Software Manual – Programming and Integration

SEA 9405/9410/9414

GPS Modules for CompactRIO™

Science & Engineering
Applications Datentechnik
GmbH

Doc. No.: HB/SEA 94xx SoftwareManual/3.1.a/Mar-2019

Subject to modifications.

S.E.A. Datentechnik GmbH takes no responsibility for damage arising out of or related to this document or the information contained in it.

Product and company names listed are trademarks or trade names of their respective companies.

Address:

S.E.A. Datentechnik GmbH
Muelheimer Strasse 7
53840 Troisdorf
Germany

Phone: +49 2241 12737-0
Fax: +49 2241 12737-14

For support please contact the support email address:

techsupport@sea-gmbh.com

Contents

1 Change Notes....4
2 Getting Started....5
2.1 General....5
2.2 End User License Agreement (EULA)....5
2.3 Symbols, Notations and Nomenclature....5
3 Installation....7
3.1 Hardware Requirements....7
3.2 Software Requirements....7
4 Quick Start....8
5 Description of Functionality....16
5.1 Platforms....16
5.2 SEA 94xx Functionality....16
5.3 Power Up Behaviour....16
6 Programming....17
6.1 Migration....17
6.2 Examples....17
6.3 Basic Concepts....17
6.4 Application Programming Interface (API)....17
6.4.1 I/O Nodes....18
6.4.2 I/O Property Nodes....18
6.4.3 I/O Method Nodes....21
6.4.3.1 Configure Message....21
6.4.3.2 Configure Rate....21
6.4.3.3 Configure SBAS....21
6.4.3.4 Configure Timepulse....22
6.4.3.5 Read RAW....23
6.4.3.6 Reset....23
6.4.3.7 Wait For POS....23
6.4.3.8 Wait For STATUS....23
6.4.3.9 Wait For TIME....24
6.4.3.10 Wait For Timepulse....24
6.5 Error Codes....24
7 Deployment....26
8 Trouble Shooting....27
A Figures....28
B Tables....29

1 Change Notes

#	Description	Changes
1	1.0	Initial release for new LabVIEW driver architecture (Module Development Kit 2.0, MDK2).
2	2.0	1. API improvements 2. Support for new GPS modules SEA 9405, SEA 9410 and SEA 9414
3	3.0	1. Main Update release with a new set of driver functions. Notes on SEA 9404 removed document.
4	3.1.a	1. Support for NI CompactRIO 904x targets added. Oldest supported LabVIEW version is now 2017.

2 Getting Started

2.1 General

Before starting to work with the SEA 9405/9410/9414 module please read this document and the complete hardware manual carefully. If there are any questions about operating the module or if any term is not understood, please contact the vendor before using the module. Please check the download area on the S.E.A. website `http://www.sea-gmbh.com` for updates of the manuals.

Refer to the hardware manual for details on operation instructions, safety guidelines and specifications for the SEA 9405/9410/9414 module.

Refer to the appropriate National Instruments™ documentation for details on National Instruments hardware.

We believe that all information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event of technical or typographical errors, we reserve the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult the vendor if errors are suspected.

2.2 End User License Agreement (EULA)

Before operating the SEA 9405/9410/9414 module and the provided software you have to agree to the terms and conditions (EULA). This agreement is part of the software installation procedure. If you do NOT agree you can send back the hardware and software package within a period of two weeks after delivery. In this case S.E.A. will refund the product price and shipping costs.

In addition, the terms and conditions are available through the LabVIEW menu after installation (Tools › SEA › SEA 94xx › Legal Information).

2.3 Symbols, Notations and Nomenclature

To improve clarity specific structuring elements (symbols) are used which have the following meaning:

Symbol	Meaning
Names	Specific names are printed using an *italic font*
[Text]	Place holders are marked by squared brackets
	Locations (paths, menus, URLs...) are printed using a `courier font`
	The yellow sign highlights important notes and warnings
	The blue mark highlights tips
	Reference to other documents

The notation SEA 94xx is a short cut for a family of modules. If you read SEA 94xx you can substitute it with the name of your particular type (SEA 9405/9410/9414). If any information contained in this manual refers exclusively to a particular module type, then this particular module name is used instead of SEA 94xx.

For a better understanding, a short list of used terms will be given:

Name	Meaning
MDK2.1	**M**odule **D**evelopment **K**it, version **2.1**. MDK2 is a framework by National Instruments for developing 3rd party cRIO modules.

GPS	**G**lobal **P**ositiong **S**ystem
NMEA	Is a standard protocol used by GPS receivers to transmit telemetry data (more precisely NMEA-0183) from **N**ational **M**arine **E**lectronics **A**ssociation.
SBAS	**S**atellite **B**ased **A**ugmentation **S**ystem
API	**A**pplication **P**rogramming **I**nterface
FPGA	**F**ield **P**rogrammable **G**ate **A**rray

3 Installation

There is a single driver software for the module SEA 9405/9410/9414. The driver software can be directly downloaded and installed through the JKI VI Package Manager. Additionally the driver software can be downloaded either from the NI Tools Network or from S.E.A. download site and installed separately.

Due to continuous improvements the required software is not enclosed with the module hardware when shipped. The latest version of the driver software can be downloaded from the S.E.A. web site:

http://www.sea-gmbh.com

On the main page select *Support* from the main menu. On the support page select *Downloads* from the local menu and click on the *Download Area* button. On the download site select your module type from the *Hardware Products* category.

The driver software is to be installed on a development PC only (refer to the hardware requirement section below). In order to install the software package double-click or open the .vip file inside the VI Package Manager and follow the instructions on the screen. This procedure installs the driver including application programming interface (API), tools, examples and all related documentation. Further resources (if available) like application notes, drawings etc. can be downloaded separately from the location as stated above.

3.1 Hardware Requirements

The SEA 94xx module requires at least a PC to program an application. In order to run the application a NI CompactRIO Real-Time controller with the SEA module inserted in any of the chassis slots is required.

The driver software requires a PC for programming (development PC) and a NI CompactRIO with FPGA chassis to run the application (run-time device).

3.2 Software Requirements

The development PC and the run-time device are described in the hardware requirements section above.

Software requirements for the development PC:

- JKI VI Package Manager (VIPM) - latest available version
- NI LabVIEW Development environment 2017 or higher
- NI LabVIEW Real-Time 2017 or higher
- NI LabVIEW FPGA 2017 or higher + suitable Xilinx tools

Software requirements for the run-time device (NI CompactRIO):

- NI LabVIEW Run-Time Engine / LabVIEW Run-Time Engine for Real-Time targets (standard installation)

SEA 94xx driver software (module support) only operates in the FPGA interface (programming) mode of a NI CompactRIO system. The *Scan Interface* and *DAQ* modes are not supported at present.

4 Quick Start

This section demonstrates how easy the SEA 94xx module can be integrated into an existing NI CompactRIO system.

Refer to the hardware manual for proper installation of the hardware.

The present (Q1/2019) the cRIO modules do not auto-discover in cRIO-904x chassis. The work-around is to manually add the modules to the LabVIEW project. NI works currently on a solution. The NI's Corrective Action Request (CAR) is #687418.

The present (Q1/2019) the cRIO modules do not auto-discover in NI-9144 and NI-9145 EtherCAT chassis. This issue only affects chassis running latest chassis firmware 16.0 and 16.1 which comes with NI-Industrial Communications for EtherCAT 16.0 and 16.1 driver. NI works currently on a solution. The NI's Corrective Action Request (CAR) #637095.

Follow the steps below to create a simple application using the SEA 94xx module (for this tutorial the SEA 9405 and NI cRIO-9040 has been used exemplary) and run it on a NI CompactRIO system:

1. Ensure that you meet the hardware and software requirements listed in the previous chapter and the required software is installed on your PC and on the NI CompactRIO system.

2. Insert the SEA 94xx module into your NI CompactRIO system in a slot of your choice (for this tutorial slot 1 has been used). Connect the GPS antenna to the module and place the antenna outside buildings or next to a window.

3. Connect your NI CompactRIO system to the development PC via Ethernet cable and ensure a correct IP configuration of participating devices. Power up all devices.

4. Install the SEA 94xx driver software. If you have the SEA 94xx setup file locally available just double-click at it and start the installation through VIPM. If you don't have the SEA 94xx setup file locally available search for 94xx in VIPM and install the SEA 94xx driver software directly from here.

5. Detect/add the NI CompactRIO system in NI-MAX Network environment (Netzwerkumgebung). Ensure that all used SEA 94xx modules are configured to 'LabVIEW FPGA', refer to the screen shot below. Close NI-MAX afterwards.

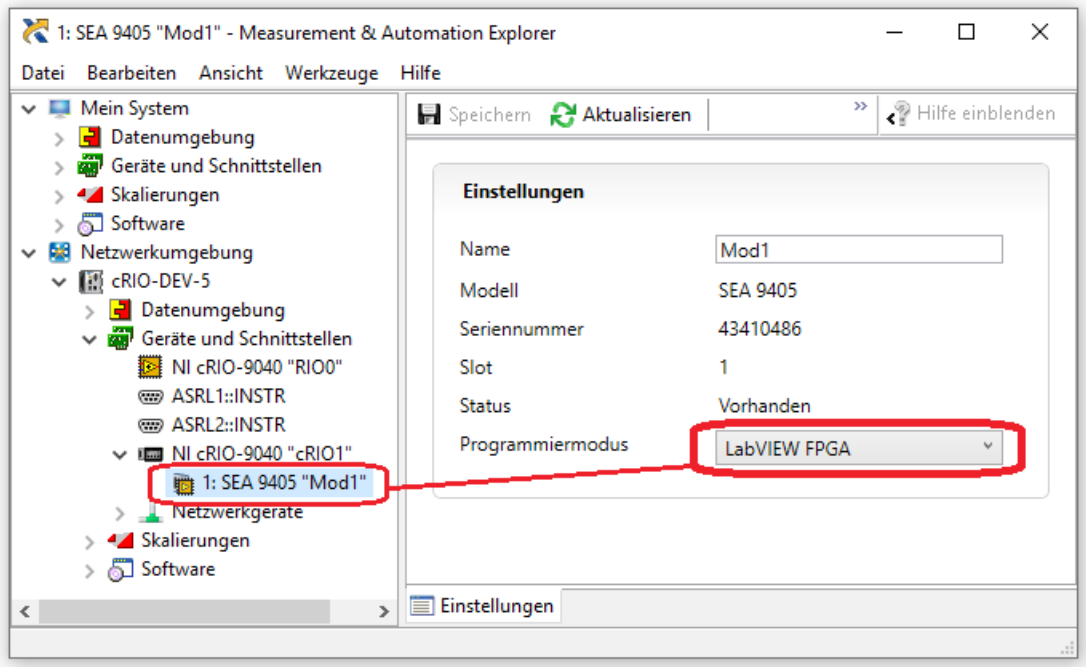

Fig. 1: Quick Start – NI-MAX configuration

6. Start LabVIEW and create a new, blank project.
7. The project explorer window appears. In this window select the uppermost item in the tree (*Project: Untitled Project X.lvproj*) and select *New → Targets and Devices* right-clicking on it, like shown below:

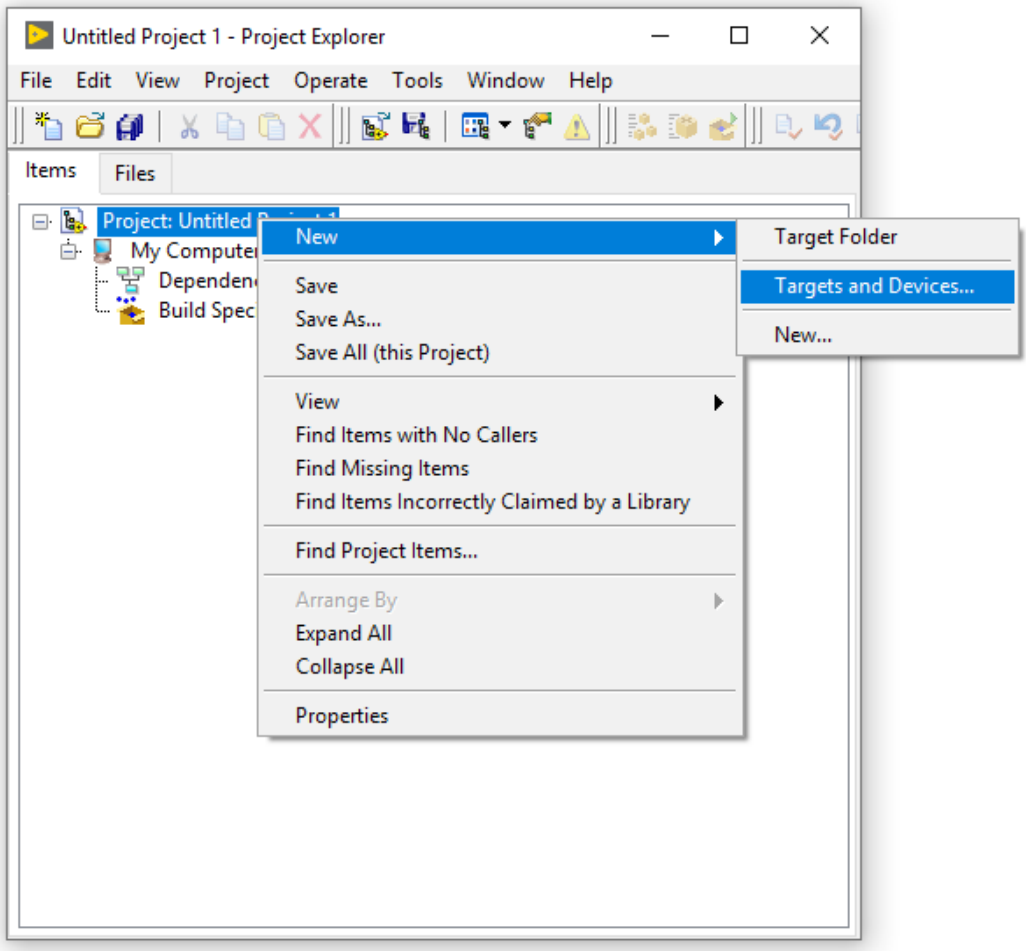

Fig. 2: Quick Start – Add new target

8. Select your NI CompactRIO system from the list of available targets or add it manually. The chassis, FPGA target and cRIO modules are detected automatically. If automatic detection fails please add the system components (chassis, FPGA target, cRIO modules) manually to the LabVIEW project.
9. Ensure that the chassis is configured to 'LabVIEW FPGA Interface' RIO programming mode.

10. The configuration is completed. After a successful discovering the project explorer window should look similar to figure 3.

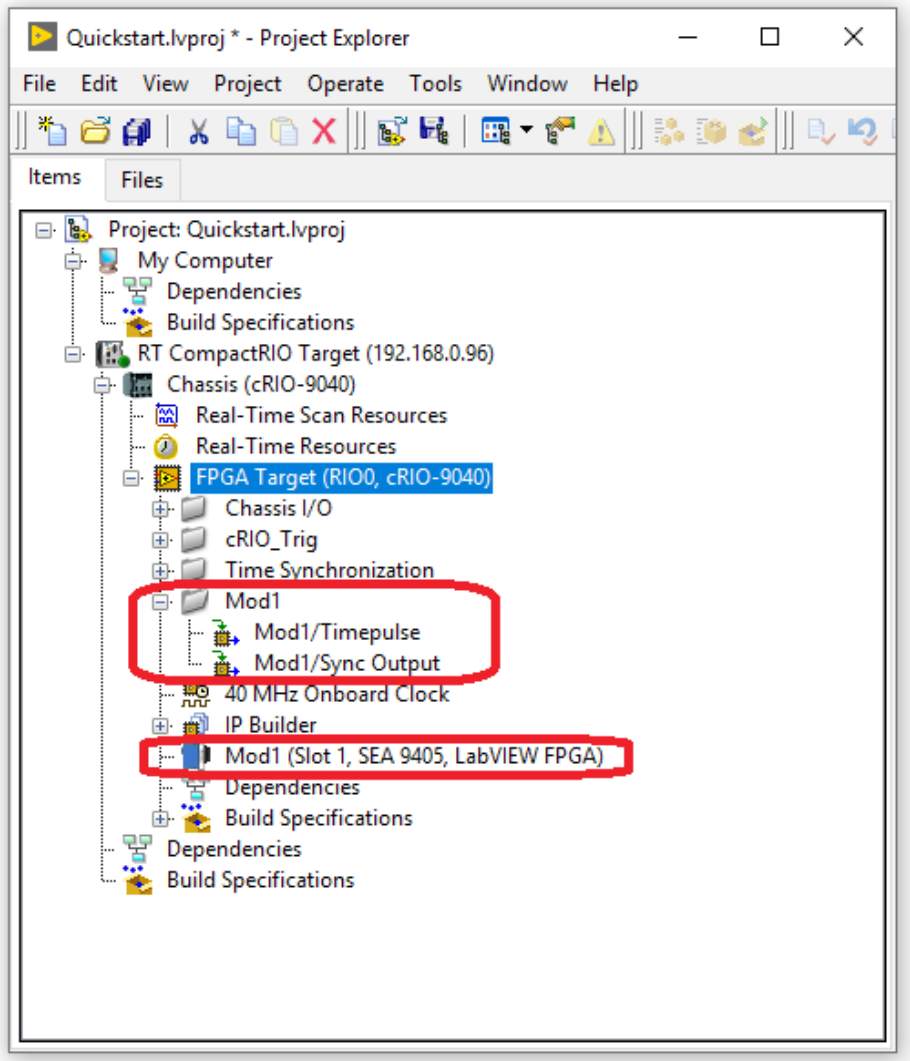

Fig. 3: Quick Start – Project explorer after completing the configuration

The project is now fully configured and the SEA 94xx (here SEA 9405) resources can be used in the user FPGA application.

You can continue from here to learn how to implement a simple FPGA application using the SEA 94xx module.

11. Create a new FPGA VI in the project explorer window. For this right-click on the *FPGA Target (RIOo...)* and select *New –› VI*. Refer to figure 4 below:

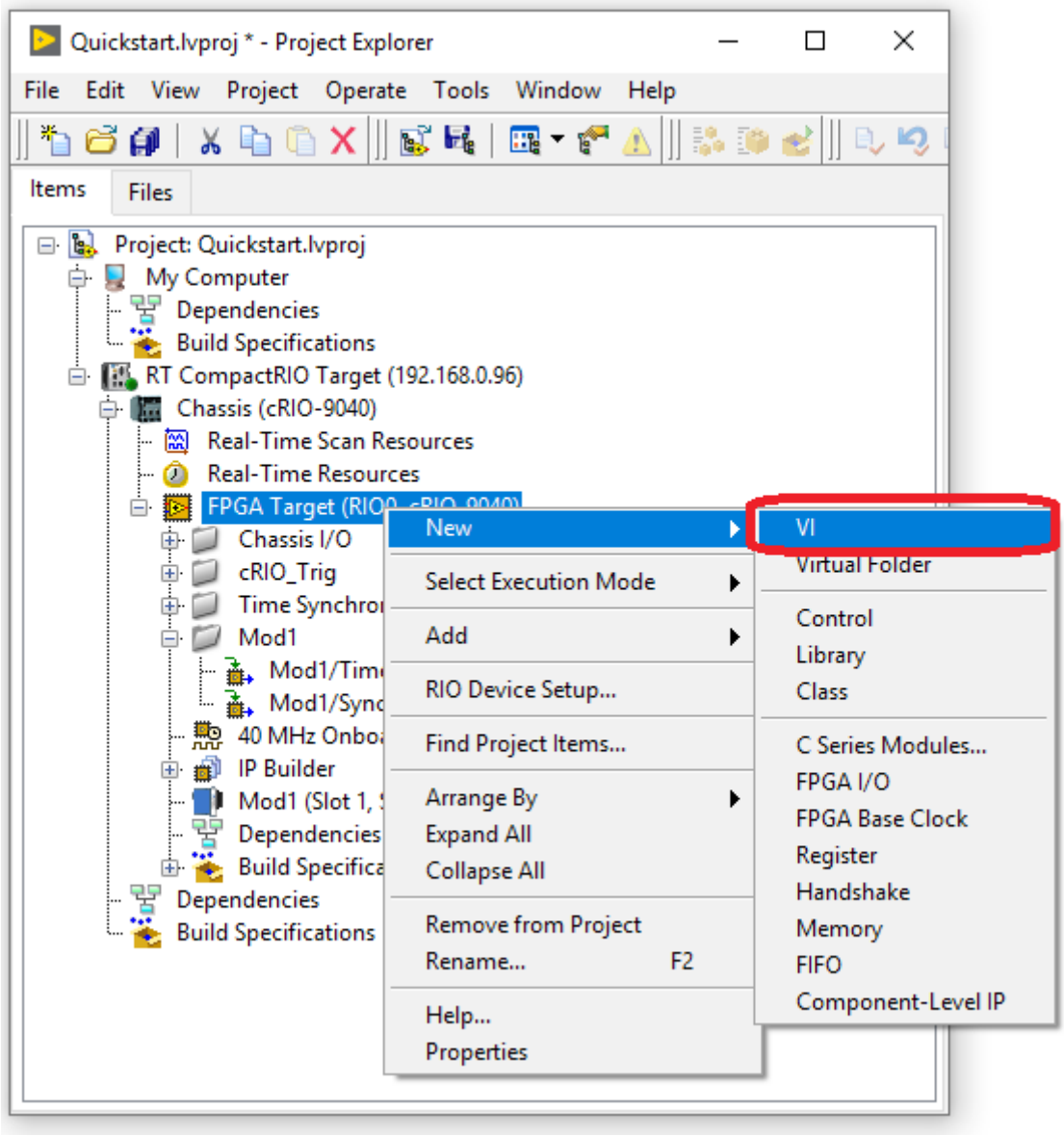

Fig. 4: Quick Start – Add new FPGA VI

12. Open the block diagram of the VI just created and insert a *Flat Sequence Structure Frame*.

13. Place a method node from the functions palette (*FPGA I/O –› I/O Method*) into the *Flat Sequence Structure Frame* (Fig. 5). Afterwards, select e.g. the item *Mod1* (exact naming depends on the cRIO slot used) as shown in figure 5.

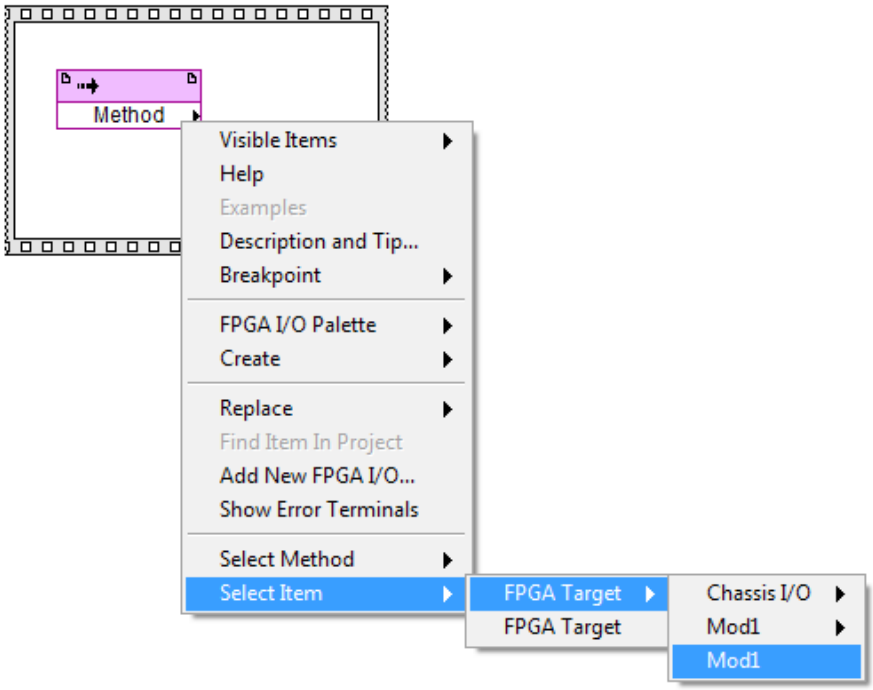

Fig. 5: Quick Start – Select module

14. Select the method *Wait For POS* (Fig. 6).

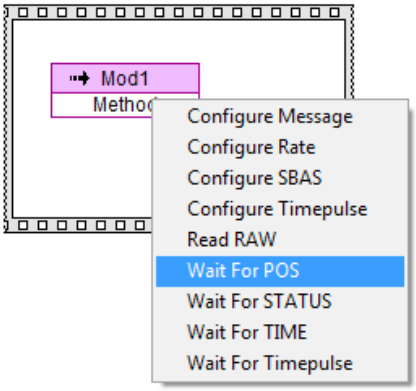

Fig. 6: Quick Start – Create method node

15. Create a *Timeout* input value of `41000000` and create a *Timed Out* indicator (Fig. 7).

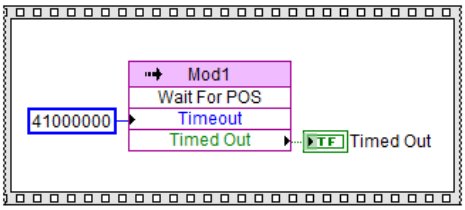

Fig. 7: Quick Start – Wire method node

16. Place a property node from the functions palette (*FPGA I/O –› I/O Property*) into a subsequent Frame (Fig. 8). Afterwards, select the item *Mod1* (exact naming depends on the cRIO slot used) as shown in figure 8.

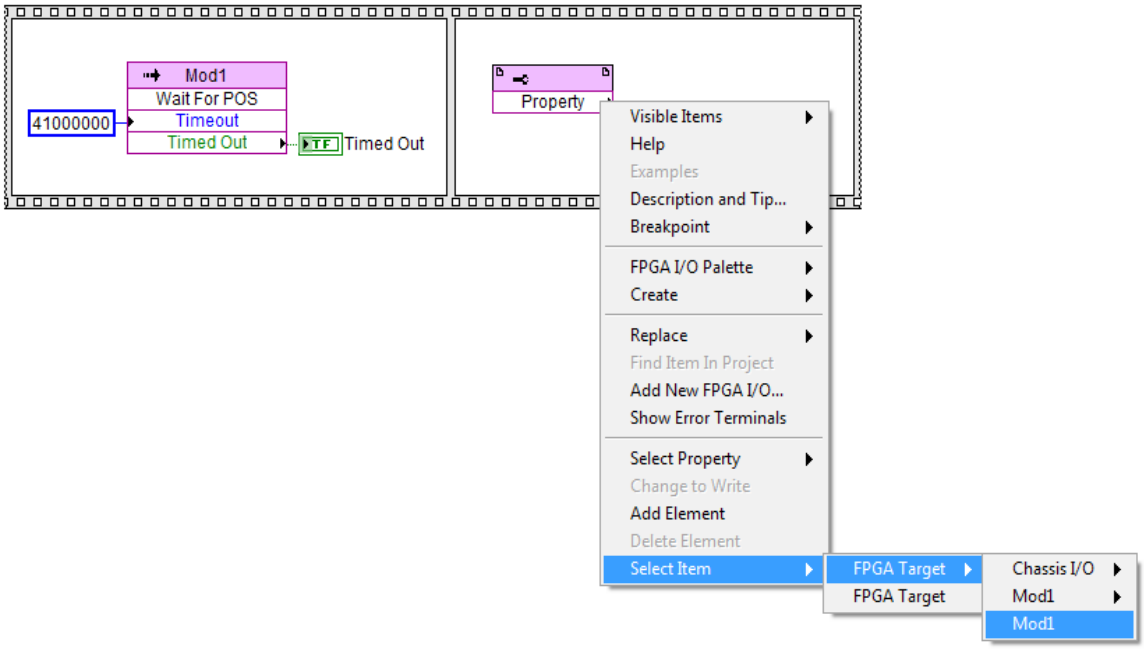

Fig. 8: Quick Start – Create property node

17. Finally select the property to be executed from the list of available property for the selected item. For this tutorial, select the *POS Latitude* property as shown in figure 9.

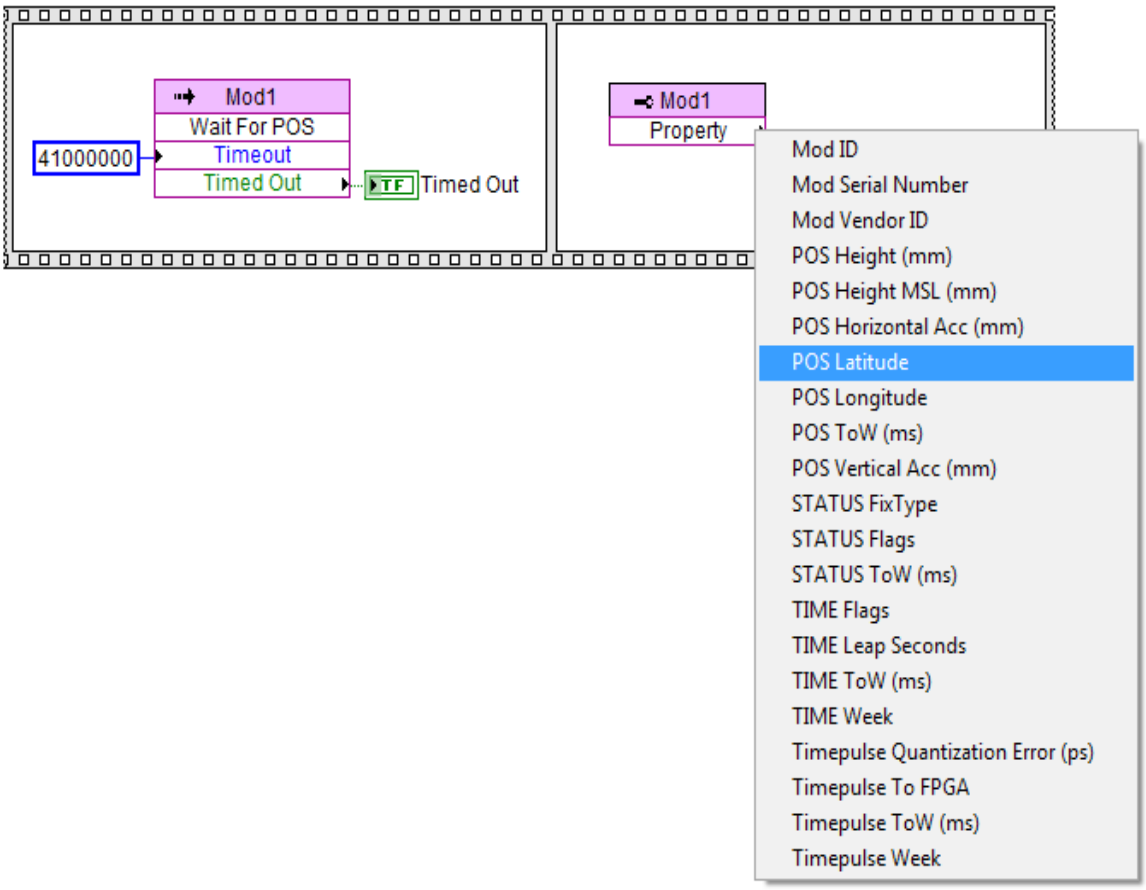

Fig. 9: Quick Start – Select property

18. Optionally repeat step 17 to add further property nodes. Complete the example placing indicators for the nodes as well as a while loop to ensure that the nodes are read continuously. The final code may look like follows:

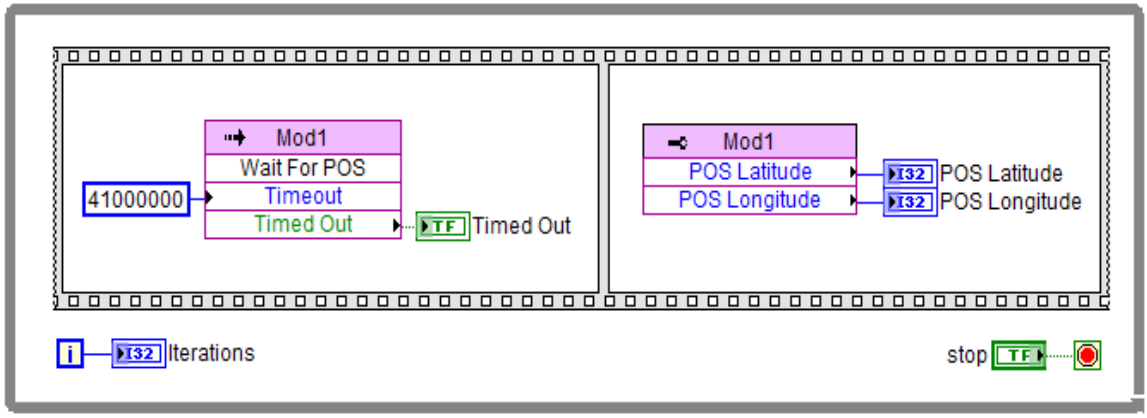

Fig. 10: Quick Start – Final block diagram

Note: The implementation of GPS data reading usually consists of two steps. In a first step, the module is ordered to wait until a new data (here navigation fix message) is available or the timeout occurs. In a second step, the associated[1] (same identifier, here POS) GPS data is read.

19. Save the created FPGA VI, create a build specification and compile it.
20. The indicators will display the current latitude and longitude in degrees (scaled by a factor `1e7`) of the GPS receiver. Note: it may take some time (up to some minutes) until the correct position values are displayed, because the GPS receiver needs to collect data from at least 4 satellites until the position values become valid.

All functions are accessible via FPGA nodes. For a complete list of function nodes please refer to chapter 6 Programming, later in this document.

1 For more details refer to chapter 6 Programming.

5 Description of Functionality

5.1 Platforms

The SEA 94xx modules are cRIO compatible modules that can be used in a wide range of carriers from *National Instruments*. All NI CompactRIO systems with a programmable (FPGA) backplane are currently supported. SEA 94xx modules are not supported in systems without a FPGA programmable backplane like NI CompactDAQ.

5.2 SEA 94xx Functionality

The SEA 94xx cRIO modules feature the reception of GPS telemetry data (basically global position and precise time information) and provide these data to be used directly within the FPGA or as a data steam within the RT target.

The SEA 94xx cRIO module is suitable for a wide variety of applications, including position monitoring absolute timing and synchronization.

The SEA 94xx cRIO modules offer:

- Direct access to geo-positional and timing data in FPGA
- Full access to GPS data stream in RT
- Configuration for geo-positional and timing data in FPGA
- Configurable *Timepulse* signal in FPGA
- Customizable *Sync Output* signal in FPGA
- Up to 4 (10) Hz message update rate
- GPS enhancements: SBAS

All functions are accessible as nodes within the FPGA programming target. Only one function can be executed at a time, which means that individual functions can be only executed subsequently.

The user has to ensure that two functions are not executed at the same time, as this may lead to malfunction.

5.3 Power Up Behaviour

At power up, a hardware reset of the SEA 94xx module is executed. The GPS receiver is automatically configured and can be used without any further initialization.

6 Programming

6.1 Migration

The GPS driver version 3 comes with some major improvements (when compared to version 2), which allow much better timing control, but requires some adaptation to the user application. These adaptations are fairly easy and self-explaining, nevertheless an additional document has been created to cover this issue in details. This document can be free downloaded from the S.E.A. web site:

http://www.sea-gmbh.com

On the main page select *Support* from the main menu. On the support page select *Downloads* from the local menu and click on the *Download Area* button. On the download site select your module type from the *Hardware Products* category.

6.2 Examples

The driver software is delivered with a set of examples that demonstrate a specific aspect of the module. Please refer to the examples to learn how to program the module and how to retrieve position and time values.

The examples are available via the LabVIEW Example Finder.

Use the search keyword GPS, 9405, 9410 or 9414 to find related examples.

6.3 Basic Concepts

The GPS telemetry data is internally organised in messages with sets of common data. There are messages containing position-, status- or time information. This implies that all data of one message can be considered as 'received at the same time'. The messages arrive subsequently in the module. This concept has impact on the design of the FPGA nodes, which are exposed to the user as the Application Programming Interface (API) for the module.

A node represents a single GPS data unit (e.g. longitude) rather than an entire message. Thus, the data from a single message is spread over multiple nodes. For a better understanding the node names have a prefix (e.g. *TIME* or *Timepulse*) to indicate the original message. All nodes with the same prefix descend from the same message. For each message a node called *ToW (ms)* is available. That node delivers a time stamp, at which the specific message has been generated. These nodes can be used to match data between different messages. Example: in order to synchronise a system clock a time stamp and the precise time alignment (time point the new time stamp is applied) is required. All those information is available from the *Timestamp* nodes. However, if the system clock needs to by synchronised to UTC the leap second information is required as well. Unfortunately, the leap second information is provided by a *TIME* node. Now, to ensure integrity of the data the *ToW (ms)* values of the *Timestamp* and *TIME* nodes have to be compared. IF they are equal the data can be considered as 'received at the same time' and used together to calculate the UTC based time stamp.

The term *Timepulse* refers to a periodic digital signal that indicates the start of every second of the GPS clock, an absolute and very precise time source. The *Timepulse* signal enables a tight time synchronisation across systems by using only the GPS radio signal, regardless their distance or location on earth. The *Timepulse* period starts with a pulse (high-level phase) with a length of a tenth of the period, followed by a low-level phase of nine tenth of the period. Per default the *Timepulse* has a period of 1 second with a pulse duration of 100 ms.

6.4 Application Programming Interface (API)

The application programming interface provides the user with access to the module, offering functions to read module's data and control module's behaviour. The functions are accessible through the LabVIEW FPGA nodes, which are located on the functions palette of a FPGA VI inside the *FPGA I/O* sub palette like shown in the screen shot below (please note that the FPGA target has a different functions palette than e.g. 'My Computer' target).

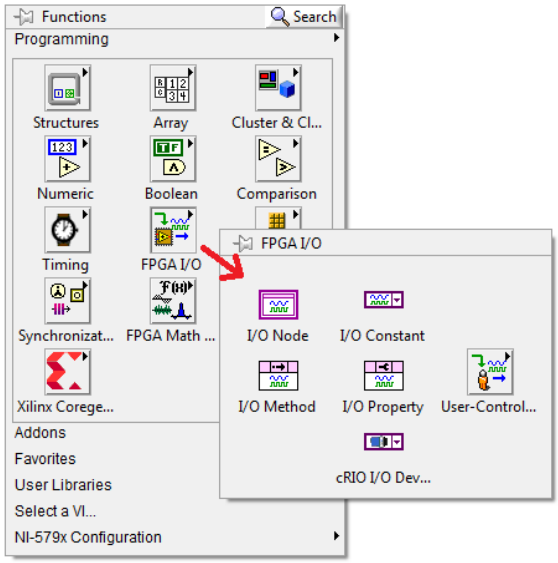

Fig. 11: FPGA Nodes on Functions Palette

The API functions are spread over three node types, depending on their purpose:

- I/O Node - contains functions that interacts with the module's hardware I/O (connectors)
- I/O Property - contains functions to retrieve the GPS telemetry data
- I/O Method - contains functions to configure the module's behaviour

The subsequent chapters describe all API functions or each supported FPGA node type.

6.4.1 I/O Nodes

I/O nodes control module's hardware connectors.

I/O Node	Data Type	Direction	Description
Sync Output	Boolean	read write	This node drives the *Sync Output* connector on the module's front, when the *Timepulse To FPGA* I/O property is set to *True*. Otherwise write this node has no effect. The current output level can also be read back.
Timepulse	Boolean	read only	This node returns the GPS *Timepulse* signal, when the *Timepulse To FPGA* I/O property is set to *True*. Otherwise this node returns *False*.

Tab. 1: I/O Nodes

6.4.2 I/O Property Nodes

I/O Property nodes retrieve GPS telemetry data.

I/O Property	Data Type	Direction	Description
Mod ID	U16	read only	The module type identification. The returned value (decimal) is 43 for SEA 9405, 44 for SEA 9410 and 46 for SEA 9414.
Mod Serial Number	U32	read only	The serial number of the module. This value (hexadecimal) is the same as what can be read on the module housing.
Mod Vendor ID	U16	read only	The vendor identification. All S.E.A. modules return 0x4711.
POS Height (mm)	I32	read only	The antenna height above ellipsoid, based on WGS84.

POS Height MSL (mm)	I32	read only	The antenna height above mean sea level.
POS Horizontal Acc (mm)	U32	read only	The estimated horizontal (2D) position accuracy[1].
POS Latitude	I32	read only	The latitude of the antenna position. The position is in degrees with a scaling factor of 10^7, see also Tab. 3.
POS Longitude	I32	read only	The longitude of the antenna position. The position is in degrees with a scaling factor of 10^7, see also Tab. 4.
POS ToW (ms)	U32	read only	The time of week for the position fix in milliseconds, see also Tab. 3 and 4.
POS Vertical Acc (mm)	U32	read only	The estimated vertical position accuracy.
STATUS FixType	Enum	read only	The current fix type. The possible fix types are described in Tab. 4.
STATUS Flags	Cluster	read only	Flags for the current state of the receiver. The flags are described in Tab. 4.
STATUS ToW (ms)	U32	read only	The time of week for the status information in milliseconds, see also Tab. 3 and 4.
TIME Flags	Cluster	read only	Flags for the information within the TIME message. They are described in Tab. 4.
TIME Leap Seconds	I8	read only	The number of leap seconds in between GPS and UTC time at the moment *TIME ToW (ms)*.
TIME ToW (ms)	U32	read only	The time of week for the TIME information in milliseconds, see also Tab. 3 and 4.
TIME Week	U16	read only	The GPS week for the TIME information, see also Tab. 3 and 4.
Timepulse Quantization Error (ps)	I32	read only	The quantization error in pico seconds of the last *Timepulse* that occurred while the *Wait for Timepulse* method node was executing.
Timepulse To FPGA	Boolean	write only	Switch for routing the *Timepulse* signal inside the module. If set to *False,* the *Timepulse* signal is routed to the *Sync Output* on the front side of the module. If set to *True* (default), the *Timepulse* signal is routed to the backplane for direct usage within a FPGA application. In that case the *Sync Output* connector on the module's front can be driven using the *Sync Output* I/O node.
Timepulse ToW (ms)	FXP ‹64,32›	read only	The time of week (see Tab. 3) of the last *Timepulse* that occurred while the *Wait for Timepulse* method node was executing. This value contains a fractional part (sub milliseconds). Please note: Although the fractional part usually equals to zero it shall always be used for calculations, as the omission may lead to incorrect results.
Timepulse Week	U16	read only	The GPS week (see Tab. 3) for the last *Timepulse* that occurred while the *Wait for Timepulse* method node was executing.

Tab. 2: I/O Property Nodes

1 The accuracy estimation highly depends on the number of satellites visible. If there are more than five satellites visible, 95% of the positional values are better than the given accuracy estimation.

The GPS telemetry data implicitly uses a few constants that need to be known for further calculations. The constants are listed in the Tab. 3 below.

The usage of constants is shown in the examples enclosed with the driver software. Please refer to the Real-Time VIs for implementation details.

Constant	Description
GPS Epoch	This constant defines an absolute point in time, at which the GPS time started to count. This value is **6. January 1980 at 00:00:00 UTC**. This constant is required to calculate an absolute time stamps based on *Week* and *ToW* data.
GPS Week Start	This constant defines an absolute point in time, at which a GPS Week starts. This value is **last Sunday at 00:00:00 UTC**[1]. This constant is required to calculate an absolute time stamp based on *ToW* data.

Tab. 3: GPS Constants

Some GPS telemetry data is delivered in a format that may be uncommon or require some explanation. The table Tab. 4 lists these nodes and explains how to interpret the particular GPS data.

The formatting of the GPS telemetry data is shown in the examples enclosed with the driver software. Please refer to the Real-Time VIs for implementation details.

I/O Property	Description
POS Longitude, POS Latitude	Values are returned in degrees multiplied with the factor 10.000.000 (10^7). To get the common used value format divide by 10.000.000 (10^7), as shown in the examples.
Week, ToW, Leap Seconds	The GPS time data is delivered as separate time components (Week, ToW, Leap Second...). To get an absolute time stamp these time components need to be composed together as shown in the examples.
STATUS Fix	Enumeration, possible values: • **no fix** - No GPS fix at all • **dead reckoning only** - not applicable (n/a) • **2D-fix** - Only position fix, no height information available • **3D-fix** - Position and height available • **GPS + dead reckoning combined** - not applicable (n/a) • **Time only fix** - Only timing information is available
STATUS Flags	Cluster, binary values (flags): • **ToW valid** - Indicates whether the *STATUS ToW (ms)* node value is valid (True). • **Week valid** - Indicates whether the *TIME Week* node value is valid (True). • **DGPS** - Indicates whether differential GPS is used (True). The GPS module is only able to use SBAS as a source for differential GPS information. The module is not able to get the DGPS accuracy information from other sources, like a fixed reference antenna for example. • **Fix OK** - Indicates whether the position and velocity[2] informations are correct (True).
TIME Flags	Cluster, binary values (flags): • **ToW valid** - Indicates whether the *TIME ToW (ms)* node value is valid (True). • **Week valid** - Indicates whether the *TIME Week* node value is valid (True).

1 The *Week* value start with zero every night form Saturday to Sunday at 00:00:00

2 Velocity is only available from NMEA messages accessible through the “Read RAW” I/O method. Refer to the examples for details.

	• **leapS valid** - Indicates whether the *TIME Leap Seconds* node value is valid (True).

Tab. 4: GPS Data Formats

6.4.3 I/O Method Nodes

I/O Method nodes configure the behaviour of the module.

6.4.3.1 Configure Message

This node enables or disables a particular GPS message. Per default only the messages returning GPS telemetry data (through I/O property nodes) are enabled. The detailed description of the interface in shown below.

Parameter	Data Type	Direction	Description
Message	Enum	write	Selects which message for the list available messages is to be en- or disabled.
Enable	Boolean	write	Controls whether the message should be enabled (True) or disabled (False). Important Note: if the messages *POS*, *STATUS*, *TIME* or *Timepulse* are disabled the respective FPGA I/O property nodes do not deliver GPS telemetry data. So, these messages should only be disabled, when performance is an issue.
OK	Boolean	read	Indicates whether the configuration change was accepted by the GPS receiver.

Tab. 5: I/O Method - Configure Message

6.4.3.2 Configure Rate

This node configures the message update rate for all active GPS messages at once. Per default the update rate is 1 second. The detailed description of the interface in shown below.

Parameter	Data Type	Direction	Description
Message Interval (ms)	U16	write	Controls the desired interval (period) for all active messages. Possible values are: • 1000 (update rate = 1Hz) • 250 (update rate = 4Hz) • 100 (update rate = 10Hz, only with SEA 9410) Values exceeding the hardware limits are coerced to the fastest possible update rates (or shortest intervals = 250 or 100).
OK	Boolean	read	Indicates whether the configuration change has been accepted.

Tab. 6: I/O Method - Configure Rate

6.4.3.3 Configure SBAS

This node configures the SBAS feature. SBAS is a Satellite-Based Augmentation System which increases the reliability, accuracy, and availability of the GPS telemetry data by using special geostationary satellites. The SBAS satellites send the correction and integrity data at the same frequency (L1) as the regular GPS satellites, so no additional receiver is required in the module. SBAS is also know as WADGPS (Wide-Area DGPS/Wide-Area Differential GPS).

Enabling SBAS causes the module to perform some more calculations, which may lead to a significant rising of the internal CPU and memory usage. As a result the data amount and/or the update rate must be decreased in order to avoid blocking of the GPS due to data overflow.

SBAS should only be used if basic accuracy is insufficient and the user is familiar with the SBAS parametrization. If unsure about the parameters use the default values as shown in Tab. 7 below.

The module is not able to get (ground-based) DGPS (Differential GPS) information from other sources like a fixed reference antenna, for example. The detailed description of the interface in shown below.

Parameter	Data Type	Direction	Description
Enable	Boolean	write	Controls whether to use SBAS (True) or not (False). When disabling SBAS also the options in *Usage* (next parameter) should be disabled (3 x False).
Usage	Cluster (booleans)	write	Controls which parts the SBAS should be used. This can be *range*, *diffCorr (differential correction)* and *integrity* information. Default: *range* = True, *diffCorr* = True, *integrity* = False
Maximum Search Channels	Enum	write	Controls the maximum number of search channels to be used. This value reaches from 0 to 3. Default: 3
Scanmode	Boolean[39]	write	This bit field controls which Pseudo Random Number (PRN) to be used. A particular PRN is assigned to a particular SBAS satellite. The lowest index refers to PRN 120 and the highest to PRN 158. Default PRNs: 120, 124, 126, 129, 133, 134, 137, 138
OK	Boolean	read	Indicates whether the configuration change has been accepted.

Tab. 7: I/O Method - Configure SBAS

6.4.3.4 Configure Timepulse

This node configures the *Timepulse* signal. The detailed description of the interface in shown below.

Parameter	Data Type	Direction	Description
Antenna Cable Delay (ns)	I16	write	Controls the signal delay due to the antenna cable. Default: 50
Pulse Length (us)	U32	write	Controls the length of the time pulse (high-level phase) in microseconds. Default: 100000 (corresponds to 100 ms)
Pulse Period (us)	U32	write	Controls the time pulse period in microseconds. For most accurate timing information a pulse period of 1000000 us (default) has to be used. Furthermore pulse periods which are non-integer dividers of 1000 ms lead to non-working timing information. Default: 1000000 (corresponds to 1 second)
OK	Boolean	read	Indicates whether the configuration change has been accepted.

Tab. 8: I/O Method - Configure Timepulse

The *Configure Timepulse* node includes an *Antenna Cable Delay* input which allows to enter an integer value corresponding to the specific signal delay in nanoseconds caused by the applied GPS antenna cable. The cable delay is calculated as the multiplication of *cable length (m) * signal velocity (ns/m)* and rounded to the next integer value.

Example: a common used S.E.A. GPS antenna (part no.: 61000002) has a 5 meter cable of type RG174 (signal velocity 5.05 ns/m). Thus, the calculated cable delay is 5 [m] * 5.05 [ns/m] = 25.25 [ns], which results in a rounded *Antenna Cable Delay* parameter value of 25.

6.4.3.5 Read RAW

This node reads the 'raw' GPS telemetry data stream. The stream contains the GPS messages as described in the section *Basic Concepts* above. The messages can be used to gain additional information. For this, however, the stream has to be parsed by the user, which is exemplary shown in the respective example enclosed with the driver. There are basically two types of messages: readable ASCII messages and binary messages. The readable ASCII messages are standard NMEA 0183 messages, which are well documented in the web. The binary messages are not for customer usage and therefore not documented, however, they may by useful for debugging purposes. The detailed description of the interface in shown below.

Parameter	Data Type	Direction	Description
Byte	U8	read	Returns the next byte from the internal RAW FIFO.
Valid	Boolean	read	Indicates whether this read operation is correct.
Overflow	Boolean	read	Indicates whether an overflow on the internal RAW FIFO had happened. This is most likely due to a too slow execution rate of this method node.

Tab. 9: I/O Method - Read RAW

6.4.3.6 Reset

This node enforces a reset of the GPS receiver unit. The reset executes a restart of the receiver according to the **Type** parameter.

Parameter	Data Type	Direction	Description
Type	Enum	write	*Hotstart*: restarts the receiver without clearing any internal data. *Warmstart*: restarts the receiver and clears the ephemeris data. *Coldstart*: restarts the receiver and clears all data (ephemeris, almanach, health, klobuchard, position...)

Tab. 10: I/O Method – Reset

The Method also resets custom settings like the selected messages or update rate. So ensure to re-run the Configure Method nodes after executing a reset.

6.4.3.7 Wait For POS

This node waits until a new navigation fix is available or the timeout occurs. If there was a new message between the last execution of the node and the recurrent execution the node also terminates. When the node terminates it also makes the position information available for the according property nodes (prefix POS). A call of this method node is mandatory to trigger an update of data in the associated property nodes. The detailed description of the interface in shown below.

Parameter	Data Type	Direction	Description
Timeout	U32	write	Controls the desired timeout in clock ticks for waiting for the new POS data.
Timed Out	Boolean	read	Indicates whether the waiting has timed out. If a timeout occurs also the position information does not get updated.

Tab. 11: I/O Method - Wait For POS

6.4.3.8 Wait For STATUS

This node waits until a new STATUS data is available or the timeout occurs. If there was a new message between the last execution of the node and the recurrent execution the node also terminates. When the node terminates it also makes the position information available for the according property nodes (prefix

STATUS). A call of this method node is mandatory to trigger an update of data in the associated property nodes. The detailed description of the interface in shown below.

Parameter	Data Type	Direction	Description
Timeout	U32	write	Controls the desired timeout in clock ticks for waiting for the new STATUS data.
Timed Out	Boolean	read	Indicates whether the waiting has timed out. If a timeout occurs also the status information does not get updated.

Tab. 12: I/O Method - Wait For STATUS

6.4.3.9 Wait For TIME

This node waits until a new TIME data is available or the timeout occurs. If there was a new message between the last execution of the node and the recurrent execution the node also terminates. When the node terminates it also makes the position information available for the according property nodes (prefix TIME). A call of this method node is mandatory to trigger an update of data in the associated property nodes. The detailed description of the interface in shown below.

Parameter	Data Type	Direction	Description
Timeout	U32	write	Controls the desired timeout in clock ticks for waiting for the new TIME data.
Timed Out	Boolean	read	Indicates whether the waiting has timed out. If a timeout occurs also the status information does not get updated.

Tab. 13: I/O Method - Wait For TIME

6.4.3.10 Wait For Timepulse

This node waits until the next *Timepulse* or the timeout occurs. After the node terminates the timing informations for the *Timepulse* are available via the regarding property nodes. The node terminates almost immediately and always within the same amount of clock cycles for every execution, after the *Timepulse* occurred, so that it can be used for time synchronization. Those values also get overwritten, if the *Timepulse* message had been deactivated using the *Configure Message* method node. A call of this method node is mandatory to trigger an update of data in the associated property nodes. The detailed description of the interface in shown below.

Parameter	Data Type	Direction	Description
Timeout	U32	write	Controls the desired timeout in clock ticks for waiting for the next *Timepulse*.
Timed Out	Boolean	read	Indicates whether the waiting has timed out.

Tab. 14: I/O Method - Wait For Timepulse

6.5 Error Codes

In case of unexpected behaviour the driver software returns an error at the *error out* terminal of the originating FPGA node. By default, the error in- and out terminals are hidden. In order to show these terminals the option S*how Error Terminals* of a node (right-clicking on the node) must be enabled.

Adding error terminals will increase FPGA usage and compilation time.

The following errors and warnings are possible:

Error Code	Description
Module Errors	
65536	Incorrect module type found. Insert the proper module type.
65537	No module found. Insert the module to the correct chassis slot.
65637	Incorrect Program Mode. Ensure that the module is configured to 'LabVIEW FPGA' in NI-MAX.

Tab. 15: Error Codes

7 Deployment

If an application containing the SEA 94xx is deployed to a run-time target system (NI CompactRIO) the following parts are necessary to be deployed along with the compiled application (i.e. `rtexe`):

- LabVIEW Run-Time Engine 2017 or higher

8 Trouble Shooting

The following hints might help you to determine if the module and/or software behave correctly and how to identify module failures.

#	Problem	How to solve
1	The module is not detected within LabVIEW.	1. Make sure you have the NI CompactRIO system powered and connected to your PC. 2. Make sure you have installed the current driver software after de-installing the previous driver. 3. Enable Error terminal in nodes and see if any errors occur.
2	GPS Data is not available.	1. Wait until LED 4 on the module blinks. The GPS receiver needs some time to collect data from at least 4 satellites until the first data is returned. 2. Check if the GPS antenna is connected to the module and placed outside buildings or next to a window. 3. Disable SBAS. 4. Enable Error terminal in nodes and see if any errors occur.
3	GPS data is not returned or some data is missing.	1. Disable SBAS. 2. Reduce messages in raw stream. 3. Reduce update rate.

Tab. 16: Trouble Shooting

A Figures

Fig. 1: Quick Start – NI-MAX configuration....8
Fig. 2: Quick Start – Add new target....9
Fig. 3: Quick Start – Project explorer after completing the configuration....10
Fig. 4: Quick Start – Add new FPGA VI....11
Fig. 5: Quick Start – Select module....12
Fig. 6: Quick Start – Create method node....13
Fig. 7: Quick Start – Wire method node....13
Fig. 8: Quick Start – Create property node....13
Fig. 9: Quick Start – Select property....14
Fig. 10: Quick Start – Final block diagram....14
Fig. 11: FPGA Nodes on Functions Palette....18

B Tables

Tab. 1: I/O Nodes....18
Tab. 2: I/O Property Nodes....19
Tab. 3: GPS Constants....20
Tab. 4: GPS Data Formats....21
Tab. 5: I/O Method - Configure Message....21
Tab. 6: I/O Method - Configure Rate....21
Tab. 7: I/O Method - Configure SBAS....22
Tab. 8: I/O Method - Configure Timepulse....22
Tab. 9: I/O Method - Read RAW....23
Tab. 10: I/O Method – Reset....23
Tab. 11: I/O Method - Wait For POS....23
Tab. 12: I/O Method - Wait For STATUS....24
Tab. 13: I/O Method - Wait For TIME....24
Tab. 14: I/O Method - Wait For Timepulse....24
Tab. 15: Error Codes....25
Tab. 16: Trouble Shooting....27