application Note AN016

Porting the syn1588® PTP Stack

Version 1.14.1 - January 2022

Overview

This application note provides an overview of the steps necessary to port the syn1588® PTP Stack. The syn1588® PTP Stack comprises two major parts: The "syn1588® PTP Library" and the "syn1588® PTP Stack Application". The former implements the Precision Time Protocol as described in IEEE 1588-2008 and 1588-2019 (plus certain PTP profiles defined in other standards, refer to the application note "**an006_ptp_profiles**") while the latter adds code to be able to run on a certain platform. syn1588® PTP Stack Application can be ported either completely as application to a POSIX-compliant operating system supporting or as a library to be used within other applications on any target platform. The syn1588® Library has already been ported to a large number of target platforms:

Intel x86 and x86-64 using Linux or Windows as operating system (included with every source code license of the syn1588® PTP Stack)

Intel MC8051 (available on request, please contact Oregano Systems for further details)

Altera Nios II (available on request, please contact Oregano Systems for further details)

Altera SoC (ARM Cortex-A9, available on request, please contact Oregano Systems for further details)

Other platforms: Nvidia Tegra 2 (ARM Cortex-A9), ARM Cortex-M3, MIPS, etc.

The entire stack is implemented in C++ with focus on standard libraries and C++ 2011. As a consequence, reusing the library or porting it to new platforms is easy and straight forward.

Figure 1 shows an overview of the syn1588® PTP Stack, the PTP Library, and its interfaces. Five distinct interfaces link the syn1588® PTP Library to the syn1588® PTP Stack application. All interfaces are declared as virtual classes in a C++ header file. To implement an interface one has to write a new C++ class inheriting from the interface class and thus implementing all required functions.

The application note "**an037_cmake_syn1588_ptpstack**" describes the CMake build environment for the standard PTP stack sources.

The five interfaces are:

- Configuration
- I/O
- Logging
- Network
- Clock

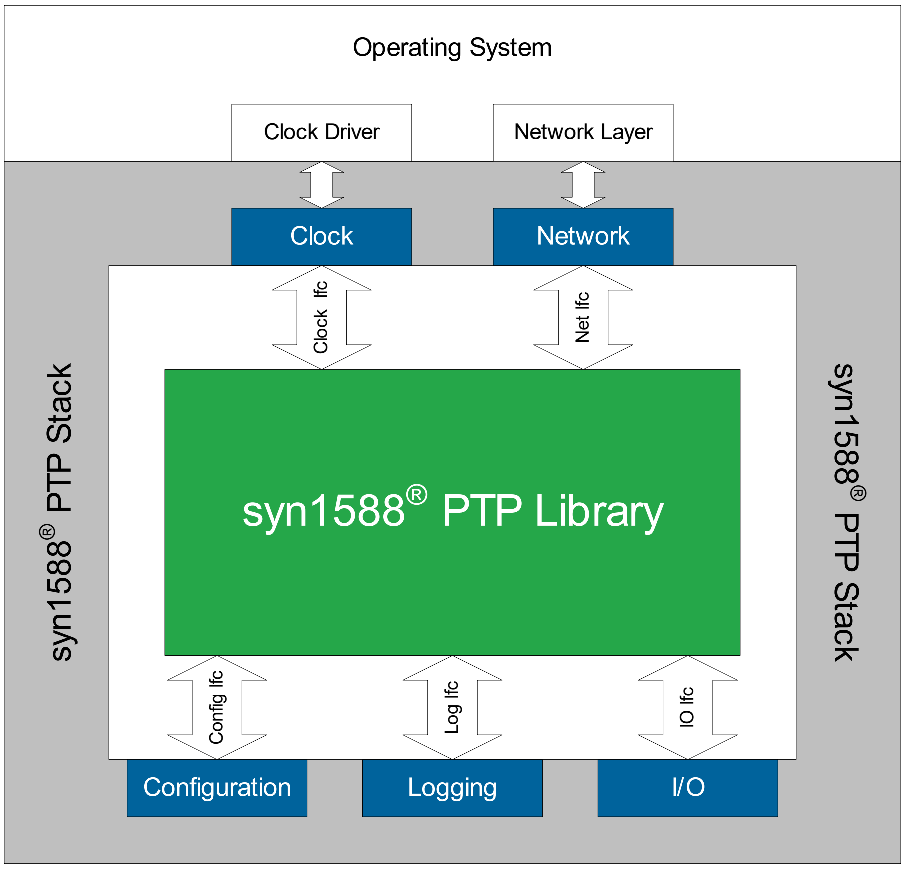

Figure 1 Overview of the syn1588® PTP Library, the syn1588® PTP Stack Application and their interfaces

The configuration interface is used to set, save, and load parameters of the syn1588® PTP Library. This includes items like the sync interval, network modes, or the log level. This set of parameters is used for initial configuration at start-up as well as for updating internal values at certain events like state changes. If parameters are changed during execution using PTP management messages, the configuration will be changed accordingly by the syn1588® PTP Stack Library. However, it is not recommended to modify configuration parameters during execution other than via the syn1588® PTP Stack Application. The I/O interface should be used instead.

The I/O interface is used for simple text I/O as well as for data I/O. The implementation of the latter is optional but may be desired for controlling, monitoring, and configuring of the syn1588® PTP Stack Library during execution. Important data structures are regularly written and read from/to the I/O interface and can be processed directly in the interface or forwarded to other applications (e.g. using shared memory or socket communication). Please refer to the application note about the Shared Memory API ("**an015_shared_memory_api**") for more details (contact Oregano Systems: contact@oregano.at).

The logging interface processes all status output of the syn1588® PTP Stack Library. It partitions logging messages into several severity levels. Therefore, it is very simple to redirect logging output to a convenient destination for further processing or analysis.

The network interface handles all network related communication of the syn1588® PTP Stack Library. It acts as a link to the underlying network infrastructure contained e.g. in an operating system or a lightweight IP stack. It transmits and receives pure PTP packets without any transport dependent headers. If an operating system is used for handling the transport, all necessary network configuration has to be performed in the inheriting class. Depending on the desired transport mechanism (e.g. IPv4, IPv6, or Layer2 Ethernet) different implementations may be required. This interface is also responsible for retrieving timestamps for sent and received PTP packets and providing them to the central PTP library. For example, the Linux network layer provides timestamps attached to the received packets (or provided via a loopback mechanism for transmit packets) via the SO_TIMESTAMPING mechanism.

The clock interface, finally, connects the syn1588® PTP Stack Library to a real time clock. This interface handles the time stamping functionality of the syn1588® PTP Stack Library as well as all clock related actions like rate adjustment, state correction, and reading the current time. It can be used to connect and control either a regular system clock (low to medium accuracy) or a high-precision hardware clock like the syn1588® Clock IP Core to the syn1588® PTP Stack Library. The clock interface handles all tasks required for adjusting the local clock. Oregano Systems provides a ready-to-use clock interface (located in `src/clock`) comprising various linear and non-linear filter structures as well as a sophisticated clock-servo to minimize the offset to the PTP master. Furthermore, the clock interface contains a hardware clock interface class providing functions for low level hardware access like reading, writing, and adjusting the clock. Porting the syn1588® PTP Stack Library to a new target clock hardware platform merely requires adjustments in the clock hardware class. However, customers are free to implement custom specific servo and filter structures by simply exchanging or modifying the respective functions in the syn1588® clock class.

For all syn1588® hardware products like the syn1588® PCIe NIC or a syn1588® Clock IP Core Oregano Systems provides an API for simplified hardware access.

Porting the syn1588® PTP Library

When merely porting the syn1588® PTP Stack Library to a new platform the five aforementioned interfaces have to be customized accordingly. The interface implementation of the syn1588® PTP Stack Application may be used as a starting point or as guidance.

Interface	Typical Design Effort [h]
Config	5
I/O	5
Logging	20
Network	50
Clock	40
Total	125

Table 1 Porting effort for syn1588® PTP Stack Library

Table 1 shows estimated efforts for porting of each interface onto a new platform depending on the desired functionality. The configuration interface is straight forward to port, because the initialization data could be defined either statically or simply read from a configuration file using a simple file parser. In more elaborate implementations the configuration class may be used for managing parameters in a non-volatile memory. On start-up, parameters may be read from the memory to be used by the syn1588® PTP Library. During operation, parameters may be written to the memory to be reused after rebooting.

The logging interface was re-implemented in release v1.14 to use libfmt as basic formatting library. This has the implication of a bigger image size of the built PTP Stack itself. If the PTP Stack is to be ported to a resource constrained environment it might be of interest to remove this dependency. With v1.14 most of the internal log lines still use printf syntax and may be re-used for different formatting libraries without changes, but with newer releases these will be replaced by the new libfmt syntax. Hence, removing libfmt will include the necessity to refactor the log lines throughout the code and will add more design effort for the logging implementation.

Depending on the target system porting effort for the network stack could vary. If the target platform has a BSD-Style socket interface porting is straightforward using the syn1588® PTP Stack Application's network implementation as a starting point. If the target platform does not support such sockets implementing the network interface may require considerable more effort.

The time for implementing the clock interface depends on how much functionality is required. For pure software time stamping only functions for reading and manipulating current time and rate of the clock source have to be implemented. For hardware time stamping additional functions for accessing the hardware timestamp registers have to be implemented or adjusted. If a custom specific servo is required as well, the effort may increase significantly.

To help with porting of the syn1588® PTP Stack Library, a detailed documentation of all five interfaces is available upon request.

Porting the syn1588® PTP Stack Application

When porting the syn1588® PTP Stack Application to a new platform running a POSIX compliant operating system (e.g. Linux or Windows), porting effort should be minimal. Typically, only the network and the clock interface (especially when using custom PTP hardware) have to be adapted. Oregano Systems has already ported all its syn1588® software products to the following target platforms:

- Windows (XP and later, Server 2003 and later, x32 and x64)
- Linux (2.6.14 to 5.0)
- FreeBSD (7.x)

Interface	Typical Design Effort [h]
Network	10
Clock	20
Total	30

Table 2 Porting effort for syn1588® PTP Stack Application

Table 2 shows estimated efforts required for porting the syn1588® PTP Stack Application to a new target platform. However, depending on the target platform, it may still be necessary to adapt other interfaces as well.

By purchasing a syn1588® PTP Stack source code license, a customer qualifies for receiving the complete source code as well as all required project files to build the syn1588® PTP Stack Application on Windows and on Linux (please see release notes **syn1588_release_OS_support** for more details about supported operating systems).

Using a Custom Clock

Integrating support for a custom (hardware) clock into the syn1588® PTP Stack Application on a Linux or Windows based environment can be achieved by implementing a single C++ class (derived from `/src/clock/hwclock_ifc.h`). The class is used as interface between the application's clock class and an arbitrary (hardware) clock implementation. As a starting point, it is sufficient to implement the setTime(), getTime(), getTimestamp(), and adjustRate() functions.

Further Information

In previous versions of this document the description of the build environment was part of this document. AS the build environment changed this description was moved to a separate Application Note "**an037_cmake_syn1588_ptpstack**".

Oregano Systems is always happy to support its customers. In case you need assistance while porting the syn1588® PTP Stack or if you have any other questions, please contact us: support@oregano.at.

Oregano Systems A Meinberg Company Franzosengraben 8 A-1030 Vienna Austria http://oregano.at contact@oregano.at	