TodSlaveClock

Reference Manual

Product Info	
Product Manager	Sven Meier
Author(s)	Sven Meier
Reviewer(s)	-
Version	1.6
Date	21.10.2020

Overview

NetTimeLogic's Time Of Day (TOD) Slave Clock is a full hardware (FPGA) only implementation of a synchronization core able to synchronize to a Time of Day source via NMEA or UBX over UART.
The whole message parsing, algorithms and calculations are implemented in the core, no CPU is required. This allows running TOD synchronization completely independent and standalone from the user application. The core can be configured either by signals or by an AXI4Light-Slave Register interface.
This core only adapts the second part of the clock, and does no drift or offset correction in the sub second range, this shall be done in a combination with the PPS Slave Clock.

Key Features:

- Time of Day Slave Clock
- Built-in UART receiver with configurable baudrate
- NMEA message parser
- Support for GPS (GPxxx), GLONASS (GLxxx), GALILEO (GAxxx), BEIDOU (GBxxx) or Combined (GNxxx) NMEA messages
- Support for NMEA GxZDA and GxRMC messages for time extraction
- Quality supervision and filtering of GxRMC messages
- Support for UBX (uBlox®) NAV_TIME_UTC and NAV_TIME_LS messages
- Hardware time conversion from Time of Day format (hh:mm:ss dd:mm:yyyy) into seconds since midnight 1.1.1970 (Linux, TAI, PTP)
- Second adjustment at the local second overflow
- In UBX mode provides Current UTC offset to TAI and Leap Second information
- In combination with a PPS Slave Clock from NetTimeLogic: synchronization accuracy: +/- 25ns
- AXI4 Light register set or static configuration

Revision History

This table shows the revision history of this document.

Version	Date	Revision
0.1	28.12.2015	First draft
1.0	13.05.2016	First release
1.1	19.05.2016	Added structured types section
1.2	20.12.2017	Status interface added
1.3	17.02.2020	Added Polarity swap mode
1.4	28.07.2020	Added more Error indications
1.5	30.07.2020	Added Id check for different GNSS system
1.6	21.10.2020	Added UBX Support

Table 1: Revision History

Content

Definitions

Definitions	
NMEA 0183	Is a combined electrical and data specification for communication between marine electronics such as echo sounder, sonars, anemometer, gyrocompass, autopilot, GPS receivers and many other types of instruments. The NMEA 0183 standard uses a simple ASCII, serial communications protocol that defines how data are transmitted in a "sentence" from one "talker" to multiple "listeners" at a time
Tod Slave Clock	A clock that can synchronize itself to NMEA 0183 messages via UART
PI Servo Loop	Proportional–integral servo loop, allows for smooth corrections
Offset	Phase difference between clocks
Drift	Frequency difference between clocks

Table 2: Definitions

Abbreviations

Abbreviations	
AXI	AMBA4 Specification (Stream and Memory Mapped)
IRQ	Interrupt, Signaling to e.g. a CPU
PPS	Pulse Per Second
TOD	Time of Day
TS	TOD Slave
GNSS	Global Navigation Satellite System
BEIDOU	Chinese GNSS System
GALILEO	European GNSS System
GLONASS	Russian GNSS System
GPS	American GNSS System (often also used instead of GNSS)
NMEA	National Marine Electronics Association

TS	Timestamp
TB	Testbench
UART/RS232	Universal Asynchronous Receiver Transmitter
LUT	Look Up Table
FF	Flip Flop
RAM	Random Access Memory
ROM	Read Only Memory
FPGA	Field Programmable Gate Array
VHDL	Hardware description Language for FPGA's
UTC	Coordinated Universal Time, popularly known as GMT (Greenwich Mean Time)
TAI	Temps Atomique International, is the international atomic time scale based on a continuous counting of the SI second. TAI is currently ahead of UTC by 36 seconds. TAI is always ahead of GPS by 19 seconds.

Table 3: Abbreviations

1 Introduction

1.1 Context Overview

The TOD Slave Clock is meant as a co-processor handling Time of Day (TOD) inputs in the form of NMEA or UBX messages via UART. It receives NMEA or UBX messages from a NMEA/UBX source (GPS receiver) via an UART/RS232 interface; it does not send any message to the source though.
This means it parses and processes NMEA or UBX messages directly in hardware, converts the time into the same format and time base as the Counter Clock and sets the time of the Counter Clock if not correct.
The TOD Slave Clock is designed to work in cooperation with the Counter Clock core from NetTimeLogic (not a requirement). It can be combined with a PPS Slave clock to synchronize for e.g. to a GPS receiver. Offset and drift are then corrected by the PPS Slave Clock to the next second and the TOD Slave Clock will correct the absolute time on seconds level.
It contains an AXI4Light slave for configuration and supervision from a CPU, this is however not required since the TOD Slave Clock can also be configured statically via signals/constants directly from within the FPGA.

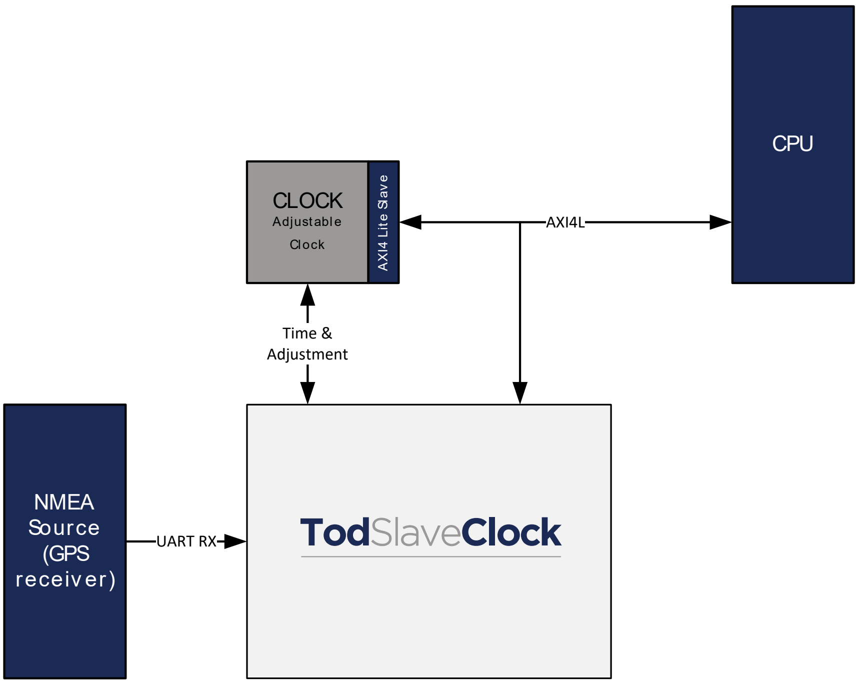

Figure 1: Context Block Diagram

1.2 Function

The TOD Slave Clock takes an UART input and converts the UART with configurable baud rate to an AXI byte stream. This byte stream parses the NMEA data stream for GxZDA and GxRMC messages or UBX data stream for NAV_TIME_UTC and NAV_TIME_LS messages and extracts the UTC time in a time of day format and converts it in case of NMEA from ASCII to binary. The next step is to convert the UTC time in the hh:mm:ss dd:mm:yyyy format to seconds since midnight 1.1.1970 (no fractions of seconds used) taking leap years into account. To this time an additional offset is added or subtracted to convert the UTC time to TAI time (in case of UBX this information is received from the GPS Receiver) or any other time base (leap seconds or different start of epoch). This time is then taken as reference time and waited for the next local second overflow before the local time is overwritten if required. If no difference exists, no overwrite takes place. This will cause a time jump of a second every now and then if the NMEA message reception drifts away (slips over a second of the local clock) from the local clock if not compensated (a PPS Slave would compensate the drift and offset and avoid these time jumps).

1.3 Architecture

The core is split up into different functional blocks for reduction of the complexity, modularity and maximum reuse of blocks. The interfaces between the functional blocks are kept as small as possible for easier understanding of the core.

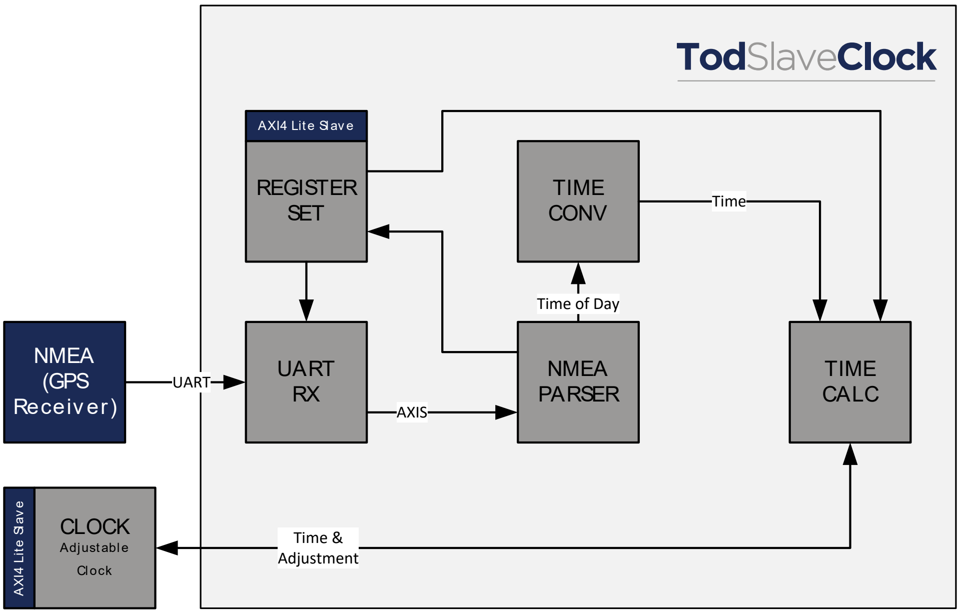

Figure 2: Architecture Block Diagram

Register Set

This block allows reading status values and writing configuration.

UART Receiver

This block is an UART Receiver which converts the serial stream into a byte aligned AXI stream.

NMEA Parser

This block parses the data stream for time messages from the NMEA or UBX source and extracts the UTC in time of day format and in case of UBX also the leap second and UTC offset information

Time Converter

This block converts the UTC time in time of day format into TAI format in seconds since 1.1.1970 without leap seconds for NMEA and with leap seconds for UBX

Time Calculator

This block adds or subtracts additional offsets for leap second corrections or different time bases, compares this with the time of the local clock, and corrects the local clock if needed on the next second wraparound of the local clock.

2 NMEA Basics

2.1 Interface

NMEA 0183 is a standard for communication between navigation equipment on ships defined by the National Marine Electronics Association which also defines how the communication between a GPS receiver and a PC shall look like.
The NMEA 0183 standard uses a simple ASCII, serial communications protocol that defines how data are transmitted in messages from one source to multiple sinks at a time.

Typical Baud rate	4800
Data bits	8
Parity	None
Stop bits	1
Handshake	None

2.2 Messages

NMEA messages always start with a "$" character, followed by the source id which is "GP" for GPS, "GL" for GLONASS, "GA" for GALILEO, "GB" for BEIDOU or "GN" for Combined, followed by a three character message type. Then a message type dependent number of fields of different lengths follow, each field separated with a "," character. The last field is terminated with a "*" character and followed by a checksum in hexadecimal format.

There are many message types defined for GNSS sources, however only a few contain the time of day: ZDA (Date and Time) and RMC (Recommended Minimum Data).

The message format of the two messages used are described in the next chapters, be aware that some GNSS receiver have higher accuracy on some values and will add fractions, so fields don't always have the same width (e.g. seconds might be with or without fractions).

2.2.1 ZDA – Date and Time

This message is specifically made for transferring time. It has the local time offset for local time but this is not used.

$GxZDA,hhmmss.ss,dd,mm,yyyy,aa,bb*CC

- x: P (GPS), L (GLONASS), A (GALILEO), B (BEIDOU), N (All)
- hh: hours (00 - 23)
- mm: minutes (00 - 59)
- ss.ss: decimal seconds (00.99 - 60.99)
- dd: day (01 – 31)
- mm: month (01 – 12)
- yyyy: year (1970 – 2106)
- aa: local zone hours (ignored)
- bb: local zone minutes (ignored)
- *CC: checksum (00-FF)

2.2.2 RMC – Recommended Minimum Data

This message is supported by all GPS receivers, it describes the minimum message that a GPS receiver has to be able to output when conforming with the NMEA 2.0standard.

$GxRMC,hhmmss.ss,S,xxxx.xxx,N,xxxx.xxx,E,vvv.vv,aaa.a,ddmmyy,vvv.v,W*CC

- x: P (GPS), L (GLONASS), A (GALILEO), B (BEIDOU), N (All)
- hh: hours (00 - 23)
- mm: minutes (00 - 59)
- ss.ss: decimal seconds (00.99 - 60.99)
- S status A=active or V=Void
- xxxx.xxx,N latitude (ignored)
- xxxx.xxx,E longitude (ignored)
- vvv.vv speed (ignored)
- aaa.a course (ignored)
- dd: day (01 – 31)
- mm: month (01 – 12)
- yyyy: year (1970 – 2069)
- vvv.v,W: magnetic variation (ignored)
- *CC: checksum (00-FF)

2.3 Message rate and phase

The message rates of these two messages shall be set to once per second if possible. It is important that the received NMEA message is received in a rather fixed phase to the second overflow (PPS) e.g. always around 500ms after the UTC second overflow other ways time jumps can happen. Only one of the messages shall be available or only one message type shall be enabled in the core.

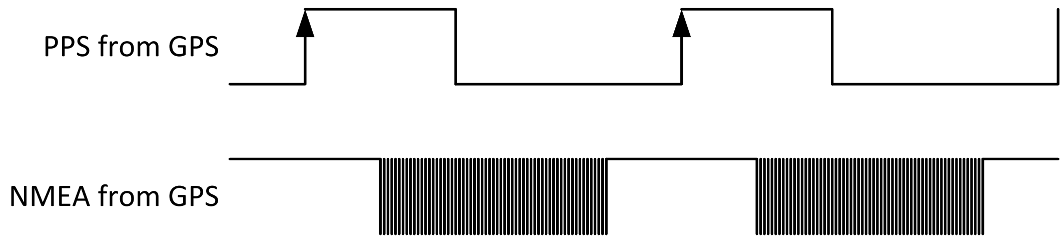

Figure 3: NMEA to PPS alignment

2.4 UTC vs TAI time bases

Both messages contain the time of day on UTC base. UTC has an offset to TAI which is the time base normally used for the Counter Clock. This time offset can be set in the core so the local clock can still run on a TAI base. UTC in comparison to TAI or GPS time has so called leap seconds. A leap second is an additional second which is either added or subtracted from the current time to adjust for the earth rotation variation over time. Until 2020 UTC had additional 37 leap seconds, therefore TAI time is currently 37 seconds ahead of UTC. The issue with UTC time is, that the time makes jumps with the leap seconds which may cause that synchronized nodes go out of sync for a couple of seconds. Leap seconds are normally introduced at midnight of either the 30 of June or 31 of December. For an additional leap second the seconds counter of the UTC time will count to 60 before wrapping around to zero, for one fewer leap second the UTC second counter will wrap directly from 58 to 0 by skipping 59 (this has not happened yet).
Be aware that this core takes no additional precautions to handle leap seconds in case of NMEA since NMEA just DOES NOT provide any information about UTC Offset and LEAP seconds, so it will make a time jump at a UTC leap second and will lose synchronization since it thinks that it has an offset of one second at tries to adjust this offset. A way to avoid this is to disable the adjustment at the two dates right before midnight (e.g. one minute earlier), wait for the leap second to happen, fetch some time server to get the new offset between TAI and UTC, set this offset

to the core and enable the core again. This way the local clock on TAI base makes no jump since the new offset is already taken into account. The only issue with this is that for the time around midnight the clock is free running without a reference.

3 UBX Basics

3.1 Interface

UBX is a proprietary protocol from uBlox® for communication between a GPS receiver and GPS Sink.
The UBX protocl uses a simple binary, serial communications protocol that defines how data are transmitted in messages from one source to multiple sinks at a time.

Typical Baud rate	4800
Data bits	8
Parity	None
Stop bits	1
Handshake	None

Multibyte values are transferred in little endian format (LSB first)

3.2 Messages

UBX messages always start with 0xB5 followed by 0x62 for synchronization of message boundaries. Then comes a Message Class byte (0x01 for the ones we look at) followed by a message ID byte and a 16bit length field. Then follows the pay-load of the length specified before followed by a two byte checksum.

There are many message types defined for GNSS sources, however only a few contain the time of day and information about leap seconds and UTC offset: NAV_TIME_UTC (Date and Time) and NAV_TIME_LS (Leap Second and UTC offset).

The message format of the two messages used are described in the next chapters,

3.2.1 NAV_TIME_UTC – Date and Time

This message is specifically made for transferring UTC time.

Message	**UBX-NAV-TIMEUTC** **UTC time solution**				
Type	Periodic/polled				
Comment	Note that during a leap second there may be more or less than 60 seconds in a minute. See the description of leap seconds in the Integration manual for details.				
Message structure	*Header*	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xb5 0x62	0x01	0x21	20	*see below*	CK_A CK_B

Payload description:

Byte offset	*Type*	*Name*	*Scale*	*Unit*	*Description*
0	U4	iTOW	-	ms	GPS time of week of the navigation epoch. See the section iTOW timestamps in Integration manual for details.
4	U4	tAcc	-	ns	Time accuracy estimate (UTC)
8	I4	nano	-	ns	Fraction of second, range -1e9 .. 1e9 (UTC)
12	U2	year	-	y	Year, range 1999..2099 (UTC)
14	U1	month	-	month	Month, range 1..12 (UTC)
15	U1	day	-	d	Day of month, range 1..31 (UTC)
16	U1	hour	-	h	Hour of day, range 0..23 (UTC)
17	U1	min	-	min	Minute of hour, range 0..59 (UTC)
18	U1	sec	-	s	Seconds of minute, range 0..60 (UTC)
19	X1	valid	-	-	Validity Flags
bit 0	U:1	validTOW	-	-	1 = Valid Time of Week (see section Time validity in Integration manual for details)
bit 1	U:1	validWKN	-	-	1 = Valid Week Number (see section Time validity in Integration manual for details)
bit 2	U:1	validUTC	-	-	1 = Valid UTC Time
bits 7…4	U:4	utcStandard	-	-	UTC standard identifier. (Not supported for protocol versions less than 15.00) • 0 = Information not available • 1 = Communications Research Labratory (CRL), Tokyo, Japan • 2 = National Institute of Standards and Technology (NIST) • 3 = U.S. Naval Observatory (USNO) • 4 = International Bureau of Weights and Measures (BIPM)

Figure 1: UBX NAV TIME UTC Frame format

3.2.2 NAV_TIME_LS – Leap Seconds and UTC Offset

This message contains information about UTC offsets and Leap Seconds.

Message	**UBX-NAV-TIMELS** **Leap second event information**						
Type	Periodic/polled						
Comment	Information about the upcoming leap second event if one is scheduled.						
Message structure	*Header* 0xb5 0x62	*Class* 0x01	*ID* 0x26	*Length (Bytes)* 24	*Payload* *see below*	*Checksum* CK_A CK_B	

Payload description:

Byte offset	*Type*	*Name*	*Scale*	*Unit*	*Description*
0	U4	`iTOW`	-	ms	GPS time of week of the navigation epoch. See the section iTOW timestamps in Integration manual for details.
4	U1	`version`	-	-	Message version (0x00 for this version)
5	U1[3]	`reserved0`	-	-	Reserved
8	U1	`srcOfCurrLs`	-	-	Information source for the current number of leap seconds. • 0 = Default (hardcoded in the firmware, can be outdated) • 1 = Derived from time difference between GPS and GLONASS time • 2 = GPS • 3 = SBAS • 4 = BeiDou • 5 = Galileo • 6 = Aided data • 7 = Configured • 255 = Unknown
9	I1	`currLs`	-	s	Current number of leap seconds since start of GPS time (Jan 6, 1980). It reflects how much GPS time is ahead of UTC time. Galileo number of leap seconds is the same as GPS. BeiDou number of leap seconds is 14 less than GPS. GLONASS follows UTC time, so no leap seconds.

Figure 1: UBX NAV TIME UTC Frame format

3.3 Message rate and phase

The message rates of these two messages shall be set to once per second if possible. It is important that the received UBX message is received in a rather fixed phase to the second overflow (PPS) e.g. always around 500ms after the UTC second overflow other ways time jumps can happen. Only one of the messages shall be available or only one message type shall be enabled in the core.

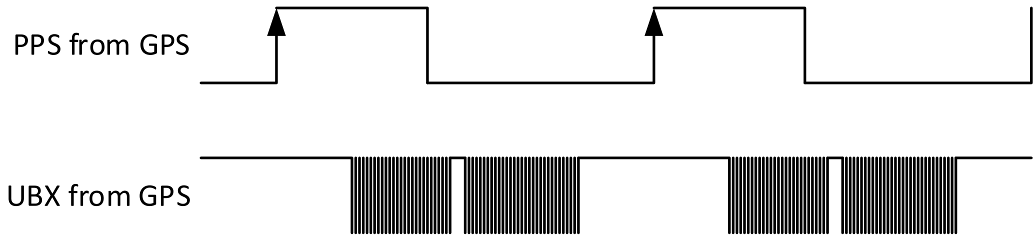

Figure 2: UBX to PPS alignment

3.4 UTC vs TAI time bases

NAV_TIME_UTC contains the time of day on UTC base. UTC has an offset to TAI which is the time base normally used for the Counter Clock. This time offset will be extracted from the NAV_TIME_LS message and corrected so the local clock can still run on a TAI base. UTC in comparison to TAI or GPS time has so called leap seconds. A leap second is an additional second which is either added or subtracted from the current time to adjust for the earth rotation variation over time. Until 2020 UTC had additional 37 leap seconds, therefore TAI time is currently 37 seconds ahead of UTC. The issue with UTC time is, that the time makes jumps with the leap seconds which may cause that synchronized nodes go out of sync for a couple of seconds. Leap seconds are normally introduced at midnight of either the 30 of June or 31 of December. For an additional leap second the seconds counter of the UTC time will count to 60 before wrapping around to zero, for one fewer leap second the UTC second counter will wrap directly from 58 to 0 by skipping 59 (this has not happened yet).

Since with UBX we get the leap second and UTC offset information, the core just disables adjustments 4 seconds before and after midnight UTC of the two dates (actually 4 dates, but only two were used so far) until a new UTC offset is available and from the calculations there is no time jump because UTC made a jump and the UTC offset was also increased/decreased which will lead to a continuous time. Also it would be possible to do the same mechanism as for NMEA where the core is disabled before midnight and enabled after midnight, but this would be redundant.

4 Register Set

This is the register set of the TOD Slave Clock. It is accessible via AXI4 Light Memory Mapped. All registers are 32bit wide, no burst access, no unaligned access, no byte enables, no timeouts are supported. Register address space is not contiguous. Register addresses are only offsets in the memory area where the core is mapped in the AXI inter connects. Non existing register access in the mapped memory area is answered with a slave decoding error.

4.1 Register Overview

Registerset Overview			
Name	Description	Offset	Access
Tod SlaveControl Reg	Tod Slave Enable Control Register	0x00000000	RW
Tod SlaveStatus Reg	Tod Slave Error Status Register	0x00000004	WC
Tod SlaveUartPolarity Reg	Tod Slave UART Polarity Register	0x00000008	RW
Tod SlaveVersionReg	Tod Slave Version Register	0x00000004	WC
Tod SlaveCorrection Reg	Tod Slave Second Corrections Register	0x00000010	RW
Tod SlaveUartBaudRate Reg	Tod Slave UART Baud Rate Register	0x00000020	RW
Tod SlaveUtcStatus Reg	Tod Slave UTC Status Register	0x00000030	RO
Tod SlaveTimeToLeapSecond Reg	Tod Slave Time to Leap Second Register	0x00000034	RO

Table 4: Register Set Overview

4.2 Register Descriptions

4.2.1 General

4.2.1.1 TOD Slave Control Register

Used for general control over the TOD Slave Clock, all configurations on the core shall only be done when disabled.

Tod SlaveControl Reg							
Reg Description							
31 30 29	28	27 26 25 24	23 22 21 20 19 18	17	16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1	0
-	PROTOCOL	GNSS	RESERVED	NMEA_ZDA_UBX_UTC	NMEA_RMC_UBX_LS	-	ENABLE
RO	RW	RW	RW	RW	RW		RW
Reset: 0x00000000							
Offset: 0x0000							

Name	Description	Bits	Access
-	Reserved, read 0	Bit: 31:29	RO
PROTOCOL	Serial Protocol: 0=NMEA, 1=UBX	Bit: 28	RW

GNSS	GNSS System to be used: 0=ALL 1=COMBINED 2=GPS 3=GLONASS 4=GALILEO 5=BEIDOU	Bit: 27:24	RW
RESERVED	Reserved, readback possible but no influence (write 0)	Bit: 23:18	RW
NMEA_ZDA_UBX_UTC	Disable NMEA ZDA (if Protocol = NMEA) Disable UBX NAV TIME UTC (if Protocol = UBX)	Bit: 17	RW
NMEA_RMC_UBX_LS	Disable NMEA RMC (if Protocol = NMEA) Disable UBX NAV TIME LS (if Protocol = UBX)	Bit: 16	RW
-	Reserved, read 0	Bit: 15:1	RO
ENABLE	Enable	Bit: 0	RW

4.2.1.2 TOD Slave Status Register

Shows the current status of the TOD Slave Clock.

Tod SlaveStatus Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																													UART_ERROR	CHECKSUM_ERROR	ERROR
RO																													WC	WC	WC
Reset: 0x00000000																															
Offset: 0x0004																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit: 31:2	RO
UART_ERROR	NMEA UART Error (sticky)	Bit: 2	WC
CHECKSUM_ERROR	NMEA Checksum Error (sticky)	Bit: 1	
PARSE_ERROR	NMEA Parser Error (sticky)	Bit: 0	

4.2.1.3 TOD Slave Polarity Register

Used for setting the UART signal polarity, shall only be done when disabled. Default value is set by the UartPolarity_Gen generic.

Tod SlavePolarity Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-																															POLARITY
RO																															RW
Reset: 0x0000000X																															
Offset: 0x0008																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:1	RO
POLARITY	UART Polarity (0 = Inversed, 1 = normal UART)	Bit: 0	RW

4.2.1.4 TOD Slave Version Register

Version of the IP core, even though is seen as a 32bit value, bits 31 down to 24 represent the major, bits 23 down to 16 the minor and bits 15 down to 0 the build numbers.

Tod SlaveVersion Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION																															
RO																															
0xXXXXXXXX																															
Offset: 0x000C																															

Name	Description	Bits	Access
VERSION	Version of the core	Bit: 31:0	RO

4.2.1.5 TOD Slave Correction Register

Correction register to compensate for leap seconds between the different time domains. NMEA is UTC time, all other time in the system is TAI, this leads to a correction of 37 seconds by 2020. NMEA has NO message which contains the current UTC offset so this register must be used to pass UTC offset to TAI. For UBX this register shall be set to 0 if the NAV_TIME_LS message is enabled.

Tod SlaveCorrection Reg	
Reg Description	
31	30 … 0
COR_SIGN	COR_S
RW	RW
Reset: 0x00000000	
Offset: 0x0010	

Name	Description	Bits	Access
COR_SIGN	Correction sign	Bit: 31	RW
COR_S	Correction in seconds to the time extracted from the NMEA => used to convert between TAI, UTC and GPS (leap seconds) for UBX this shall be set to 0 if NAV_TIME_LS is enabled	Bit: 30:0	RW

4.2.1.6 TOD Slave UART Baud Rate Register

This set the receive baud rate of the UART. The baud rate can only be changed when the core is disabled. Otherwise the changes have no effect. Only the most common baud rates are available from a range of 1.2k to 2m baud.

Tod SlaveUartBaudRate Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-																												BAUD_RATE			
RO																												RW			
Reset: 0x0000000X																															
Offset: 0x0020																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit: 31:4	RO

BAUD_RATE	Encoded Baudrate of the UART receiver: 0 => 1200 1 => 2400 2 => 4800 3 => 9600 4 => 19200 5 => 38400 6 => 57600 7 => 115200 8 => 230400 9 => 460800 10 => 921600 11 => 1000000 12 => 2000000 >12 => not allowed undefinded Default can be set by generic	Bit: 3:0	RW

4.2.1.7 TOD Slave UTC Status Register

This Register is only available in UBX mode and only filled if UBX is selected as Protocol and the UBX NAV TIME LS message not disabled, otherwise it will be all 0. This allows to read the current Status for the UTC time, e.g. UTC Offset to TAI (our clock runs in TAI time), the announcement of a leap second and which one (Leap59 or Leap61) also it marks if either the UTC Offset information or the Leap Second information is valid (as indicated by the GPS receiver and some checks). The Leap Indications are earliest available 12h before the leap second event and maximum 12 hours after the event happened (depending ion the announcement by the GPS receiver)

Tod SlaveUtcStatus Reg								
Reg Description								
31 – 17	16	15	14	13	12	11 – 9	8	7 – 0
-	LEAP_INFO_VALID	-	LEAP61	LEAP59	LEAP_ANNOUNCE	-	UTC_INFO_VALID	UTC_OFFSET
RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset: 0x00000000								
Offset: 0x0030								

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:17	RO

LEAP_INFO_VALID	Leap Second Information valid = 1	Bit:31:1	RO
-	Reserved, read 0	Bit:15	RO
LEAP61	Reserved, read 0	Bit:14	RO
LEAP59	Reserved, read 0	Bit:13	RO
LEAP_ANNOUNCE	Announce that a Leap Second will happen within the next 12 h	Bit:12	RO
-	Reserved, read 0	Bit:11:9	RO
UTC_INFO_VALID	UTC Offset Information valid = 1	Bit: 8	RO
UTC_OFFSET	Current UTC Offset to TAI	Bit: 7:0	RO

4.2.1.8 TOD Slave Time To Leap Second Register

This Register is only available in UBX mode and only filled if UBX is selected as Protocol and the UBX NAV TIME LS message not disabled, otherwise it will be all 0. This shows the number of Seconds to the next Leap Second (if positive >0) or the number of Seconds since the last Leap Second (if negative <0) or that the Leap Second is in progress (if 0).

Tod SlaveTimeToLeapSecond Reg
Reg Description
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
TIME_TOLEAP
RO
Reset: 0x00000000
Offset: 0x0034

Name	Description	Bits	Access
TIME_TO_LEAP	Time in Seconds to next Leap Second (>0) or since last Leap Second (<0)	Bit: 31:0	RO

5 Design Description

The following chapters describe the internals of the TOD Slave Clock: starting with the Top Level, which is a collection of subcores, followed by the description of all subcores.

5.1 Top Level – Tod Slave

5.1.1.1 Parameters

The core must be parametrized at synthesis time. There are a couple of parameters which define the final behavior and resource usage of the core.

Name	Type	Size	Description
GpsSupport_Gen	boolean	1	Support for GPS (GPxxx) NMEA messages
GlonassSupport_Gen	boolean	1	Support for GLONASS (GPxxx) NMEA messages
GalileoSupport_Gen	boolean	1	Support for GALILEO (GPxxx) NMEA messages
BeidouSupport_Gen	boolean	1	Support for BEIDOU (GPxxx) NMEA messages
CombinedGnss Support_Gen	boolean	1	Support for Combined (GPxxx) NMEA messages
AllGnssSupport_Gen	boolean	1	Support for any GNSS identifier
GxZdaMessage Support_Gen	boolean	1	Support for GxZDA Messages: true = supported, false = not supported
GxRmcMessage Support_Gen	boolean	1	Support for GxRMC Message: true = supported, false = not supported
NmeaSupport_Gen	boolean	1	Support for NMEA Protocol
UbxNavTimeLs MessageSupport_Gen	boolean	1	Support for UBX_NAV_TIME_LS Messages: true = supported, false = not

			supported
UbxNavTimeUtc MessageSupport_Gen	boolean	1	Support for UBX_NAV_TIME_UTC Message: true = supported, false = not supported
UbxSupport_Gen	boolean	1	Support for UBX Protocol
StaticConfig_Gen	boolean	1	If Static Configuration or AXI is used: true = Static, false = AXI
NmeaCorrection_Gen	natural	1	NMEA and UBX correction in seconds for when the message arrive to the next second overflow. There are some GPS receiver which send the NMEA of the next second and some of the current. Default is the next, then no correction is needed
ClockClkPeriod Nanosecond_Gen	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
UartBaudRate_Gen	natural	1	Default Baudrate encoded: 0 => 1200 1 => 2400 2 => 4800 3 => 9600 4 => 19200 5 => 38400 6 => 57600 7 => 115200 8 => 230400 9 => 460800 10 => 921600 11 => 1000000 12 => 2000000
UartPolarity_Gen	boolean	1	true = normal UART (idle '1') false = inversed
AxiAddressRang	std_logic_vector	32	AXI Base Address

Low_Gen			
AxiAddressRange High_Gen	std_logic_vector	32	AXI Base Address plus Regis-terset Size Default plus 0xFFFF
Sim_Gen	boolean	1	If in Testbench simulation mode: true = Simulation, false = Synthesis

Table 5: Parameters

One of the two parameters GxZdaMessageSupport_Gen and GxZdaMessageSupport_Gen has to be true.

5.1.1.2 Structured Types

5.1.1.2.1 Clk_Time_Type

Defined in Clk_Package.vhd of library ClkLib
Type represents the time used everywhere. For this type overloaded operators + and – with different parameters exist.

Field Name	Type	Size	Description
Second	std_logic_vector	32	Seconds of time
Nanosecond	std_logic_vector	32	Nanoseconds of time
Fraction	std_logic_vector	2	Fraction numerator (mostly not used)
Sign	std_logic	1	Positive or negative time, 1 = negative, 0 = positive.
TimeJump	std_logic	1	Marks when the clock makes a time jump (mostly not used)

Table 6: Clk_Time_Type

5.1.1.2.2 Clk_TimeAdjustment_Type

Defined in Clk_Package.vhd of library ClkLib
Type represents the time used everywhere. For this type overloaded operators + and – with different parameters exist.

Field Name	Type	Size	Description
TimeAdjustment	Clk_Time_Type	1	Time to adjust
Interval	std_logic_vector	32	Adjustment interval, for the drift correction this is the denumerator of the rate in nanoseconds (TimeAdjustment every Interval = drift rate), for offset correction this is the period in which the time shall be corrected(TimeAdjustment in Interval), for setting the time this has no mining.
Valid	std_logic	1	Whether the Adjustment is valid or not

Table 7: Clk_TimeAdjustment_Type

5.1.1.2.3 Tod_SlaveStaticConfig_Type

Defined in Tod_SlaveAddrPackage.vhd of library TodLib
This is the type used for static configuration.

Field Name	Type	Size	Description
Protocol	std_logic	1	0=NMEA 1=UBX
Gnss	std_logic_vector	4	Which GNSS mechanism shall be used (mainly used for NMEA) 0=ALL 1=COMBINED 2=GPS 3=GLONASS 4=GALILEO 5=BEIDOU
DisableMessages	std_logic_vector	8	Bit 0: Disable NMEA ZDA (if Protocol = NMEA), Disable UBX NAV TIME UTC (if Proto-

			col = UBX) Bit 1: Disable NMEA RMC (if Protocol = NMEA), Disable UBX NAV TIME LS (if Protocol = UBX) Bits 2-7: Reserved
Polarity	std_logic	1	'1' = normal UART, '0' = inversed signal level UART
Correction	Clk_Time_Type	1	Time to correct the parsed time to correct UTC to TAI or another base.
UartBaudRate	std_logic_vector	4	Baudrate encoded: 0 => 1200 1 => 2400 2 => 4800 3 => 9600 4 => 19200 5 => 38400 6 => 57600 7 => 115200 8 => 230400 9 => 460800 10 => 921600 11 => 1000000 12 => 2000000

Table 8: Tod_SlaveStaticConfig_Type

5.1.1.2.4 Tod_SlaveStaticConfigVal_Type

Defined in Tod_SlaveAddrPackage.vhd of library TodLib
This is the type used for valid flags of the static configuration.

Field Name	Type	Size	Description
Enable_Val	std_logic	1	Enables the TOD Slave

Table 9: Tod_SlaveStaticConfigVal_Type

5.1.1.2.5 Tod_SlaveStaticStatus_Type

Defined in Tod_SlaveAddrPackage.vhd of library TodLib
This is the type used for static status supervision.

Field Name	Type	Size	Description
CoreInfo	Clk_CoreInfo_Type	1	Infor about the Cores state

Table 10: Tod_SlaveStaticConfig_Type

5.1.1.2.6 Tod_SlaveStaticStatusVal_Type

Defined in Tod_SlaveAddrPackage.vhd of library TodLib
This is the type used for valid flags of the static status supervision.

Field Name	Type	Size	Description
CoreInfo_Val	std_logic	1	Core Info valid

Table 11: Tod_SlaveStaticConfigVal_Type

5.1.1.3 Entity Block Diagram

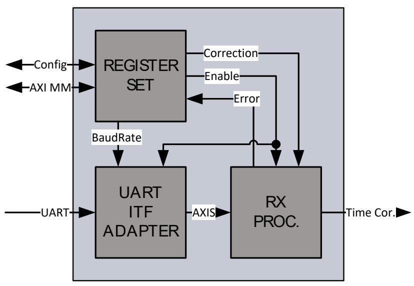

Figure 3: TOD Slave Clock

5.1.1.4 Entity Description

Rx Processor

This module handles all incoming NMEA or UBX message. It extracts the time from the NMEA GxZDA or NMEA GxRMC or UBX NAV_TIME_UTC messages, converts the time from the Time of Day format (with UTC Offset in case of UBX NAV_TIME_LS) to seconds since 1.1.1970 and does the offset and time adjustment of the clock aligned with the local clocks second overflow.
See 5.2.1 for more details.

UART Interface Adapter

This module converts the serial UART signal to an AXI stream. It handles the RS232 protocol data stream with one start, eight data, one stop and no parity. AXI stream from this module is 8 bit width. It can handle baud rates from 1.2k up to 1m.
See 5.2.2 for more details.

Registerset

This module is an AXI Light Memory Mapped Slave. It provides access to all registers and allows configuring the TOD Slave Clock. It can be configured to either run in AXI or StaticConfig mode. If in StaticConfig mode, the configuration of the registers is done via signals and can be easily done from within the FPGA without CPU. If in AXI mode, an AXI Master has to configure the Datasets with AXI writes to the registers, which is typically done by a CPU
See 5.2.3 for more details.

5.1.1.5 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
GpsSupport_Gen	-	boolean	1	Support for GPS (GPxxx) NMEA messages
GlonassSupport_Gen	-	boolean	1	Support for GLONASS (GPxxx) NMEA messages
GalileoSupport_Gen	-	boolean	1	Support for GALILEO (GPxxx) NMEA messages
BeidouSupport_Gen	-	boolean	1	Support for BEIDOU (GPxxx) NMEA messages
CombinedGnss Support_Gen	-	boolean	1	Support for Combined (GPxxx) NMEA messages
AllGnssSupport_Gen	-	boolean	1	Support for any GNSS identifier
GxZdaMessage	-	boolean	1	Support for GxZDA

Support_Gen				Messages
GxRmcMessage Support_Gen	-	boolean	1	Support for GxRMC Message
NmeaSupport_Gen	-	boolean	1	Support for NMEA Protocol
UbxNavTimeLs MessageSupport_Gen	-	boolean	1	Support for UBX_NAV_TIME_LS Messages: true = supported, false = not support-ed
UbxNavTimeUtc MessageSupport_Gen	-	boolean	1	Support for UBX_NAV_TIME_U TC Message: true = supported, false = not support-ed
UbxSupport_Gen	-	boolean	1	Support for UBX Protocol
StaticConfig_Gen	-	boolean	1	If Static Configura-tion or AXI is used
NmeaCorrection_Gen	-	natural	1	NMEA correction in seconds for when the message arrive to the next second overflow.
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond
UartBaudRate_Gen	-	natural	1	Default Baudrate encoded: 0 => 1200 1 => 2400 2 => 4800 3 => 9600 4 => 19200 5 => 38400 6 => 57600 7 => 115200

				8 => 230400 9 => 460800 10 => 921600 11 => 1000000 12 => 2000000
UartPolarity_Gen	-	boolean	1	true = normal UART (idle ‘1’) false = inversed
AxiAddressRang Low_Gen	-	std_logic_vector	32	AXI Base Address
AxiAddressRange High_Gen	-	std_logic_vector	32	AXI Base Address plus Registerset Size
Sim_Gen	-	boolean	1	If in Testbench simulation mode
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Config				
StaticConfig_DatIn	in	Tod_Slave StaticConfig_Type	1	Static Configuration
StaticConfig_ValIn	in	Tod_Slave StaticConfigVal _Type	1	Static Configuration valid
Status				
StaticStatus_DatOut	out	Tod_Slave StaticStatus_Type	1	Static Status
StaticStatus_ValOut	out	Tod_Slave StaticStatusVal _Type	1	Static Status valid
Timer				
Timer1ms_EvtIn	in	std_logic	1	Millisecond timer adjusted with the Clock
Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Adjusted Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted Clock

				Time valid
AXI4 Light Slave				
AxiWriteAddrValid _ValIn	in	std_logic	1	Write Address Valid
AxiWriteAddrReady _RdyOut	out	std_logic	1	Write Address Ready
AxiWriteAddrAddress _AdrIn	in	std_logic_vector	32	Write Address
AxiWriteAddrProt _DatIn	in	std_logic_vector	3	Write Address Protocol
AxiWriteDataValid _ValIn	in	std_logic	1	Write Data Valid
AxiWriteDataReady _RdyOut	out	std_logic	1	Write Data Ready
AxiWriteDataData _DatIn	in	std_logic_vector	32	Write Data
AxiWriteDataStrobe _DatIn	in	std_logic_vector	4	Write Data Strobe
AxiWriteRespValid _ValOut	out	std_logic	1	Write Response Valid
AxiWriteRespReady _RdyIn	in	std_logic	1	Write Response Ready
AxiWriteResp Response_DatOut	out	std_logic_vector	2	Write Response
AxiReadAddrValid _ValIn	in	std_logic	1	Read Address Valid
AxiReadAddrReady _RdyOut	out	std_logic	1	Read Address Ready
AxiReadAddrAddress _AdrIn	in	std_logic_vector	32	Read Address
AxiReadAddrProt _DatIn	in	std_logic_vector	3	Read Address Protocol
AxiReadDataValid _ValOut	out	std_logic	1	Read Data Valid
AxiReadDataReady _RdyIn	in	std_logic	1	Read Data Ready
AxiReadData Response_DatOut	out	std_logic_vector	2	Read Data
AxiReadDataData _DatOut	out	std_logic_vector	32	Read Data Re-sponse
Time of Day Input				
Uart_DatIn	in	std_logic	1	UART from the NMEA source
Time Adjustment Output				
TimeAdjustment _DatOut	out	Clk_TimeAdjustment _Type	1	Time to set hard

TimeAdjustment _ValOut	out	std_logic	1	Time valid
Offset Adjustment Output				
OffsetAdjustment _DatOut	out	Clk_TimeAdjustment _Type	1	Calculated new Offset between Master and Slave (unused)
OffsetAdjustment _ValOut	out	std_logic;	1	Calculated new Offset valid
Drift Adjustment Output				
DriftAdjustment _DatOut	out	Clk_TimeAdjustment _Type	1	Calculated new Drift between Master and Slave Slave (un-used)
DriftAdjustment _ValOut	out	std_logic;	1	Calculated new Drift valid Slave (unused)
Offset Adjustment Input				
OffsetAdjustment _DatIn	in	Clk_TimeAdjustment _Type	1	Calculated new Offset after the PI Servo loop Slave (unused)
OffsetAdjustment _ValIn	in	std_logic;	1	Calculated new Offset after the PI Servo loop valid Slave (unused)
Drift Adjustment Input				
DriftAdjustment _DatIn	in	Clk_TimeAdjustment _Type	1	Calculated new Drift after the PI Servo loop Slave (unused)
DriftAdjustment _ValIn	in	std_logic	1	Calculated new Drift after the PI Servo loop valid Slave (unused)

Table 12: TOD Slave Clock

5.2 Design Parts

The TOD Slave Clock core consists of a couple of subcores. Each of the subcores itself consist again of smaller function block. The following chapters describe these subcores and their functionality.

5.2.1 RX Processor

5.2.1.1 Entity Block Diagram

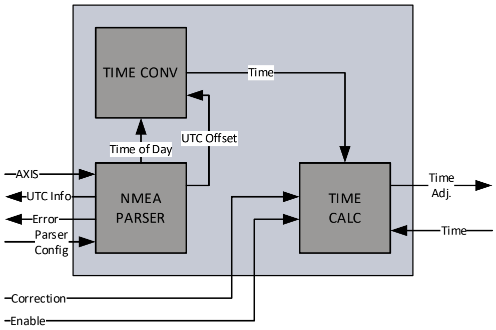

Figure 4: RX Processor

5.2.1.2 Entity Description

NMEA Parser

This module parses all incoming NMEA frames. It extracts the time in case of NMEA from GxZDA or GxRMC frames in case of UBX from NAV_TIME_UTC, checks if the data is valid if GxRMC or UBX is used and checks the CRC. The time is converted from ASCII decimal values to binary values for NMEA. No local time offset is used if GxZDA is used. After extraction the UTC time is in the format hh:mm:ss dd:mm:yyyy which will be passed to the time converter for conversion. In case of UBX NAV_TIME_LS support also the current UTC offset is passed to the time convertion to get from UTC to TAI.

Time Converter

This module converts the time from the Time of Day format: hh:mm:ss dd:mm:yyyy into seconds since midnight 1.1.1970. It loops over the years, months and days taking the leap years into account and finally adds the seconds of the hours, minutes and seconds. If UBX is used also the UTC corrections (in case of UBX) are taken into account. After this conversion a final correction is done if the received second is for the past second or next second. Then this timestamp is passed to the time calculation module.

Time Calculation

This module calculates the reference second by adding or subtracting additional seconds from the Correction register to the received timestamp. It then checks that at least two messages were received before starting to correct the clock value. It waits until the local clock reaches the second boundary and sets the new time if the second part of the local time was different than expected. If the second part of the time is as expected, no correction is done.

5.2.1.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond
Sim_Gen	-	boolean	1	If in Testbench simulation mode
RX Processor				
GpsSupport_Gen	-	boolean	1	Support for GPS (GPxxx) NMEA messages
GlonassSupport_Gen	-	boolean	1	Support for GLONASS (GPxxx) NMEA messages
GalileoSupport_Gen	-	boolean	1	Support for GALI-LEO (GPxxx) NMEA messages
BeidouSupport_Gen	-	boolean	1	Support for BEIDOU (GPxxx) NMEA messages
CombinedGnss Support_Gen	-	boolean	1	Support for Com-bined (GPxxx) NMEA messages
AllGnssSupport_Gen	-	boolean	1	Support for any GNSS identifier
GxZdaMessage Support_Gen	-	boolean	1	Support for GxZDA Messages
GxRmcMessage	-	boolean	1	Support for GxRMC

Support_Gen				Message
NmeaSupport_Gen	-	boolean	1	Support for NMEA Protocol
UbxNavTimeLs MessageSupport_Gen	-	boolean	1	Support for UBX_NAV_TIME_LS Messages: true = supported, false = not support-ed
UbxNavTimeUtc MessageSupport_Gen	-	boolean	1	Support for UBX_NAV_TIME_UTC Message: true = supported, false = not support-ed
UbxSupport_Gen	-	boolean	1	Support for UBX Protocol
NmeaCorrection_Gen	-	natural	1	NMEA correction in seconds for when the message arrive to the next second overflow.
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Timer				
Timer1ms_EvtIn	in	std_logic	1	Millisecond timer adjusted with the Clock
Time of Day Error Output				
Tod_ErrOut	out	std_logic_vector	2	Marks a parser error
Parser Config Input				
TodParser Config_DatIn	in	Tod_Parser Config_Type	1	Parser COnfigura-tion
UTC Info Output				
TodUtcInfo_DatOut	out	Tod_UtcInfo_Type	1	UTC Information
Enable Input				
Enable_EnaIn	in	std_logic	1	Enables the correc-

				tion
Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Adjusted Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted Clock Time valid
Axi Input				
AxisValid_ValIn	in	std_logic	1	AXI Stream frame input
AxisReady_ValOut	out	std_logic	1	
AxisData_DatIn	in	std_logic_vector	8	
AxisStrobe_ValIn	in	std_logic_vector	1	
AxisKeep_ValIn	in	std_logic_vector	1	
AxisLast_ValIn	in	std_logic	1	
AxisUser_DatIn	in	std_logic_vector	2	
Time of Day Correction Input				
TodCorrection_DatIn	in	Clk_Time_Type	1	Additional correction to convert from UTC to a different time format with an offset
Time Adjustment Output				
TimeAdjustment_DatOut	out	Clk_TimeAdjustment_Type	1	Time to set hard
TimeAdjustment_ValOut	out	std_logic	1	Time valid

Table 13: RX Processor

5.2.2 UART Interface Adapter

5.2.2.1 Entity Block Diagram

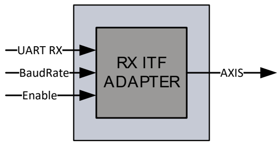

Figure 5: UART Interface Adapter

5.2.2.2 Entity Description

RX Interface Adapter

This module converts the serial UART signal to an AXI stream. It handles the RS232 protocol data stream with one start, eight data (LSB first), one stop and no parity. Data is oversampled and center aligned sampling is done. Metastability flipflops handle the asynchronous input. AXI stream from this module is 8 bit width. It can handle baud rates from 1.2k up to 2m baud. It also has an error detection internally to decide if a byte was valid or not. The receiver has no buffer and only pushes the byte to the next module. The next module has a half bit time on UART to acknowledge the receipt otherwise the byte is dropped. Since the next module can handle byte streams up to 400mbit no bytes will be dropped under normal conditions.

5.2.2.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond
Interface Adapter				
UartBaudRate_Gen	-	natural	1	Default Baudrate encoded: 0 => 1200 1 => 2400 2 => 4800 3 => 9600 4 => 19200

				5 => 38400 6 => 57600 7 => 115200 8 => 230400 9 => 460800 10 => 921600 11 => 1000000 12 => 2000000
UartPolarity_Gen	-	boolean	1	true = normal UART (idle '1') false = inversed
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Enable Input				
Enable_EnaIn	in	std_logic	1	Enables the Uart
UART Error Output				
Uart_ErrOut	err	std_logic	1	UART error detect-ed (wrong baud rate)
UART Input				
Uart_DatIn	in	std_logic	1	UART from the NMEA source
UART Baud Rate Input				
UartBaudRate_DatIn	in	std_logic_vector	4	Baudrate encoded: 0 => 1200 1 => 2400 2 => 4800 3 => 9600 4 => 19200 5 => 38400 6 => 57600 7 => 115200 8 => 230400 9 => 460800 10 => 921600 11 => 1000000 12 => 2000000
UART Polarity Input				

UartPolarity_DatIN	in	std_logic	1	UART polarity true = normal UART (idle '1') false = inversed
Axi Output				
AxisValid_ValOut	out	std_logic	1	AXI Stream frame output
AxisReady_ValIn	in	std_logic	1	
AxisData_DatOut	out	std_logic_vector	8	
AxisStrobe_ValOut	out	std_logic_vector	1	
AxisKeep_ValOut	out	std_logic_vector	1	
AxisLast_ValOut	out	std_logic	1	
AxisUser_DatOut	out	std_logic_vector	2	

Table 14: UART Interface Adapter

5.2.3 Registerset

5.2.3.1 Entity Block Diagram

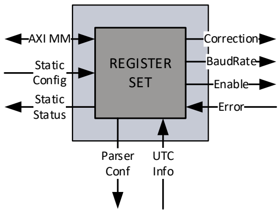

Figure 6: Registerset

5.2.3.2 Entity Description

Register Set

This module is an AXI Light Memory Mapped Slave. It provides access to all registers and allows configuring the TOD Slave Clock. AXI4 Light only supports 32 bit wide data access, no byte enables, no burst, no simultaneous read and writes and no unaligned access. It can be configured to either run in AXI or StaticConfig mode. If in StaticConfig mode, the configuration of the registers is done via signals and can be easily done from within the FPGA without CPU. For each configuration parameter a valid signal is available, the enable signal shall be set last (or simultaneously). To change configuration parameters the clock has to be disabled and enabled again, the correction value can be changed at runtime. If in AXI mode, an AXI Master has to configure the registers with AXI writes to the registers, which is typically done by a CPU. Parameters can in this case also be changed at runtime. There is also a Static Status which is put out as a Vector which contains information which otherwise can also be read via AXI from Registers.

5.2.3.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
Register Set				
UartBaudRate_Gen	-	natural	1	Default Baudrate encoded: 0 => 1200 1 => 2400 2 => 4800

				3 => 9600 4 => 19200 5 => 38400 6 => 57600 7 => 115200 8 => 230400 9 => 460800 10 => 921600 11 => 1000000 12 => 2000000
UartPolarity_Gen	-	boolean	1	true = normal UART (idle '1') false = inversed
GpsSupport_Gen	-	boolean	1	Support for GPS (GPxxx) NMEA messages
GlonassSupport_Gen	-	boolean	1	Support for GLONASS (GPxxx) NMEA messages
GalileoSupport_Gen	-	boolean	1	Support for GALI-LEO (GPxxx) NMEA messages
BeidouSupport_Gen	-	boolean	1	Support for BEIDOU (GPxxx) NMEA messages
CombinedGnss Support_Gen	-	boolean	1	Support for Com-bined (GPxxx) NMEA messages
AllGnssSupport_Gen	-	boolean	1	Support for any GNSS identifier
GxZdaMessage Support_Gen	-	boolean	1	Support for GxZDA Messages
GxRmcMessage Support_Gen	-	boolean	1	Support for GxRMC Message
NmeaSupport_Gen	-	boolean	1	Support for NMEA Protocol
UbxNavTimeLs MessageSupport_Gen	-	boolean	1	Support for UBX_NAV_TIME_LS

				Messages: true = supported, false = not supported
UbxNavTimeUtc MessageSupport_Gen	-	boolean	1	Support for UBX_NAV_TIME_UTC Message: true = supported, false = not supported
UbxSupport_Gen	-	boolean	1	Support for UBX Protocol
StaticConfig_Gen	-	boolean	1	If Static Configuration or AXI is used
AxiAddressRange Low_Gen	-	std_logic_vector	32	AXI Base Address
AxiAddressRange High_Gen	-	std_logic_vector	32	AXI Base Address plus Registerset Size
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Config				
StaticConfig_DatIn	in	Tod_Slave StaticConfig_Type	1	Static Configuration
StaticConfig_ValIn	in	Tod_Slave StaticConfigVal _Type	1	Static Configuration valid
Status				
StaticStatus_DatOut	out	Tod_Slave StaticStatus_Type	1	Static Status
StaticStatus_ValOut	out	Tod_Slave StaticStatusVal _Type	1	Static Status valid
AXI4 Light Slave				
AxiWriteAddrValid _ValIn	in	std_logic	1	Write Address Valid
AxiWriteAddrReady _RdyOut	out	std_logic	1	Write Address

				Ready
AxiWriteAddrAddress _AdrIn	in	std_logic_vector	32	Write Address
AxiWriteAddrProt _DatIn	in	std_logic_vector	3	Write Address Protocol
AxiWriteDataValid _ValIn	in	std_logic	1	Write Data Valid
AxiWriteDataReady _RdyOut	out	std_logic	1	Write Data Ready
AxiWriteDataData _DatIn	in	std_logic_vector	32	Write Data
AxiWriteDataStrobe _DatIn	in	std_logic_vector	4	Write Data Strobe
AxiWriteRespValid _ValOut	out	std_logic	1	Write Response Valid
AxiWriteRespReady _RdyIn	in	std_logic	1	Write Response Ready
AxiWriteResp Response_DatOut	out	std_logic_vector	2	Write Response
AxiReadAddrValid _ValIn	in	std_logic	1	Read Address Valid
AxiReadAddrReady _RdyOut	out	std_logic	1	Read Address Ready
AxiReadAddrAddress _AdrIn	in	std_logic_vector	32	Read Address
AxiReadAddrProt _DatIn	in	std_logic_vector	3	Read Address Protocol
AxiReadDataValid _ValOut	out	std_logic	1	Read Data Valid
AxiReadDataReady _RdyIn	in	std_logic	1	Read Data Ready
AxiReadData Response_DatOut	out	std_logic_vector	2	Read Data
AxiReadDataData _DatOut	out	std_logic_vector	32	Read Data Re-sponse
UART Baud Rate Output				
UartBaud Rate_DatOut	out	std_logic_vector	4	Baudrate encoded: 0 => 1200 1 => 2400 2 => 4800 3 => 9600 4 => 19200 5 => 38400 6 => 57600 7 => 115200

				8 => 230400 9 => 460800 10 => 921600 11 => 1000000 12 => 2000000
UART Polarity Output				
UartPolarity_DatOut	out	std_logic	1	UART polarity true = normal UART (idle '1') false = inversed
Correction Output				
TodCorrection_DatOut	out	Clk_Time_Type	1	Additional correction to the received UTC time
UTC Info Input				
TodUtcInfo_DatOut	in	Tod_UtcInfo_Type	1	UTC Information
Parser Config Output				
TodParser Config_DatIn	out	Tod_Parser Config_Type	1	Parser Configuration
Error Input				
Tod_ErrIn	in	std_logic_vector	3	An error happened
Enable Output				
TodSlave Enable_DatOut	out	std_logic	1	Enable TOD Slave Clock

Table 15: Registerset

5.3 Configuration example

In both cases the enabling of the core shall be done last, after or together with the configuration.

5.3.1 Static Configuration

```
constant TodStaticConfigSlave_Con : Tod_SlaveStaticConfig_Type := (
  Protocol              => '0', -- NMEA
  Gnss                  => std_logic_vector(to_unsigned(Tod_SlaveGnss_AllGnss_Con,4)),
  DisableMessages       => x"01", -- no ZDA
  Polarity              => '1',
  Correction            => (
    Second              => x"00000025", -- UTC 37 leap seconds
    Nanosecond          => (others => '0'), -- no nanoseconds
    Fraction            => (others => '0'), -- no fractions
    Sign                => '0', -- UTC correct in positive
    TimeJump            => '0'), -- no
  UartBaudRate          => x"7"—115200 (same enum as with generic)
);

constant TodStaticConfigValSlave_Con : Tod_SlaveStaticConfigVal_Type := (
  Enable_Val            => '1'
);
```

Figure 7: Static Configuration

The UartBaudRate, Protocol and Gnss has to be configured before enabling; changes on this value only have an effect on a transition from disabled to enabled. The Correction value can be set at runtime and has immediate effect; only the seconds and sign part of the correction are used.

5.3.2 AXI Configuration

The following code is a simplified pseudocode from the testbench: The base address of the TOD Slave Clock is 0x10000000.

```
-- TOD SLAVE
-- Config
-- correction of plus 37 second to convert UTC to TAI for NMEA
AXI WRITE 10000010 00000025
-- change baud rate to 115200
AXI WRITE 10000020 00000007

-- enable TOD Slave, NMEA, no ZDA and all GNSS
AXI WRITE 10000000 00010001
```

Figure 8: AXI Configuration

In the example the clock gets a correction of 36 seconds to correct UTC to TAI and the baud rate is set to 115200 baud/s

5.4 Clocking and Reset Concept

5.4.1 Clocking

To keep the design as robust and simple as possible, the whole TOD Slave Clock, including the Counter Clock and all other cores from NetTimeLogic are run in one clock domain. This is considered to be the system clock. Per default this clock is 50MHz. Where possible also the interfaces are run synchronous to this clock. For clock domain crossing asynchronous fifos with gray counters or message patterns with meta-stability flip-flops are used. Clock domain crossings for the AXI interface is moved from the AXI slave to the AXI interconnect.

Clock	Frequency	Description
System		
System Clock	50MHz (Default)	System clock where the Tod Slave runs on as well as the counter clock etc.
UART Interface		
UART RX	1.2 kHz – 1MHz	No clock, asynchronous data signal, external receive clock from the UART. Must be defined for the core prior to use of the interface not all frequencies apply.
AXI Interface		
AXI Clock	50MHz (Default)	Internal AXI bus clock, same as the system clock

Table 16: Clocks

5.4.2 Reset

In connection with the clocks, there is a reset signal for each clock domain. All resets are active low. All resets can be asynchronously set and shall be synchronously released with the corresponding clock domain. All resets shall be asserted for the first couple (around 8) clock cycles. All resets shall be set simultaneously and released simultaneously to avoid overflow conditions in the core. See the reference designs top file for an example of how the reset shall be handled.

Reset	Polarity	Description
System		
System Reset	Active low	Asynchronous set, synchronous release

		with the system clock
AXI Interface		
AXI Reset	Active low	Asynchronous set, synchronous release with the AXI clock, which is the same as the system clock

Table 17: Resets

6 Resource Usage

Since the FPGA Architecture between vendors and FPGA families differ there is a split up into the two major FPGA vendors.

6.1 Altera (Cyclone V)

Configuration	FFs	LUTs	BRAMs	DSPs
Minimal (Static Config, NMEA and RMC only)	570	1922	0	0
Maximal (AXI, NMEA, UBX, all GNSS and all Messages)	889	2724	0	0

Table 18: Resource Usage Altera

6.2 Xilinx (Artix 7)

Configuration	FFs	LUTs	BRAMs	DSPs
Minimal (Static Config, NMEA and RMC only)	561	1210	0	0
Maximal (AXI, NMEA, UBX, all GNSS and all Messages)	934	1735	0	0

Table 19: Resource Usage Xilinx

7 Delivery Structure

```
AXI                                  -- AXI library folder
 |-Library                           -- AXI library component sources
 |-Package                           -- AXI library package sources

CLK                                  -- CLK library folder
|-Library                            -- CLK library component sources
|-Package                            -- CLK library package sources

COMMON                               -- COMMON library folder
 |-Library                           -- COMMON library component sources
 |-Package                           -- COMMON library package sources

PPS                                  -- PPS library folder
 |-Package                           -- PPS library package sources

SIM                                  -- SIM library folder
 |-Doc                               -- SIM library command documentation
 |-Package                           -- SIM library package sources
 |-Testbench                         -- SIM library testbench template sources
 |-Tools                             -- SIM simulation tools

TOD                                  -- TOD library folder
 |-Core                              -- TOD library cores
 |-Doc                               -- TOD library cores documentations
 |-Library                           -- TOD library component sources
 |-Package                           -- TOD library package sources
 |-Refdesign                         -- TOD library cores reference designs
 |-Testbench                         -- TOD library cores testbench sources and sim/log
```

8 Testbench

The Tod Slave testbench consist of 3 parse/port types: AXI, CLK and TOD.
The TOD transmit port takes the CLK port time as reference and send the timestamp generated by this clock as NMEA messages. The TOD receiver port takes the time of the Clock instance as reference and the NMEA data stream from the TOD transmit port. Once the clock is synchronized the CLK port and Clock generated time should be the same.. In addition for configuration and result checks an AXI read and write port is used in the testbench and for accessing more than one AXI slave also an AXI interconnect is required.

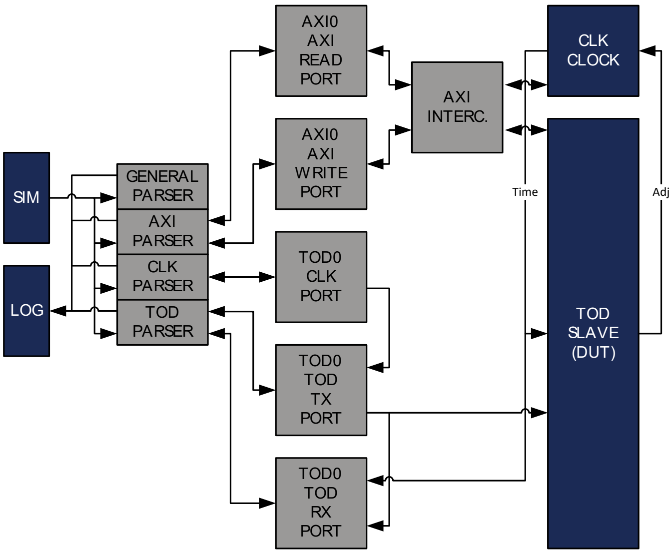

Figure 9: Testbench Framework

For more information on the testbench framework check the Sim_ReferenceManual documentation.

With the Sim parameter set the time base for timeouts are divided by 1000 to 100000 to speed up simulation time.

8.1 Run Testbench

1. Run the general script first

```
source XXX/SIM/Tools/source_with_args.tcl
```

2. Start the testbench with all test cases

```
src XXX/TOD/Testbench/Core/TodSlave/Script/run_Tod_Slave_Tb.tcl
```

3. Check the log file LogFile1.txt in the XXX/TOD/Testbench/Core/TodSlave/Log/ folder for simulation results.

9 Reference Designs

The TOD Slave reference design contains a PLL to generate all necessary clocks (cores are run at 50 MHz), an instance of the TOD Slave Clock IP core and an instance of the Adjustable Counter Clock IP core (needs to be purchased separately). Optionally it also contains an instance of a PPS Slave Clock IP core and an instance of a PPS Master Clock IP core (both have to be purchased separately). To instantiate the optional IP cores, change the corresponding generics (PpsMasterAvailable_Gen, PpsSlaveAvailable_Gen) to true via the tool specific wizards.
The Reference Design with a PPS and TOD Slave Clock is intended to be connected to a GPS receiver with a baudrate of 9600. If another baud rate shall be used this can be set via the Static Configuration. The absolute second is corrected via the TOD Slave Clock and the Phase and Frequency is corrected via the PPS Slave Clock. The PPS Master Clock is used to create a PPS output which is compensated for the output delay and has a configurable duty cycle, if not available an uncompensated PPS is directly generated out of the MSB of the Time.
All generics can be adapted to the specific needs.

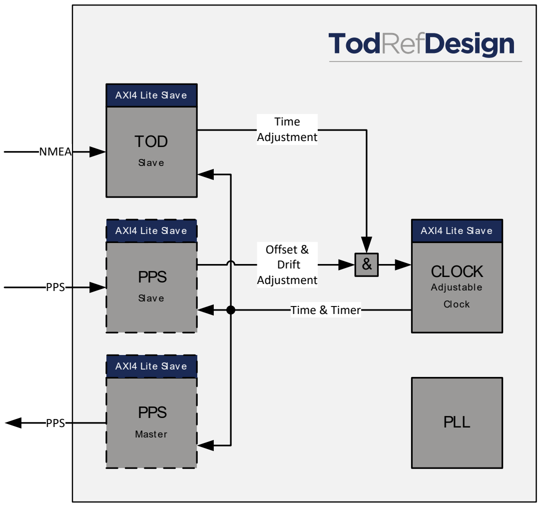

Figure 10: Reference Design

9.1 Altera: Terasic SocKit

The SocKit board is an FPGA board from Terasic Inc. with a Cyclone V SoC FPGA from Altera. (http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=205&No=816)

1. Open Quartus 16.x

2. Open Project /TOD/Refdesign/Altera/SocKit/TodSlave/TodSlave.qpf
3. If the optional cores PPS Slave and PPS Master Clock are available add the files from the corresponding folders (PPS/Core, PPS/Library, PPS/Package and CLK/Library)
4. Change the generics (PpsMasterAvailable_Gen, PpsSlaveAvailable_Gen) in Quartus (in the settings menu, not in VHDL) to true for the optional cores that are available.
5. Rerun implementation
6. Download to FPGA via JTAG

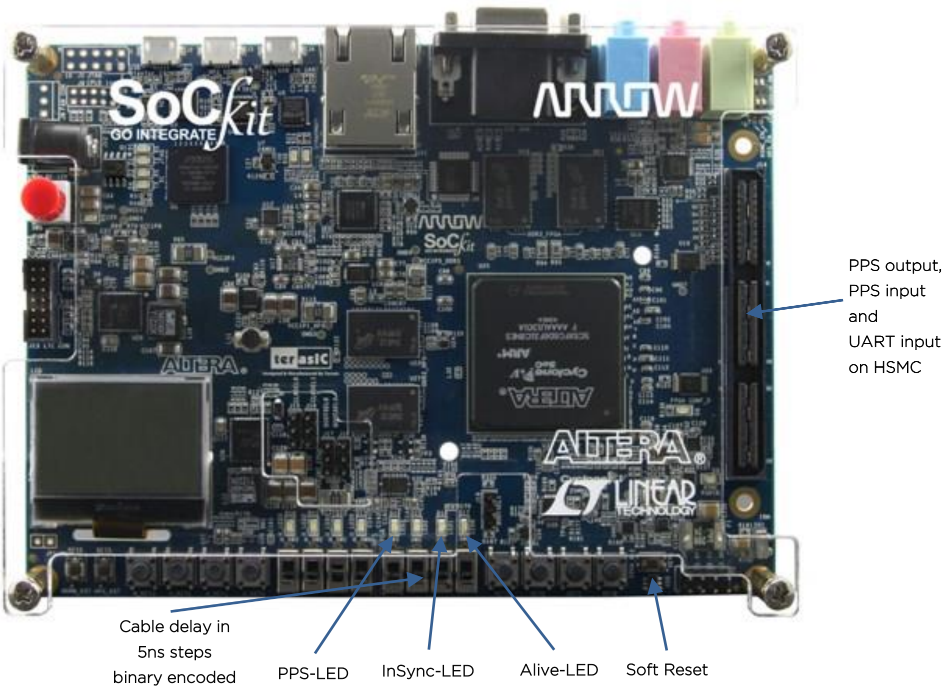

Figure 11: SocKit (source Terasic Inc)

For the ports on the HSMC connector the GPIO to HSMC adapter from Terasic Inc. was used.

9.2 Xilinx: Digilent Arty

The Arty board is an FPGA board from Digilent Inc. with an Artix7 FPGA from Xilinx. (http://store.digilentinc.com/arty-board-artix-7-fpga-development-board-for-makers-and-hobbyists/

1. Open Vivado 2017.4
2. Run TCL script /TOD/Refdesign/Xilinx/Arty/TodSlave/TodSlave.tcl

a. This has to be run only the first time and will create a new Vivado Project
3. If the project has been created before open the project and do not rerun the project TCL
4. If the optional cores PPS Slave and PPS Master Clock are available add the files from the corresponding folders (PPS/Core, PPS/Library, PPS/Package and CLK/Library) to the corresponding Libraries (PpsLib and ClkLib).
5. Change the generics (PpsMasterAvailable_Gen, PpsSlaveAvailable_Gen) in Vivado (in the settings menu, not in VHDL) to true for the optional cores that are available.
6. Rerun implementation
7. Download to FPGA via JTAG

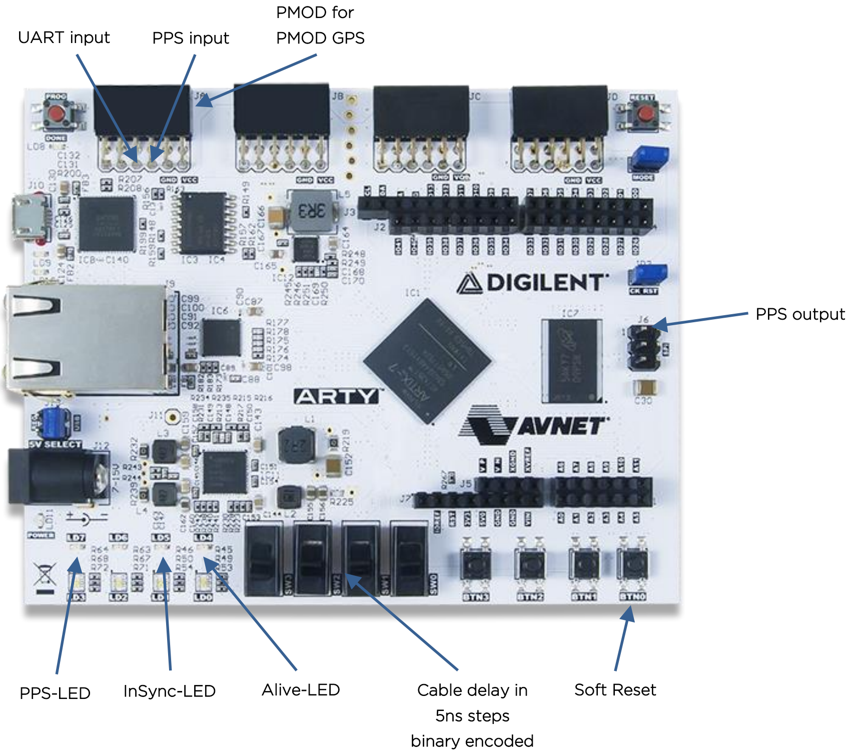

Figure 12: Arty (source Digilent Inc)

9.2.1 GPS receiver

As stated in earlier chapters the NMEA source often is a GPS receiver.
The GPS receiver used in the reference design is a PMOD GPS receiver from Digilent Inc. (http://store.digilentinc.com/pmodgps-gps-receiver/) which can be directly connected to the upper row of PMOD JA on the Arty. This receiver requires quite direct view to the sky, so an extension cable might be needed.

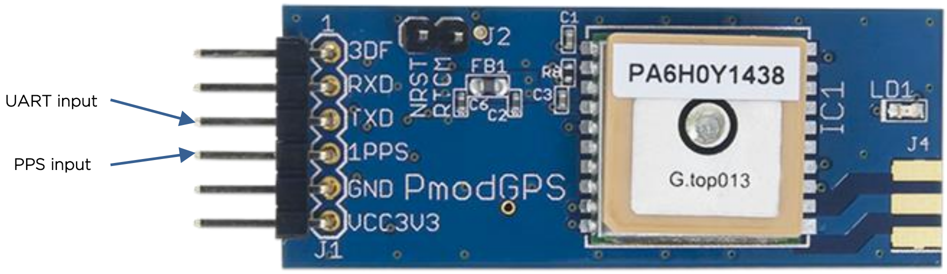

Figure 13: PMOD GPS (source Digilent Inc)

A List of tables

B List of figures