Time Service Facility FPGA PSM Specification

Document WINNF-TS-3004-App03

Version V1.1.1

18 January 2022

TERMS, CONDITIONS & NOTICES

This document has been prepared by the Software Defined Systems (SDS) Harmonized Timing Service Task Group to assist The Software Defined Radio Forum Inc. (or its successors or assigns, hereafter “the Forum”). It may be amended or withdrawn at a later time and it is not binding on any member of the Forum or of the Harmonized Timing Service Task Group.

Contributors to this document that have submitted copyrighted materials (the Submission) to the Forum for use in this document retain copyright ownership of their original work, while at the same time granting the Forum a non-exclusive, irrevocable, worldwide, perpetual, royalty-free license under the Submitter’s copyrights in the Submission to reproduce, distribute, publish, display, perform, and create derivative works of the Submission based on that original work for the purpose of developing this document under the Forum's own copyright.

Permission is granted to the Forum’s participants to copy any portion of this document for legitimate purposes of the Forum. Copying for monetary gain or for other non-Forum related purposes is prohibited.

THIS DOCUMENT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS SPECIFICATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE FORUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS DOCUMENT.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the specification set forth in this document, and to provide supporting documentation.

This document was developed following the Forum's policy on restricted or controlled information (Policy 009) to ensure that that the document can be shared openly with other member organizations around the world. Additional Information on this policy can be found here: http://www.wirelessinnovation.org/page/Policies_and_Procedures

Although this document contains no restricted or controlled information, the specific implementation of concepts contain herein may be controlled under the laws of the country of origin for that implementation. Readers are encouraged, therefore, to consult with a cognizant authority prior to any further development.

Wireless Innovation Forum ™ and SDR Forum ™ are trademarks of the Software Defined Radio Forum Inc.

Table of Contents

TERMS, CONDITIONS & NOTICES i
Table of Contents ii
List of Figures iii
List of Tables iii
Contributors iv
Time Service Facility FPGA PSM specification 1
1 Introduction 1
1.1 Reference definitions 1
1.2 Conformance 2
1.2.1 Radio platform items 2
1.2.2 Radio application items 2
1.3 Document structure 2
2 FPGA functional interfaces 3
2.1 Specification approach 3
2.1.1 Base RTL signals 3
2.1.2 Primitive prefixes 3
2.2 Interfaces specification 4
2.2.1 TimeService::TerminalTime::TerminalTimeAccess 4
2.2.2 TimeService::SystemTime::SystemTimeAccess 5
2.2.3 TimeService::SystemTime::StandardTimeProvision 9
2.2.4 TimeService::StandardTimes::ReferencesNotification 11
2.2.5 TimeService::SpecificTimes::SpecificTimeHandling 12
2.2.6 TimeService::SpecificTimes::SettingsNotification 14
3 FPGA PSM constants 16
3.1 PIM version 16
4 VHDL programming 17
4.1 VHDL library 17
4.2 pkg_tsf_api_types.vhd 18
4.3 pkg_tsf_primitives_parameters.vhd 20
5 References 23
5.1 Referenced documents 23
END OF THE DOCUMENT 24

List of Figures

Figure 1 Positioning of FPGA PSM interfaces 1
Figure 2 Dynamic behavior for *getTerminalTime()* 4
Figure 3 Dynamic behavior for *getTerminalTimeRateUncertainty()* 5
Figure 4 Dynamic behavior for *getCurrentTAI()* 6
Figure 5 Dynamic behavior for *getCurrentUTC()* 7
Figure 6 Dynamic behavior for *getLastUpdateTAI()* 8
Figure 7 Dynamic behavior for *getLastUpdateUTC()* 9
Figure 8 Dynamic behavior for *provideTAI()* 10
Figure 9 Dynamic behavior for *provideUTC()* 11
Figure 10 Dynamic behavior for *notifyStandardTimeReference()* 12
Figure 11 Dynamic behavior for *setSpecificTime()* 13
Figure 12 Dynamic behavior for *getSpecificTime()* 14
Figure 13 Dynamic behavior for *notifySpecificTimeSetting()* 15

List of Tables

Table 1 Definitions from *Time Service Facility PIM Specification* 1
Table 2 Definitions from *Principles for WInnForum Facility Standards* 2
Table 3 Definitions from *WInnForum Facilities PSMs Mapping Rules* 2
Table 4 Base RTL signals from *WInnForum Facilities PSMs Mapping Rules* 3
Table 5 RTL signals for *getTerminalTime()* 4
Table 6 RTL signals for *getTerminalTimeRateUncertainty()* 4
Table 7 RTL signals for *getCurrentTAI()* 5
Table 8 RTL signals for *getCurrentUTC()* 6
Table 9 RTL signals for *getLastUpdateTAI()* 7
Table 10 RTL signals for *getLastUpdateUTC()* 8
Table 11 RTL signals for *provideTAI()* 9
Table 12 RTL signals for *provideUTC()* 10
Table 13 RTL signals for *notifyStandardTimeReference()* 11
Table 14 RTL signals for *setSpecificTime()* 12
Table 15 RTL signals for *getSpecificTime()* 13
Table 16 RTL signals for *notifySpecificTimeSetting()* 14

Contributors

The following individuals and their organization of affiliation are credited as Contributors to development of the specification, for having been involved in the work group that developed the draft then approved by WInnForum member organizations:

- Marc Adrat, Fraunhofer FKIE,
- Guillaume Delbarre, DGA,
- David Hagood, Cynosure,
- Olivier Kirsch, KEREVAL,
- Frederic Le Roy, ENSTA Bretagne,
- David Murotake, HiKE,
- Eric Nicollet, Thales,
- Kevin Richardson, MITRE.

Time Service Facility FPGA PSM specification

1 Introduction

This document WINNF-TS-3004-App03 is the *FPGA PSM specification* of *WInnForum Time Service Facility* V1.1.0.

It derives from *Time Service Facility PIM Specification* [Ref1] in accordance with *Principles for WInnForum Facility Standards* [Ref2].

It addresses the FPGA programming paradigm, applying the mapping rules of the FPGA section of *WInnForum Facilities PSMs Mapping Rules* [Ref3] and specifically reporting any deviation to those rules.

The following figure positions the interfaces addressed by the *FPGA PSM specification*:

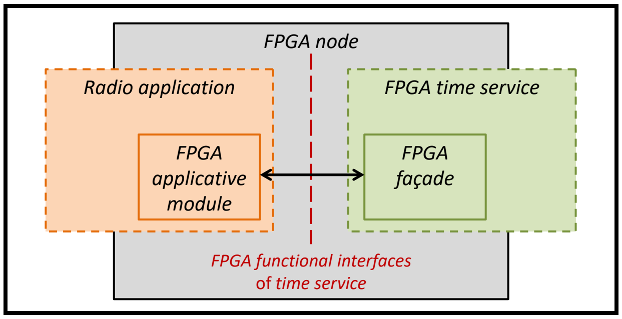

Figure 1 Positioning of FPGA PSM interfaces

As depicted, the *FPGA PSM specification* addresses the *FPGA functional interfaces* of *time services*, positioned, within an *FPGA node*, between the *FPGA applicative module*s of *radio applications* and *FPGA façades* of *time service* instances.

The *FPGA PSM specification* considers RTL (Register-Transfer Level) [Ref4] interfaces, specifying RTL signals and the associated chronogram, independently from the used programming language (e.g., VHDL or Verilog).

It also provides normative source files applicable in case of VHDL programming.

1.1 Reference definitions

The *Time Service Facility FPGA PSM specification* applies the following definitions from *Time Service Facility PIM Specification* [Ref1]:

Topic	Used definitions
Time service concepts	*time service, Time Service Facility*

Table 1 Definitions from *Time Service Facility PIM Specification*

The *Time Service Facility FPGA PSM specification* applies the following definitions from *Principles for WInnForum Facility Standards* [Ref2]:

Topic	Used definitions
Base concepts	*radio application*
Architecture concepts	*façade*
WInnForum facilities	*facility, PIM specification*, *PSM specification*
Primitives	*primitive*, *parameter*, *exception*, *type*

Table 2 Definitions from *Principles for WInnForum Facility Standards*

The *Time Service Facility FPGA PSM specification* applies the following definitions from *WInnForum Facilities PSMs Mapping Rules* [Ref3]:

Topic	Used definitions
Specification purpose	*FPGA PSM specification*, *FPGA functional interfaces*
Software architecture	*FPGA node*, *FPGA façade*, *FPGA applicative module*
RTL signals origin	*origin, caller, callee*
Base RTL signals	*primitive prefix, structural RTL signals*, *semantics RTL signals*, *parameters RTL signals*, *exception RTL signals*

Table 3 Definitions from *WInnForum Facilities PSMs Mapping Rules*

The term “*unspecified”* indicates an aspect explicitly left to implementer’s decisions.

1.2 Conformance

1.2.1 Radio platform items

An *FPGA façade* of a *time service* implementation **is conformant with** the *Time Service Facility FPGA PSM specification* if it provides an FPGA implementation of related *primitives*.

An *FPGA time service* **is defined as** a *time service* implementation with all of its *FPGA façades* being conformant with the *FPGA PSM specification*.

1.2.2 Radio application items

An *FPGA applicative module* of a *radio application* **is conformant with** the *Time Service Facility FPGA PSM specification* if it can use *FPGA façades* conformant with the *FPGA PSM specification* without using any non-standard *primitive* for the *time service*.

1.3 Document structure

Section 2 specifies the normative content for the *FPGA functional interfaces*.

Section 3 specifies the normative content for FPGA constants.

Section 4 specifies the normative files to be used for VHDL programming.

2 FPGA functional interfaces

This normative section specifies the *FPGA functional interfaces* for *time service*, according to the FPGA section of *WInnForum Facilities PSMs Mapping Rules* [Ref3].

2.1 Specification approach

2.1.1 Base RTL signals

The following base RTL signals from the FPGA section of *WInnForum Facilities PSMs Mapping Rules* [Ref3] are used:

Base RTL signal	Used definitions
Structural RTL signals	**CLK**, **RST**
Semantics RTL signals	**EN**, **RDY**
Parameters RTL signals	**EN_IN**, **DATA_IN**, **EN_OUT**, **DATA_OUT**
Exception RTL signals	**IRQ**

Table 4 Base RTL signals from *WInnForum Facilities PSMs Mapping Rules*

For each base RTL signal, the complete RTL signal name, its *origin* (using the concept of *caller* and *callee*) and its format are specified.

2.1.2 Primitive prefixes

The *primitive prefixes* for *time service* follow the related indications of *WInnForum Facilities PSMs Mapping Rules* [Ref3].

A *primitive prefix* concatenates:

- The **TSF_** field, for *time service*,
- The **<instNum>_** field, optionally numbering instances of a *time service* in case there are more than one (starting count from 1),
- The **<PRIM_NAME>_** field, identifying the *primitive* using a screaming snake case transcription of the *PIM specification* name.

2.2 Interfaces specification

2.2.1 *TimeService::TerminalTime::TerminalTimeAccess*

2.2.1.1 getTerminalTime()

The RTL signals for *getTerminalTime()* **are specified by** the following table:

RTL signal name TSF_{<instNum>_}GET_TERMINAL_TIME_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
EN_OUT	*FPGA app module*	1-bit signal
DATA_OUT.terminal_time	*FPGA app module*	Two 32-bit vectors

Table 5 RTL signals for *getTerminalTime()*

The dynamic behavior for *getTerminalTime()* **is specified by** the following chronogram:

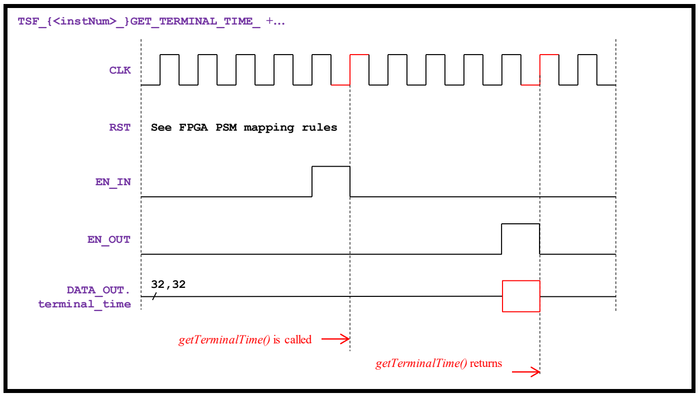
TSF_{<instNum>_}GET_TERMINAL_TIME_ +...
CLK
RST
See FPGA PSM mapping rules
EN_IN
EN_OUT
DATA_OUT.
terminal_time
32,32
getTerminalTime() is called
getTerminalTime() returns

Figure 2 Dynamic behavior for *getTerminalTime()*

2.2.1.2 getTerminalTimeRateUncertainty()

The RTL signals for *getTerminalTimeRateUncertainty()* **are specified by** the following table:

RTL signal name TSF_{<instNum>_}GET_TERMINAL_TIME_RATE_UNCERTAINTY_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
EN_OUT	*FPGA app module*	1-bit signal
DATA_OUT.terminal_time_rate_uncertainty	*FPGA app module*	32-bit vector -- signed

Table 6 RTL signals for *getTerminalTimeRateUncertainty()*

The dynamic behavior for *getTerminalTimeRateUncertainty()* **is specified by** the following chronogram:

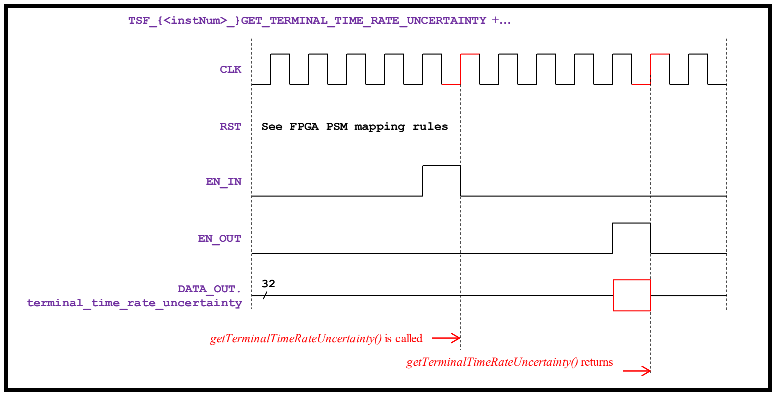
TSF_{<instNum>_}GET_TERMINAL_TIME_RATE_UNCERTAINTY +...
CLK
RST
See FPGA PSM mapping rules
EN_IN
EN_OUT
DATA_OUT.
terminal_time_rate_uncertainty
32
getTerminalTimeRateUncertainty() is called
getTerminalTimeRateUncertainty() returns

Figure 3 Dynamic behavior for *getTerminalTimeRateUncertainty()*

2.2.2 TimeService::SystemTime::SystemTimeAccess

2.2.2.1 getCurrentTAI()

The RTL signals for *getCurrentTAI()* **are specified by** the following table:

RTL signal name `TSF_{<instNum>_}GET_CURRENT_TAI_ +`	**Origin**	**Format**
`CLK`	*FPGA façade*	1-bit signal
`RST`	*FPGA façade*	1-bit signal
`EN_IN`	*FPGA façade*	1-bit signal
`EN_OUT`	*FPGA app module*	1-bit signal
`DATA_OUT.current_tai`	*FPGA app module*	Two 32-bit vectors
`DATA_OUT.time_stamp`	*FPGA app module*	Two 32-bit vectors
`DATA_OUT.time_uncertainty`	*FPGA app module*	Two 32-bit vectors

Table 7 RTL signals for *getCurrentTAI()*

The dynamic behavior for *getCurrentTAI()* **is specified by** the following chronogram:

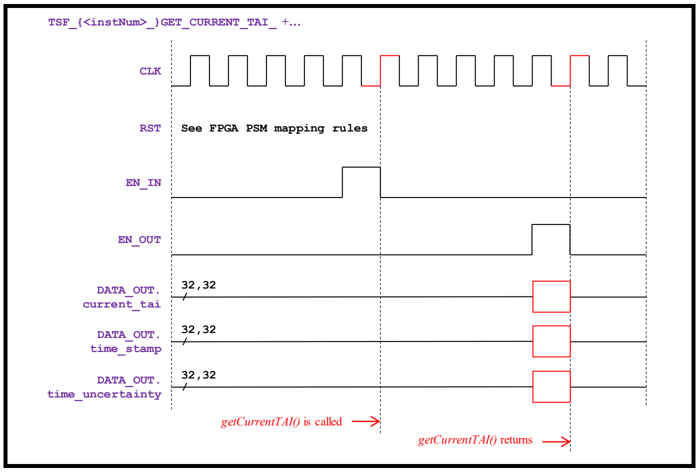

Figure 4 Dynamic behavior for *getCurrentTAI()*

2.2.2.2 getCurrentUTC()

The RTL signal*s* for *getCurrentUTC()* **are specified by** the following table:

RTL signal name TSF_{<instNum>_}GET_CURRENT_UTC_ +	**Origin**	**Format**
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
EN_OUT	*FPGA app module*	1-bit signal
DATA_OUT.current_utc	*FPGA app module*	Two 32-bit vectors
DATA_OUT.time_stamp	*FPGA app module*	Two 32-bit vectors
DATA_OUT.time_uncertainty	*FPGA app module*	Two 32-bit vectors

Table 8 RTL signals for *getCurrentUTC()*

The dynamic behavior for *getCurrentUTC()* **is specified by** the following chronogram:

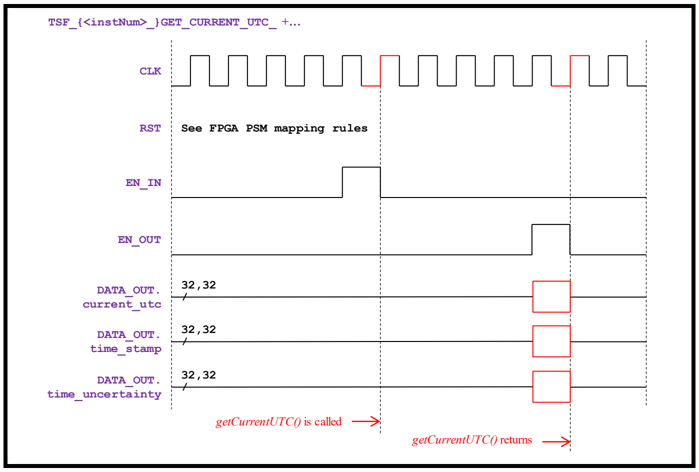

Figure 5 Dynamic behavior for *getCurrentUTC()*

2.2.2.3 getLastUpdateTAI()

The RTL signals for *getLastUpdateTAI()* **are specified by** the following table:

RTL signal name TSF_{<instNum>_}GET_LAST_UPDATE_TAI_ +	**Origin**	**Format**
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
EN_OUT	*FPGA app module*	1-bit signal
DATA_OUT.last_update_tai	*FPGA app module*	Two 32-bit vectors
DATA_OUT.time_stamp	*FPGA app module*	Two 32-bit vectors
DATA_OUT.time_uncertainty	*FPGA app module*	Two 32-bit vectors

Table 9 RTL signals for *getLastUpdateTAI()*

The dynamic behavior for *getLastUpdateTAI()* **is specified by** the following chronogram:

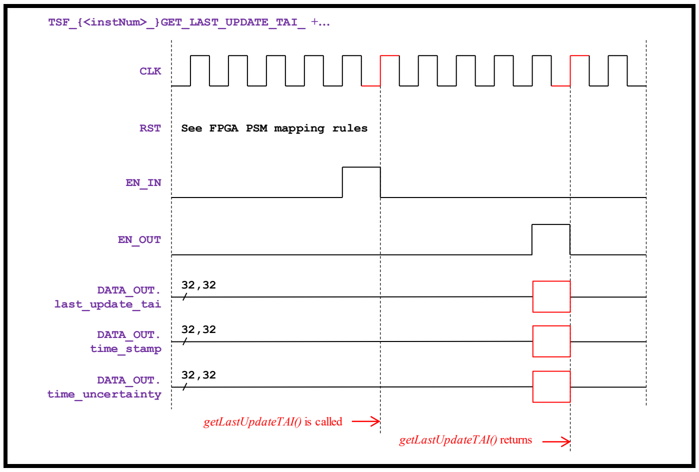
TSF_{<instNum>_}GET_LAST_UPDATE_TAI_ +...
CLK
RST
See FPGA PSM mapping rules
EN_IN
EN_OUT
DATA_OUT.
last_update_tai
32,32
DATA_OUT.
time_stamp
32,32
DATA_OUT.
time_uncertainty
32,32
getLastUpdateTAI() is called
getLastUpdateTAI() returns

Figure 6 Dynamic behavior for *getLastUpdateTAI()*

2.2.2.4 getLastUpdateUTC()

The RTL signals for *getLastUpdateUTC()* **are specified by** the following table:

RTL signal name TSF_{<instNum>_}GET_LAST_UPDATE_UTC_ +	**Origin**	**Format**
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
EN_OUT	*FPGA app module*	1-bit signal
DATA_OUT.last_update_utc	*FPGA app module*	Two 32-bit vectors
DATA_OUT.time_stamp	*FPGA app module*	Two 32-bit vectors
DATA_OUT.time_uncertainty	*FPGA app module*	Two 32-bit vectors

Table 10 RTL signals for *getLastUpdateUTC()*

The dynamic behavior for *getLastUpdateUTC()* **is specified by** the following chronogram:

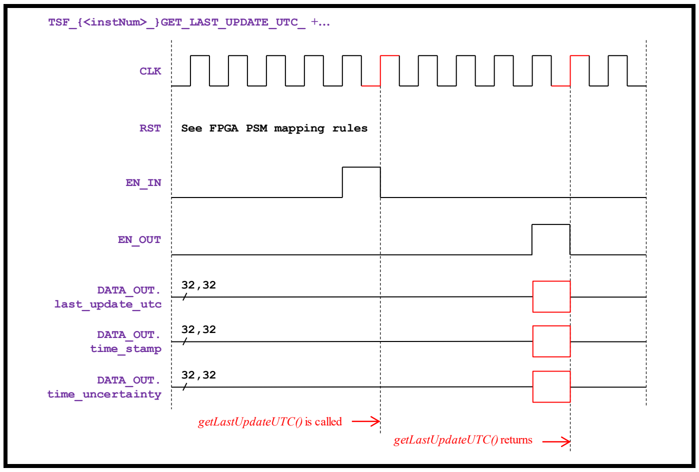
TSF_{<instNum>_}GET_LAST_UPDATE_UTC_ +...
CLK
RST
See FPGA PSM mapping rules
EN_IN
EN_OUT
DATA_OUT.
last_update_utc
32,32
DATA_OUT.
time_stamp
32,32
DATA_OUT.
time_uncertainty
32,32
getLastUpdateUTC() is called
getLastUpdateUTC() returns

Figure 7 Dynamic behavior for *getLastUpdateUTC()*

2.2.3 TimeService::SystemTime::StandardTimeProvision

2.2.3.1 provideTAI()

The RTL signals for *provideTAI()* **are specified by** the following table:

RTL signal name TSF_{<instNum>_}PROVIDE_TAI_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
DATA_IN.provided_tai	*FPGA app module*	Two 32-bit vectors
DATA_IN.time_stamp	*FPGA app module*	Two 32-bit vectors
DATA_IN.time_uncertainty	*FPGA app module*	Two 32-bit vectors
DATA_IN.source_id	*FPGA app module*	8-bit vector
IRQ_FUTURE_TIME_STAMP (optional)	*FPGA app module*	1-bit signal

Table 11 RTL signals for *provideTAI()*

The dynamic behavior for *provideTAI()* **is specified by** the following chronogram:

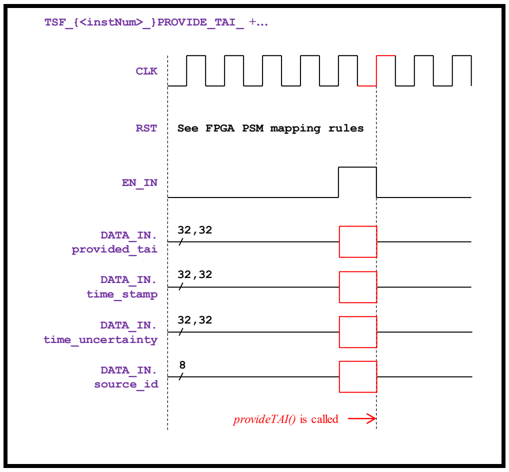

Figure 8 Dynamic behavior for *provideTAI()*

2.2.3.2 provideUTC()

The RTL signals for *provideUTC()* **are specified by** the following table:

RTL signal name TSF_{<instNum>_}PROVIDE_UTC_ +	**Origin**	**Format**
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
DATA_IN.provided_utc	*FPGA app module*	Two 32-bit vectors
DATA_IN.time_stamp	*FPGA app module*	Two 32-bit vectors
DATA_IN.time_uncertainty	*FPGA app module*	Two 32-bit vectors
DATA_IN.source_id	*FPGA app module*	8-bit vector
IRQ_FUTURE_TIME_STAMP (optional)	*FPGA app module*	1-bit signal

Table 12 RTL signals for *provideUTC()*

The dynamic behavior for *provideUTC()* **is specified by** the following chronogram:

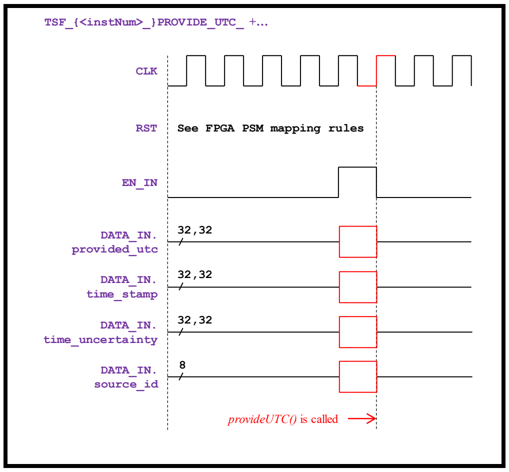

Figure 9 Dynamic behavior for *provideUTC()*

2.2.4 *TimeService::StandardTimes::ReferencesNotification*

2.2.4.1 notifyStandardTimeReference()

The RTL signals for *notifyStandardTimeReference()* **are specified by** the following table:

RTL signal name TSF_{<instNum>_}NOTIFY_STANDARD_TIME_REFERENCE_ +	**Origin**	**Format**
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
DATA_IN.reference_tai	*FPGA app module*	Two 32-bit vectors
DATA_IN.reference_utc	*FPGA app module*	Two 32-bit vectors
DATA_IN.time_stamp	*FPGA app module*	Two 32-bit vectors
DATA_IN.time_uncertainty	*FPGA app module*	Two 32-bit vectors
DATA_IN.source_id	*FPGA app module*	8-bit vector

Table 13 RTL signals for *notifyStandardTimeReference()*

The `int` type for *SourceId* in the *PIM specification* (not a valid IDL type) **maps to** an 8-bit vector.

The dynamic behavior for *notifyStandardTimeReference()* **is specified by** the following chronogram:

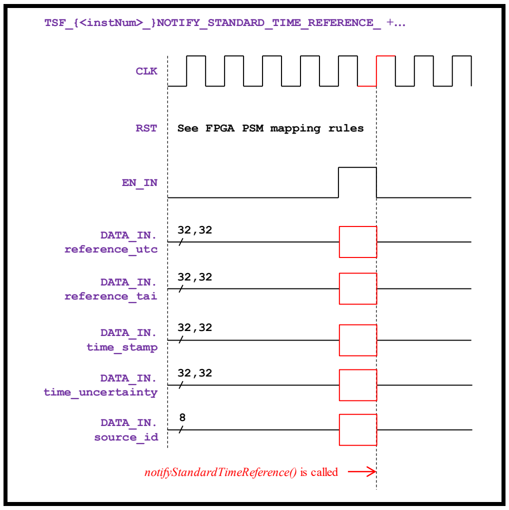

Figure 10 Dynamic behavior for *notifyStandardTimeReference()*

2.2.5 TimeService::SpecificTimes::SpecificTimeHandling

2.2.5.1 setSpecificTime()

The RTL signals for *setSpecificTime()* **are specified by** the following table:

RTL signal name `TSF_{<instNum>_}SET_SPECIFIC_TIME_` +	Origin	Format
`CLK`	*FPGA façade*	1-bit signal
`RST`	*FPGA façade*	1-bit signal
`EN_IN`	*FPGA façade*	1-bit signal
`DATA_IN.specific_time_id`	*FPGA app module*	16-bit vector
`DATA_IN.specific_time`	*FPGA app module*	Two 32-bit vectors
`DATA_IN.time_stamp`	*FPGA app module*	Two 32-bit vectors
`DATA_IN.time_uncertainty`	*FPGA app module*	Two 32-bit vectors
`IRQ_FUTURE_TIME_STAMP` (optional)	*FPGA app module*	1-bit signal
`IRQ_INVALID_SPECIFIC_TIME_ID` (optional)	*FPGA app module*	1-bit signal

Table 14 RTL signals for *setSpecificTime()*

The `int` type for *specificTimeId* in the *PIM specification* (not a valid IDL type) **maps to** a 16-bit vector.

The dynamic behavior for *setSpecificTime()* **is specified by** the following chronogram:

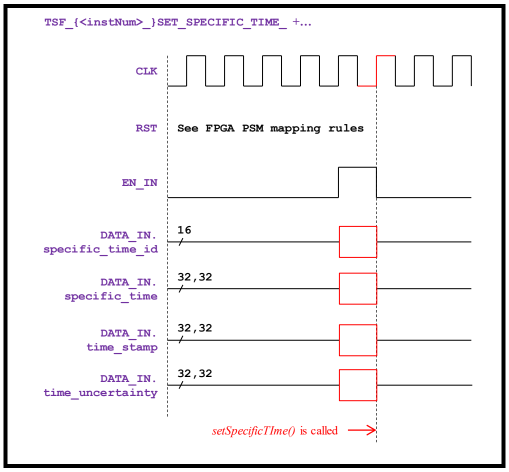

Figure 11 Dynamic behavior for *setSpecificTime()*

2.2.5.2 getSpecificTime()

The RTL signals for *getSpecificTime()* **are specified by** the following table:

RTL signal name `TSF_{<instNum>_}GET_SPECIFIC_TIME_` +	**Origin**	**Format**
`CLK`	*FPGA façade*	1-bit signal
`RST`	*FPGA façade*	1-bit signal
`EN_IN`	*FPGA façade*	1-bit signal
`EN_OUT`	*FPGA app module*	1-bit signal
`DATA_IN.specific_time_id`	*FPGA app module*	16-bit vector
`DATA_OUT.specific_time`	*FPGA app module*	Two 32-bit vectors
`DATA_OUT.time_stamp`	*FPGA app module*	Two 32-bit vectors
`DATA_OUT.time_uncertainty`	*FPGA app module*	Two 32-bit vectors
`IRQ_INVALID_SPECIFIC_TIME_ID` (optional)	*FPGA app module*	1-bit signal

Table 15 RTL signals for *getSpecificTime()*

The dynamic behavior for *getSpecificTime()* **is specified by** the following chronogram:

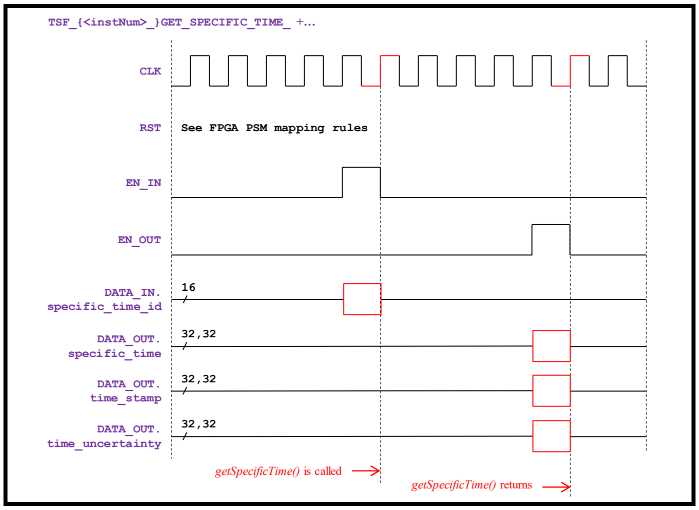

Figure 12 Dynamic behavior for *getSpecificTime()*

2.2.6 *TimeService::SpecificTimes::SettingsNotification*

2.2.6.1 notifySpecificTimeSetting()

The RTL signals for *notifySpecificTimeSetting()* **are specified by** the following table:

RTL signal name TSF_{<instNum>_}NOTIFY_SPECIFIC_TIME_SETTING_ +	Origin	Format
CLK	*FPGA façade*	1-bit signal
RST	*FPGA façade*	1-bit signal
EN_IN	*FPGA façade*	1-bit signal
DATA_IN.specific_time_id	*FPGA app module*	16-bit vector
DATA_IN.specific_time	*FPGA app module*	Two 32-bit vectors
DATA_IN.time_stamp	*FPGA app module*	Two 32-bit vectors
DATA_IN.time_uncertainty	*FPGA app module*	Two 32-bit vectors

Table 16 RTL signals for *notifySpecificTimeSetting()*

The dynamic behavior for *notifySpecificTimeSetting()* **is specified by** the following chronogram:

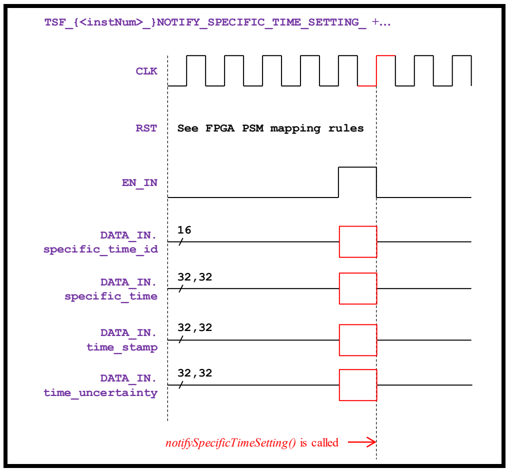

Figure 13 Dynamic behavior for *notifySpecificTimeSetting()*

3 FPGA PSM constants

This normative section specifies FPGA PSM constants.

3.1 PIM version

In accordance with the FPGA section of *WInnForum Facilities PSMs Mapping Rules* [Ref3], the `TSF_PIM_VERSION` constant **is equal to** `0x010100`.

4 VHDL programming

This section specifies additional normative concepts supported by *FPGA time services* programmed using VHDL [Ref5], essentially through specification of VHDL packages to be used by conformant *FPGA façades* and *FPGA applicative modules* (see section 1.2).

The supported VHDL versions are vhdl-93 and all subsequent versions.

The specified VHDL packages have been successfully compiled using:

- Modelsim 10.4a,
- Syntax Checker of Xilinx Vivado 2021.1.

4.1 VHDL library

The specified packages need to be compiled in **`tsf_api`** library.

4.2 pkg_tsf_api_types.vhd

The **pkg_tsf_api_types.vhd** file **is defined as** the standard VHDL package for the *types* of the *FPGA PSM specification*.

Specific *types* are introduced for *sourceId* and *specificTimeId* parameters, specified as ***int*** (not a valid IDL *type*) by *Time Service Facility PIM Specification* [Ref1].

The content of **pkg_tsf_api_types.vhd is specified as**:

```
library ieee;
use ieee.std_logic_1164.all;

package pkg_tsf_api_types is

  -- Constant reflecting version of the PIM Specification
  constant C_TSF_PIM_VERSION : std_logic_vector( 23 downto 0):= X"010100";

  -- SourceId_type (int in the [PIM], no dedicated type)
  subtype SourceId_type is std_logic_vector(7 downto 0);

  -- SpecificTimeId_type (int in the [PIM], no dedicated type)
  subtype SpecificTimeId_type is std_logic_vector(15 downto 0);

  -- TimeValue ([PIM] §3.4.1)
  -- struct TimeValue {
  --  long seconds,             // in seconds
  --  long nanoseconds};        // in nanoseconds (<1.000.000.000)
  --  const TimeValue UndefinedTime = {0xFFFFFFFF, 0xFFFFFFFF};

  type TimeValue_type is record
   seconds      :            std_logic_vector(31 downto 0);
   nanoseconds :std_logic_vector(31 downto 0);
  end record TimeValue_type;
  constant UndefinedTime : TimeValue_type := (
   seconds => X"FFFFFFFF",
   nanoseconds => X"FFFFFFFF");

  -- TimeUncertainty ([PIM] §3.4.2)
  -- typedef long TimeUncertainty;
  subtype TimeUncertainty_type is std_logic_vector(31 downto 0);
  constant Beyond2SecTimeUncertainty : TimeUncertainty_type     := X"FFFFFFF0";
  constant Beyond4SecTimeUncertainty : TimeUncertainty_type     := X"FFFFFFF1";
  constant Beyond8SecTimeUncertainty : TimeUncertainty_type     := X"FFFFFFF2";
  constant Beyond16SecTimeUncertainty : TimeUncertainty_type    := X"FFFFFFF3";
  constant Beyond32SecTimeUncertainty : TimeUncertainty_type    := X"FFFFFFF4";
  constant Beyond64SecTimeUncertainty : TimeUncertainty_type    := X"FFFFFFF5";
  constant Beyond128SecTimeUncertainty : TimeUncertainty_type   := X"FFFFFFF6";
  constant Beyond256SecTimeUncertainty : TimeUncertainty_type   := X"FFFFFFF7";
  constant Beyond512SecTimeUncertainty : TimeUncertainty_type   := X"FFFFFFF8";
  constant Beyond1024SecTimeUncertainty : TimeUncertainty_type  := X"FFFFFFF9";
  constant Beyond2048SecTimeUncertainty : TimeUncertainty_type  := X"FFFFFFFA";
  constant Beyond4096SecTimeUncertainty : TimeUncertainty_type  := X"FFFFFFFB";
  constant Beyond8192SecTimeUncertainty : TimeUncertainty_type  := X"FFFFFFFC";
  constant Beyond16384SecTimeUncertainty : TimeUncertainty_type := X"FFFFFFFD";
  constant UnknownTimeUncertainty   : TimeUncertainty_type      := X"FFFFFFFE";
  constant UndefinedTimeUncertainty  : TimeUncertainty_type     := X"FFFFFFFF";

  -- RateUncertainty ([PIM] §3.4.3)
  -- typedef long RateUncertainty;
  subtype RateUncertainty_type is std_logic_vector(31 downto 0);
  constant UnknownRateUncertainty : RateUncertainty_type := X"FFFFFFFF";

end package pkg_tsf_api_types;
```

4.3 pkg_tsf_primitives_parameters.vhd

The file `pkg_tsf_primitives_parameters.vhd` **is defined as** the standard VHDL package for *parameters* of the *FPGA PSM specification primitives*.

For *primitives* with optional implementation of *exceptions* specified by the *PIM specification*, the corresponding dedicated 1-bit RTL signal is not specified in the VDHL file.

The content of `pkg_tsf_primitives_parameters.vhd` **is specified as**:

```
library ieee;
use ieee.std_logic_1164.all;

library tsf_api;
use tsf_api.pkg_tsf_api_types.all;

package pkg_tsf_primitives_parameters is

  -- TimeService::TerminalTime::TerminalTimeAccess ([PIM] §3.1.2)
  -- getTerminalTime ([PIM] §3.1.2.1.2)
  type getTerminalTime_outputType is record
    terminal_time : TimeValue_type;
  end record;
  -- PIM Exceptions: none

  -- getTerminalTimeUncertainty ([PIM] § 3.1.2.2.2)
  type getTerminalTimeUncertainty_outputType is record
    terminalTimeRateUncertainty : RateUncertainty_type;
  end record;
  -- PIM Exceptions: none

  -- TimeService::SystemTime::SystemTimeAccess([PIM] §3.1.3)
  -- getCurrentTAI ([PIM] §3.1.3.1.2)
  type getCurrentTAI_outputType is record
    currentTAI      : TimeValue_type;
    timeStamp       : TimeValue_type;
    timeUncertainty : TimeUncertainty_type;
  end record;
  -- PIM Exceptions: none

  -- getCurrentUTC ([PIM] §3.1.3.2.2)
  type getCurrentUTC_outputType is record
    currentUTC      : TimeValue_type;
    timeStamp       : TimeValue_type;
    timeUncertainty : TimeUncertainty_type;
  end record;
  -- PIM Exceptions: none

  -- getLastUpdateTAI ([PIM] §3.1.3.3.2)
  type getLastUpdateTAI_outputType is record
    lastUpdateTAI   : TimeValue_type;
    timeStamp       : TimeValue_type;
    timeUncertainty : TimeUncertainty_type;
  end record;
  -- PIM Exceptions: none

  -- getLastUpdateUTC ([PIM] §3.1.3.4.2)
  type getLastUpdateUTC_outputType is record
    lastUpdateUTC   : TimeValue_type;
    timeStamp       : TimeValue_type;
    timeUncertainty : TimeUncertainty_type;
```

```
end record;
-- PIM Exceptions: none

-- TimeService::SystemTime::StandardTimeProvision ([PIM] §3.1.4)
-- provideTAI ([PIM] §3.1.4.1.2)
type provideTAI_inputType is record
  providedTAI     : TimeValue_type;
  timeStamp       : TimeValue_type;
  timeUncertainty : TimeUncertainty_type;
  sourceId        : SourceId_type;
end record;
-- PIM Exceptions: FutureTimeStamp

-- provideUTC ([PIM] §3.1.4.2.2)
type provideUTC_inputType is record
  providedUTC     : TimeValue_type;
  timeStamp       : TimeValue_type;
  timeUncertainty : TimeUncertainty_type;
  sourceId        : SourceId_type;
end record;
-- PIM Exceptions: FutureTimeStamp

--  TimeService::StandardTimes::ReferencesNotification ([PIM] §3.1.5)
-- notifyStandardTimeReference(§3.1.5.1.2)
type notifyStandardTimeReference_inputType is record
  referenceTAI    : TimeValue_type;
  referenceUTC    : TimeValue_type;
  timeStamp       : TimeValue_type;
  timeUncertainty : TimeUncertainty_type;
  sourceId        : SourceId_type;
end record;
-- PIM Exceptions: none

-- TimeService::SpecificTimes::SpecificTimeHandling ([PIM] §3.1.6)
-- setSpecificTime([PIM] §3.1.6.1.2)
type setSpecificTime_inputType is record
  specificTimeId  : SpecificTimeId_type;
  specificTime    : TimeValue_type;
  timeStamp       : TimeValue_type;
  timeUncertainty : TimeUncertainty_type;
end record;
-- PIM Exceptions: FutureTimeStamp | InvalidSpecificTimeId

-- getSpecificTime([PIM] §3.1.6.2.2)
type getSpecificTime_inputType is record
  specificTimeId  : SpecificTimeId_type;
end record;
type getSpecificTime_outputType is record
  specificTime    : TimeValue_type;
  timeStamp       : TimeValue_type;
  timeUncertainty : TimeUncertainty_type;
end record;
-- PIM Exceptions : InvalidSpecificTimeId

--  TimeService::SpecificTimes::SettingsNotification ([PIM] §3.1.7)
-- notifySpecificTimeSetting([PIM] §3.1.7.1.2)
type notifySpecificTimeSetting_type  is record
  specificTimeId  : SpecificTimeId_type;
  specificTime    : TimeValue_type;
  timeStamp       : TimeValue_type;
  timeUncertainty : TimeUncertainty_type;
end record;
```

```
  -- PIM Exceptions: none

end package pkg_tsf_primitives_parameters;
```

5 References

5.1 Referenced documents

[Ref1] *Time Service Facility PIM Specification*, The Wireless Innovation Forum, WINNF-TS-3004, V1.1.1, 18 January 2022
https://sds.wirelessinnovation.org/specifications-and-recommendations
https://winnf.memberclicks.net/assets/work_products/Specifications/WINNF-TS-3004-V1.1.1.pdf

[Ref2] *Principles for WInnForum Facility Standards*, The Wireless Innovation Forum, WINNF-TR-2007, V1.0.0, 13 October 2020
https://sds.wirelessinnovation.org/specifications-and-recommendations
https://winnf.memberclicks.net/assets/work_products/Reports/WINNF-TR-2007-V1.0.0.pdf

[Ref3] *WInnForum Facilities PSMs Mapping Rules*, The Wireless Innovation Forum, WINNF-TR-2008, V1.0.1, 18 January 2022
https://sds.wirelessinnovation.org/specifications-and-recommendations
https://winnf.memberclicks.net/assets/work_products/Reports/WINNF-TR-2008-V1.0.1.pdf

[Ref4] *Register-transfer level*, Wikipedia
https://en.wikipedia.org/wiki/Register-transfer_level

[Ref5] *Standard VHDL Language Reference Manual*, IEEE, 2002
http://ieeexplore.ieee.org/document/1003477/

The URLs above were successfully accessed at release date.

END OF THE DOCUMENT