Software Manual – Programming and Integration

SEA 9406

Multi GNSS High Precision Modules for NI CompactRIO™

s·e·a Science & Engineering Applications Datentechnik GmbH

Doc. No.: HB/SEA 9406 SoftwareManual/1.0.a/Apr-2022

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

#	Description	Changes
1	1.0.a	Initial release for LabVIEW 2017 (or newer) supporting all current NI CompactRIO systems. Based on Module Development Kit 2.1 (MDK2.1).

Tab. 1: Change notes

2 Getting Started

2.1 General

Before starting to work with the SEA 9406 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 `https://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 9406 module.

Refer to the appropriate NI™ documentation for details on NI™ 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 9406 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™ `Tools` menu after installation.

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
ℹ	The blue mark highlights tips
📘	Reference to other documents

Tab. 2: Structuring elements/symbols

Development device is a PC with LabVIEW™ Development Environment to create and compile (FPGA) the user application.

Run-time device is the computer that runs the compiled (FPGA) user application. It is typically an NI CompactRIO™ system.

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

Name	Meaning
API	**A**pplication **P**rogramming **I**nterface
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
IMU	**I**nertial **M**easurement **U**nit
MDK2.1	**M**odule **D**evelopment **K**it, version **2.1**. MDK2 is a framework by NI for developing 3rd party cRIO modules.
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.
NTRIP	**N**etworked **T**ransport of **R**TCM via **I**nternet **P**rotocol
RTCM	**R**adio **T**echnical **C**ommission for **M**aritime Services
RTK	**R**eal-**T**ime **K**inematics
SBAS	**S**atellite **B**ased **A**ugmentation **S**ystem
WT	**W**heel **T**ick

Tab. 3: Glossary

3 Installation

SEA 9406 modules require a special driver software (also called module support) for operation. 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 the 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:

https://www.sea-gmbh.com

Navigation: 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 device only (refer to the hardware requirements 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 9406 module requires at least a PC to program and compile the user application. In order to run the application an NI CompactRIO Real-Time controller with the SEA module inserted in an arbitrary chassis slot is required.

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

The cRIO modules do not auto-discover in NI cRIO-904x/905x chassis. This is a known behaviour. The workaround is to manually add the modules to the LabVIEW project. For more details refer to: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000YSbmCAG&l=en-GB

3.2 Software Requirements

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

Software requirements for the development device:

- 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
- NI CompactRIO drivers

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

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

The 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 9406 module can be integrated into an existing NI CompactRIO system.

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

The cRIO modules do not auto-discover in NI cRIO-904x/905x chassis. This is a known behaviour. The workaround is to manually add the modules to the LabVIEW project. For more details refer to: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000YSbmCAG&l=en-GB

Follow the steps below to create a simple application using the SEA 9406 module and run it on a NI CompactRIO system:

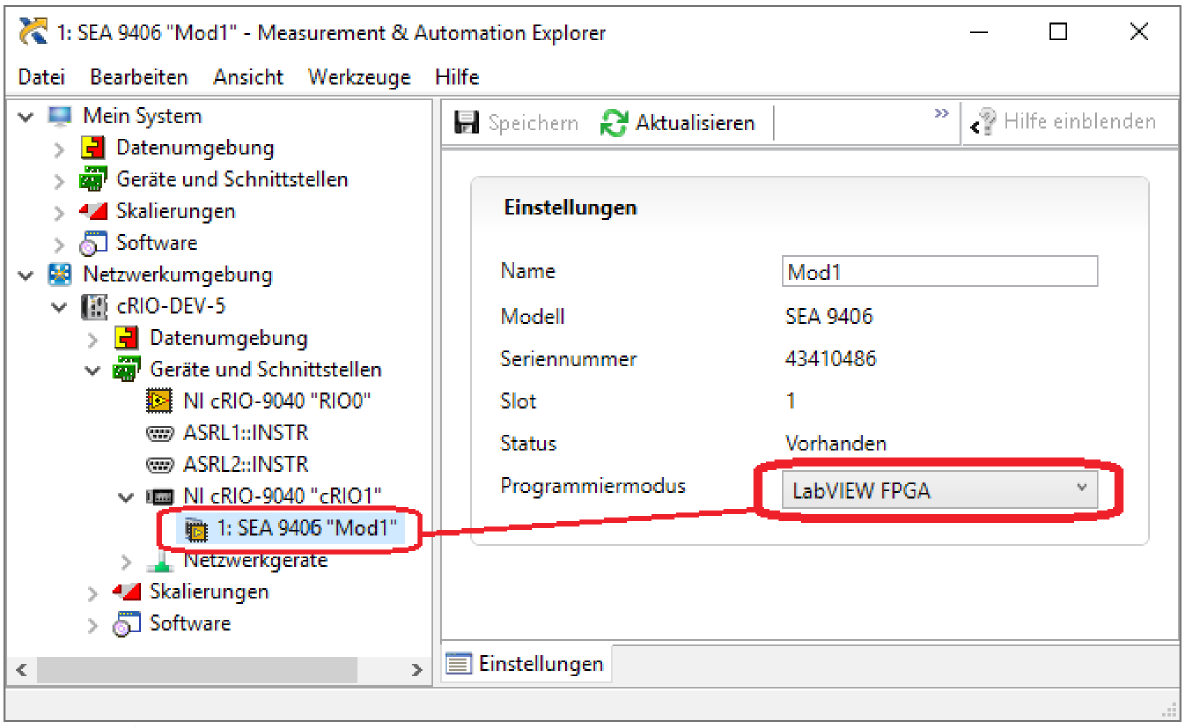
1. Ensure that you meet the hardware and software requirements listed in the previous chapter and the required software is installed on your PC and on the NI CompactRIO system.

2. Insert the SEA 9406 module into your NI CompactRIO system in a slot of your choice (for this tutorial SEA 9406 in slot 1 has been used). Connect the GNSS antenna to the module and place the antenna outside buildings or next to a window for a good sky view.

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

4. Install the module's driver software. If you have the setup file locally available just double-click at it and start the installation through VIPM. If you don't have the setup file locally available search for *9406* in VIPM and install the driver software directly from here.

5. Optionally: detect/add the NI CompactRIO system in NI-MAX Network environment (German: *Netzwerkumgebung*). Ensure that all utilized modules are configured to *LabVIEW FPGA*, refer to the screen shot below. Close NI-MAX afterwards.


1: SEA 9406 "Mod1" - Measurement & Automation Explorer

Einstellungen

Name	Mod1
Modell	SEA 9406
Seriennummer	43410486
Slot	1
Status	Vorhanden
Programmiermodus	LabVIEW FPGA

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:

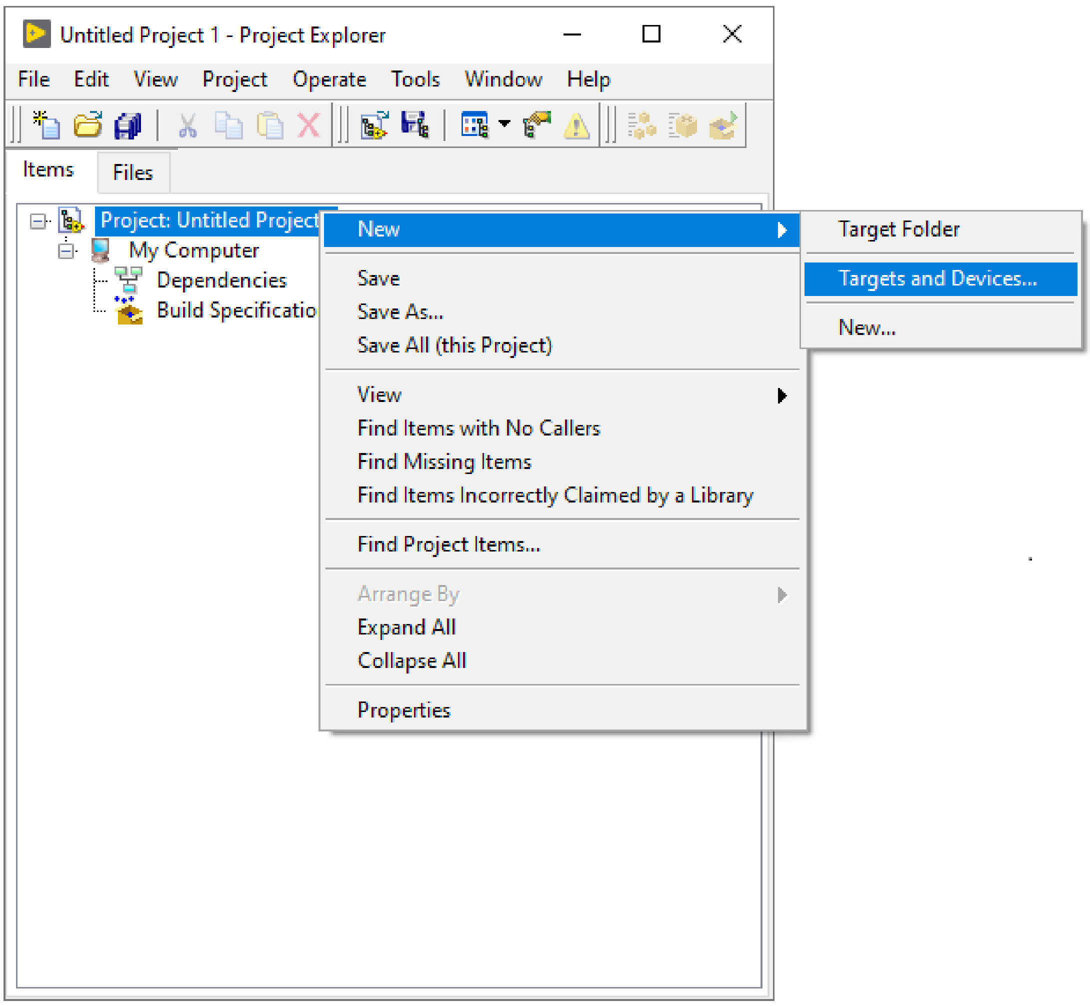
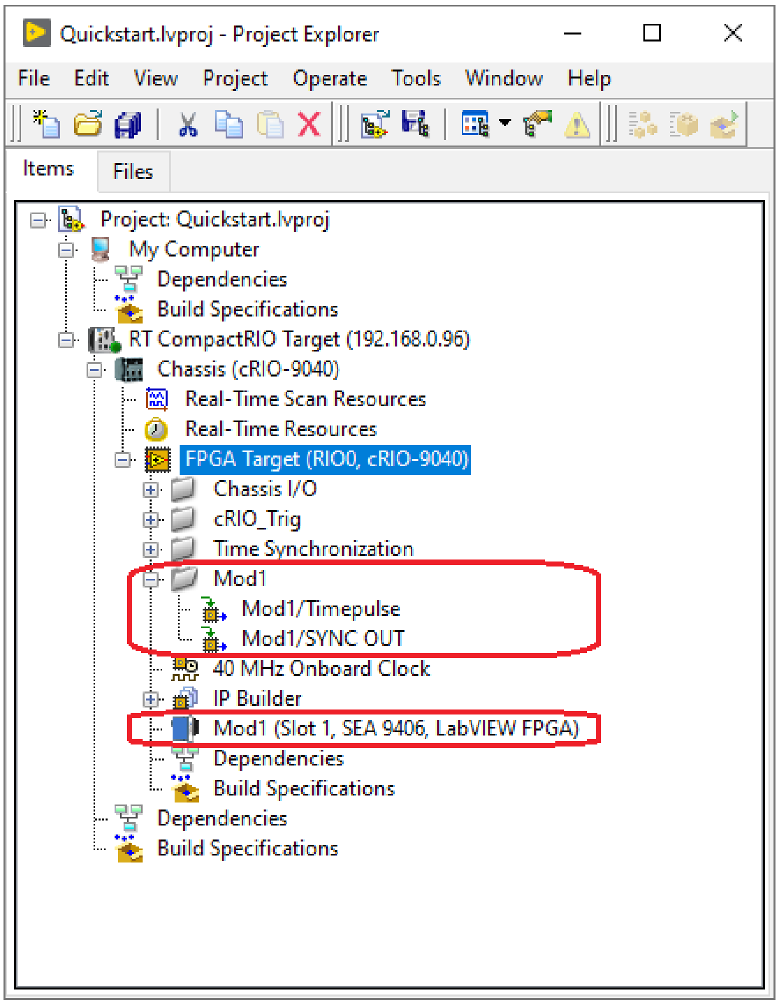


Fig. 2: Quick Start – Add new target

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.


Quickstart.lvproj - Project Explorer

```
Project: Quickstart.lvproj
  My Computer
    Dependencies
    Build Specifications
  RT CompactRIO Target (192.168.0.96)
    Chassis (cRIO-9040)
      Real-Time Scan Resources
      Real-Time Resources
      FPGA Target (RIO0, cRIO-9040)
        Chassis I/O
        cRIO_Trig
        Time Synchronization
        Mod1
          Mod1/Timepulse
          Mod1/SYNC OUT
        40 MHz Onboard Clock
        IP Builder
        Mod1 (Slot 1, SEA 9406, LabVIEW FPGA)
        Dependencies
        Build Specifications
    Dependencies
    Build Specifications
```

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

The project is now fully configured and the SEA 9406 resources can be used to create a user FPGA application.

You can continue from here to learn how to implement a simple FPGA application using the SEA 9406 module.

11. Create a new FPGA VI in the project explorer window. For this right-click on the *FPGA Target (RIO0...)* and select *New* –> *VI*. Refer to figure 4 below:

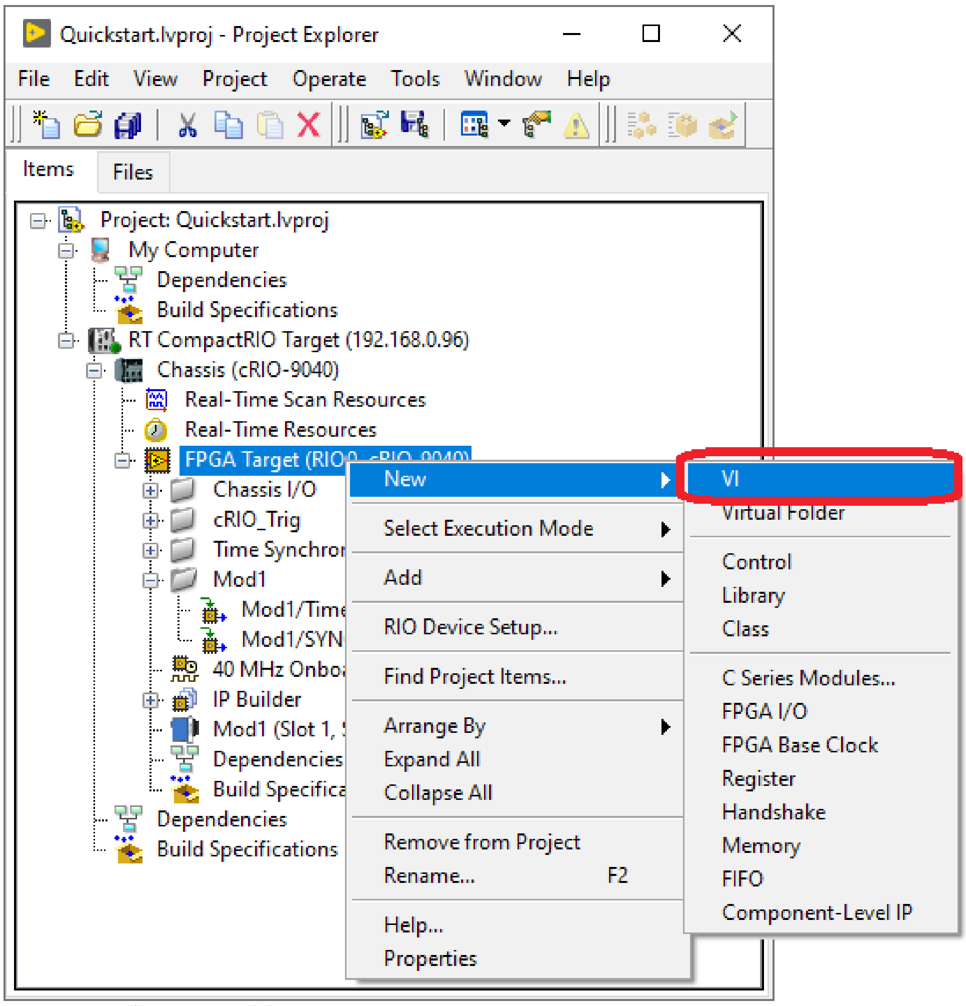


Fig. 4: Quick Start – Add new FPGA VI

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.

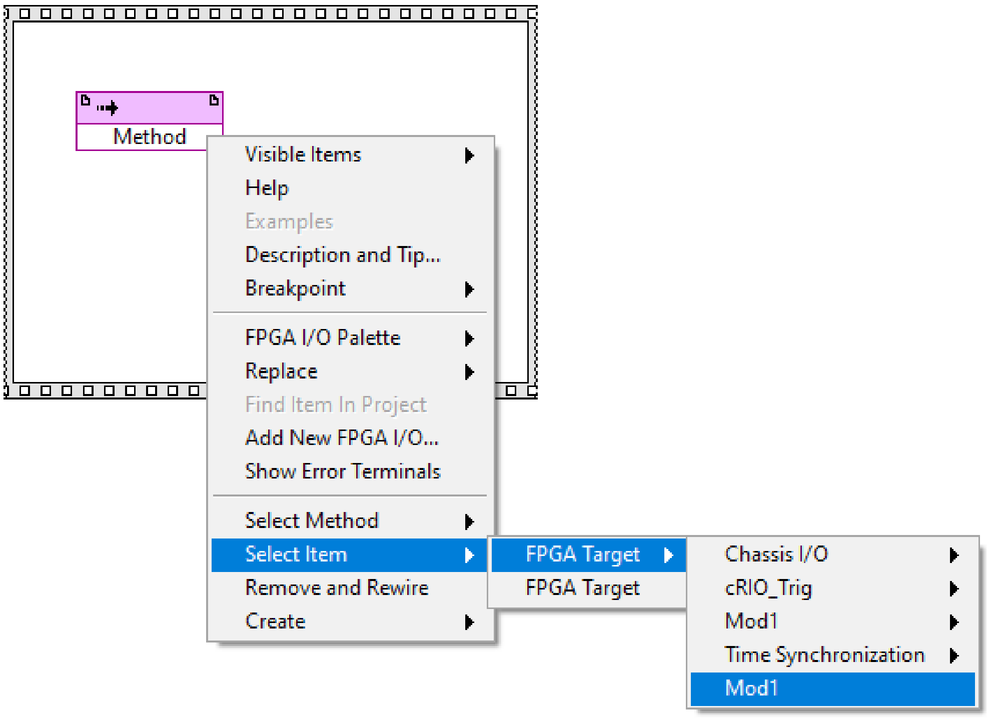


Fig. 5: Quick Start – Select module

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

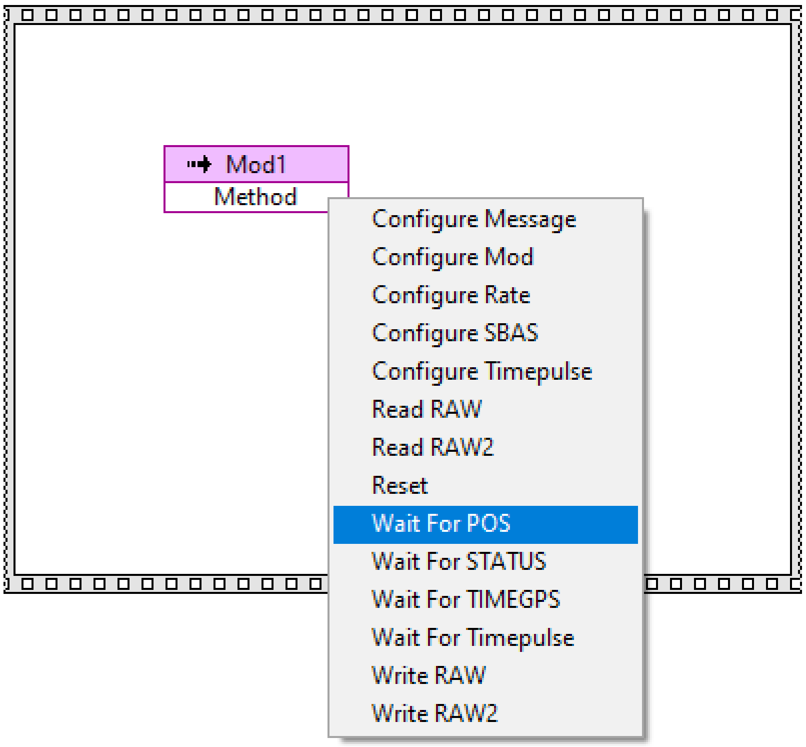


Fig. 6: Quick Start – Create method node

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

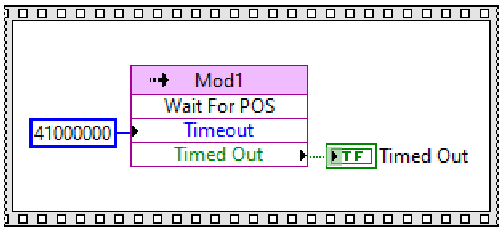


Fig. 7: Quick Start – Wire method node

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.

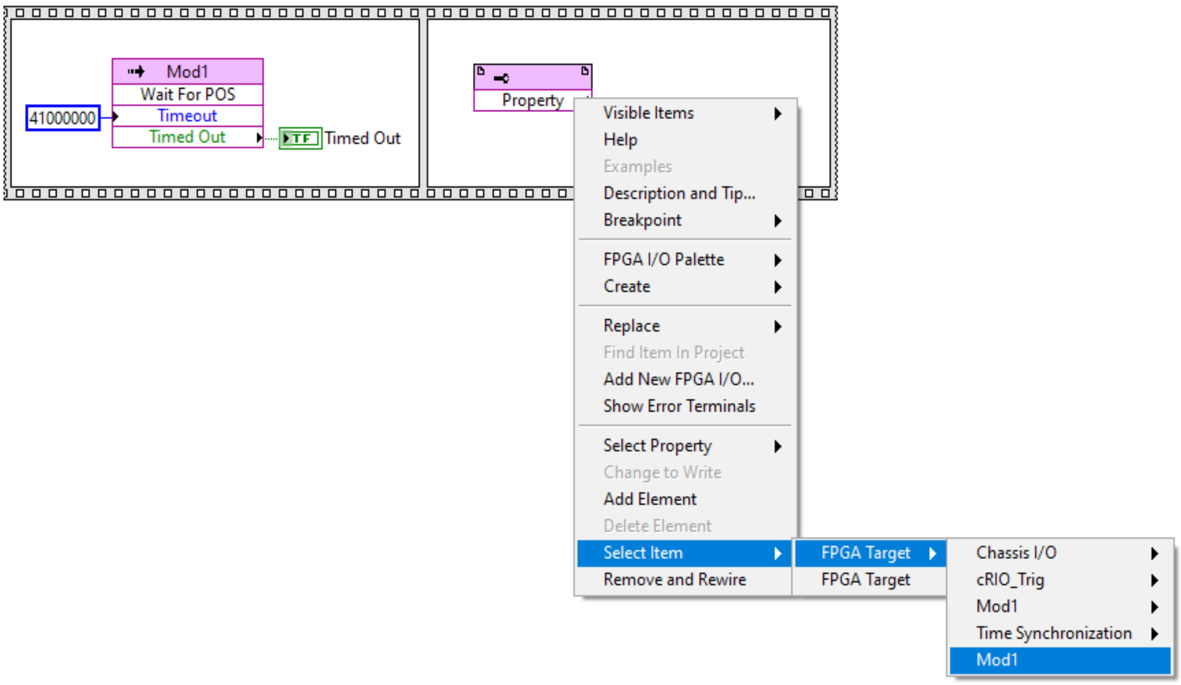


Fig. 8: Quick Start – Create property node

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

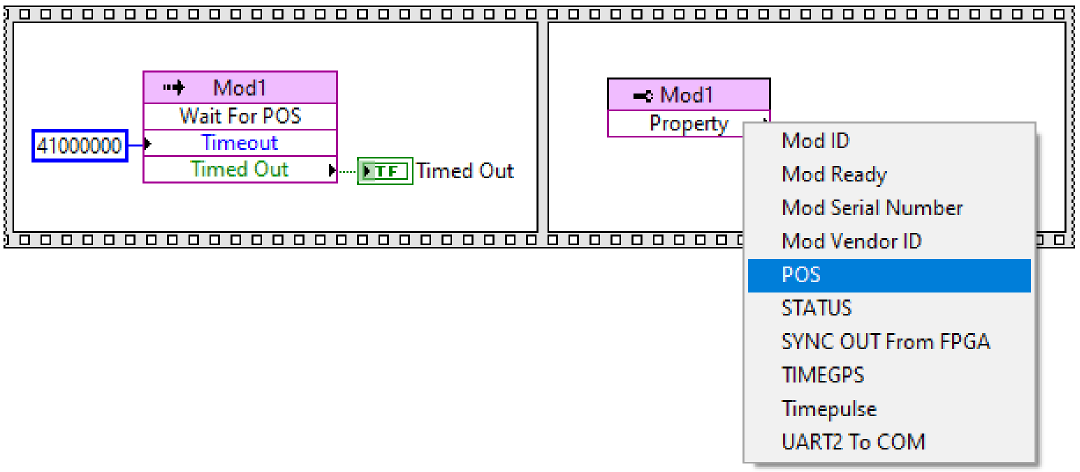


Fig. 9: Quick Start – Select property

18. Connect the *POS* property with an *Unbundle By Name* function (*Cluster & Class Palette*) and select element *Latitude* as shown in figure 14.

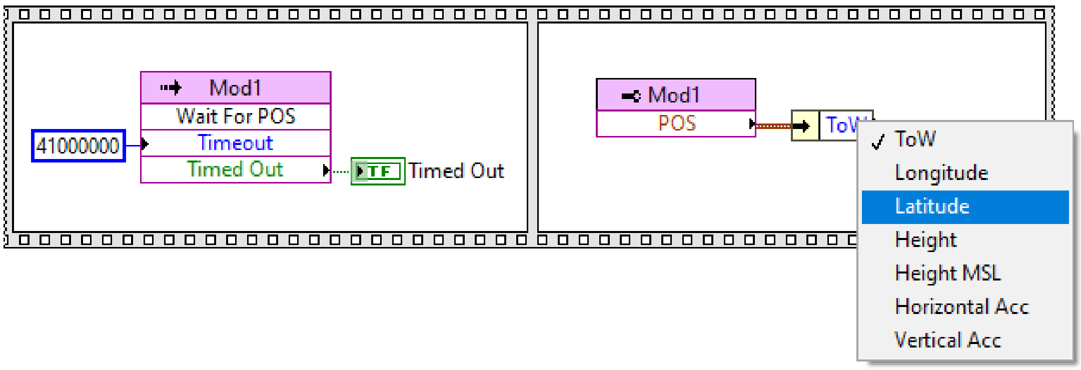


Fig. 10: Quick Start – Add "Unbundle By Name" function and select "Latitude"

19. Optionally repeat step 18 to add further property Cluster elements. Complete the example by placing indicators for the Cluster elements as well as a While Loop to ensure that the node is read continuously. The final code may look like follows:

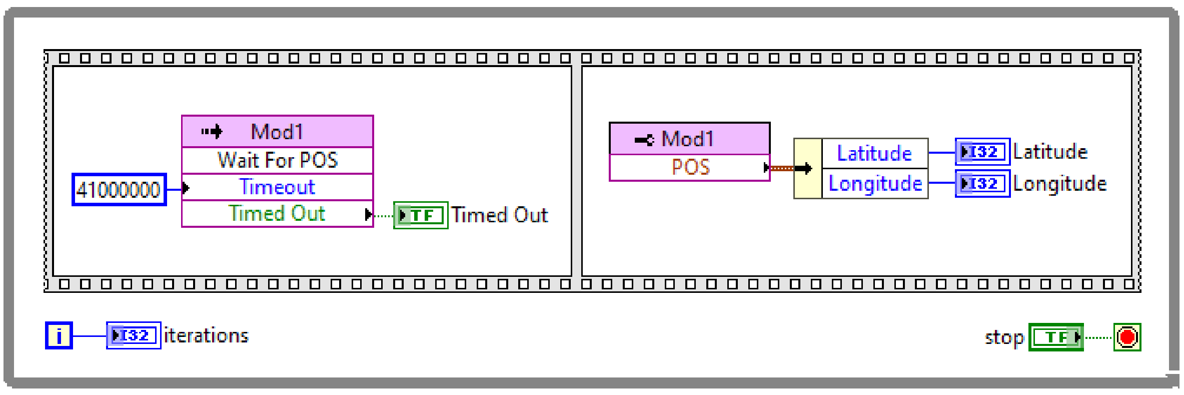


Fig. 11: Quick Start – Final block diagram

<u>Note:</u> The implementation of GNSS 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) GNSS data is read.

20. Save the created FPGA VI, create a build specification and compile it.

21. The indicators will display the current latitude and longitude in degrees (scaled by a factor `1e7`) of the GNSS receiver. Note: it may take some time (up to some minutes) until the correct position values are displayed, because the GNSS 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 9406 modules are cRIO compatible modules that can be used in a wide range of carriers from *NI*. All NI CompactRIO systems with a programmable (FPGA) backplane are currently supported. SEA 9406 modules are not supported in systems without an FPGA programmable backplane like NI CompactDAQ™.

5.2 SEA 9406 Functionality

The SEA 9406 cRIO modules feature the reception of GNSS telemetry data (basically global position and precise time information) and provide these data to be used directly within the FPGA or optionally as a data stream within the RT target.

The SEA 9406 cRIO module is suitable for a wide variety of applications, including position monitoring absolute timing and synchronization.

The SEA 9406 cRIO modules offer:

- Concurrent usage of all available GNSS (GPS, BeiDou, GLONASS, Galileo)
- RTK/RTCM 3.x (via NTRIP) – requires additional/external service

  RTCM is a binary data protocol used to supply GNSS receivers with real-time differential correction information. NTRIP is an open standard protocol for streaming differential information over the internet in accordance with specifications published by RTCM.
- Optional sensor (IMU, wheel tick) support
- Direct access to geo-positional and timing data in FPGA
- Full access to GNSS 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 20 Hz message update rate (excl. raw data)
- GNSS 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 9406 module is executed. The GNSS receiver is automatically configured and can be used without any further initialization.

6 Programming

6.1 Examples

The driver software is delivered with a set of examples that demonstrate a specific aspect of the module functionality. Please refer to the examples to learn how to use/program the module.

The examples are available via the *NI Example Finder* (LabVIEW menu › *Help* › *Find Examples*...).

Use the search keyword *9406* to find related examples.

6.2 Basic Concepts

The GNSS telemetry data is organised in sets of common data called messages. There are messages containing position-, status- or time information. The messages arrive periodically in the module. This concept has impact on the Application Programming Interface (API) and hence on the design of the FPGA nodes, which are exposed to the user.

An FPGA node with telemetry data typically represents an entire message rather than a single GNSS data unit (e.g. longitude) and is provided as a LabVIEW data cluster. The particular data unit can be easily extracted using the respective LabVIEW function (i.e. *Unbundle By Name*).

Every message is tagged with a time stamp called ToW (Time of Week), which is the time since the GNSS epoch (see Tab. 7). This implies that all data of one message can be considered as 'received at the same time'. Furthermore, messages containing the same time ToW can be considered as originated from the same navigation solution. 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 *Timepulse* node. However, if the system clock needs to be synchronised to UTC the leap second information is required as well. Unfortunately, the leap second information is provided by a *TIMEGPS* node. Now, to ensure integrity of the data the *ToW* values of the *Timestamp* and *TIMEGPS* 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* (or *TP* or *PPS*) refers to a periodic digital signal that indicates the start of every second of the GNSS clock and provides an absolute and very precise time source. The *Timepulse* signal enables a tight time synchronisation across systems by using only the GNSS radio signal, regardless of 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.

6.3 Application Programming Interface (API)

The Application Programming Interface (API) provides the user with access to the module's capabilities, offering functions to read the module's data and control the module's behaviour. The functions are accessible through the LabVIEW FPGA nodes, which are located on the functions palette of an 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).

The API functions are spread over three different 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 GNSS 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.

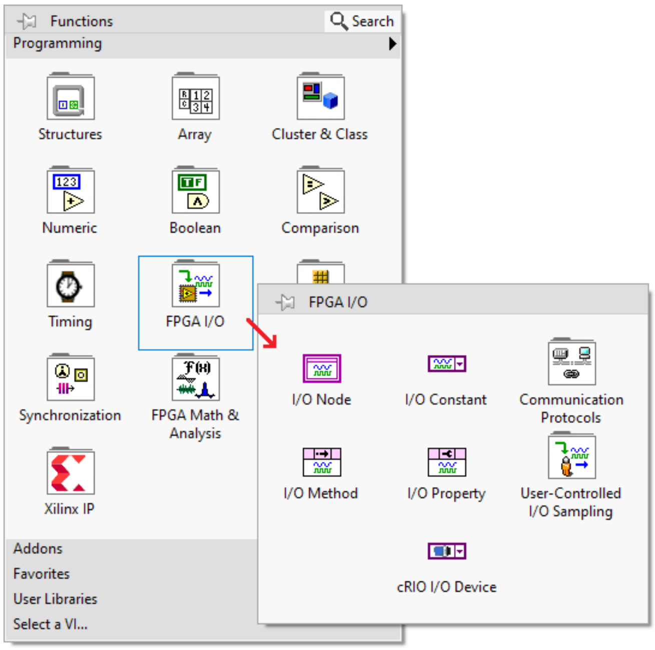


Fig. 12: FPGA Nodes on Functions Palette

6.3.1 I/O Nodes

I/O nodes control module's hardware connectors.

I/O Node	Data Type	Direction	Description
SYNC OUT	Boolean	read/ write	This node drives the *SYNC OUT* connector on the module's front, when the *SYNC OUT From 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 *Timepulse* hardware signal from the GNSS receiver.

Tab. 4: I/O Nodes

6.3.2 I/O Property Nodes

I/O Property nodes retrieve GNSS telemetry data and controls some module switches.

Property	Data Type	Direction	Description
Mod ID	U16	read only	The module type identification. The returned value (decimal) is 18 for all SEA 9406 variants.
Mod Ready	Boolean	read only	Returns *False* if the module is (re)starting; returns *True* if the module is ready for operation. The node can thus be used to indicate the completion of a device initialization.
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	Cluster	read only	LabVIEW data cluster with data of the last navigation fix. Each data unit of the cluster is described in the software.

Property	Data Type	Direction	Description
STATUS	Cluster	read only	LabVIEW data cluster with status information. Each data unit of the cluster is described in the software.
SYNC OUT From FPGA	Boolean	write only	Switches the signal source of the *SYNC OUT* connector on the module. If set to *False,* the hardware *Timepulse* signal from the GNSS receiver is directly routed to the *SYNC OUT* connector. If set to *True* (default), the *SYNC OUT* connector is driven by the *SYNC OUT* I/O FPGA node.
TIMEGPS	Cluster	read only	LabVIEW data cluster with timing information. This data shall be used for timing applications. Each data unit of the cluster is described in the software.
Timepulse	Cluster	read only	LabVIEW data cluster with information related to the hardware *Timepulse* signal. Each data unit of the cluster is described in the software.
UART2 To COM	Boolean	write only	Switches the second serial port of the GNSS receiver between module's backplane and the *COM* connector on the module's front. If set to *False,* the UART2 is routed to the module's backplane. If set to *True* (default), the UART2 is routed to the *COM* connector. The UART2 is used for RTK.

Tab. 5: I/O Property Nodes

Some GNSS telemetry data is delivered in a format that may be uncommon or require some explanation. Table 6 lists these nodes and explains how to interpret the particular GNSS data.

The units of measurement are documented in the LabVIEW context help. Just open the context help (function key *F1*) and hover with the mouse over a particular data unit inside the cluster.

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

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 GNSS 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.GNSS Fix	Enumeration, possible values: • **no fix** – No GNSS 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 • **GNSS + dead reckoning combined** – not applicable (n/a) • **Time only fix** – Only timing information is available

Tab. 6: GNSS Data Formats

The GNSS telemetry data implicitly uses a few constants that need to be known for further calculations. The constants are listed in Tab. 7 below.

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

Constant	Description
GNSS Epoch	This constant defines an absolute point in time, at which the GNSS 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.
GNSS Week Start	This constant defines an absolute point in time, at which a GNSS 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. 7: GPS Constants

6.3.3 I/O Method Nodes

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

6.3.3.1 Configure Mod

This node changes the initial configuration (baud rate) of the communication interfaces. It only needs to be executed if other than the default baud rate is required on any of the ports. During the reconfiguration the property node *Mod Ready* goes off.

Parameter	Data Type	Direction	Description
Baudrate	Enum	write	Required baud rate for the selected port (below). A list of available baud rates are provided by the input (enum). Default value is 38400.
Port	Enum	write	Port for which the baud rate should be changed. A list of available ports is provided by the input (enum).

Tab. 8: I/O Method – Configure Mod

6.3.3.2 Configure Message

This node enables or disables a particular GNSS message. Per default only the messages returning GNSS 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 enabled or disabled.
Enable	Boolean	write	Controls whether the message should be enabled (*True*) or disabled (*False*). ⚠ Important Note: If the messages *POS*, *STATUS*, *TIMEGPS* or *Timepulse* are disabled the respective FPGA I/O property nodes do not deliver GNSS 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 GNSS receiver.

Tab. 9: I/O Method – Configure Message

6.3.3.3 Configure Rate

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

1 The *Week* value starts with zero every night from Saturday to Sunday at 00:00:00.

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 = 1 Hz) • 250 (update rate = 4 Hz) • 100 (update rate = 10 Hz) • 50 (update rate = 20 Hz) Values exceeding the hardware limits are coerced to the fastest possible update rates (or shortest interval = 50 ms).
OK	Boolean	read	Indicates whether the configuration change has been accepted.

Tab. 10: I/O Method – Configure Rate

6.3.3.4 Configure SBAS

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

Enabling SBAS causes the module to perform some more calculations, which may lead to a significant rise 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 GNSS 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. 11 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 **P**seudo **R**andom **N**umber (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. 11: I/O Method – Configure SBAS

6.3.3.5 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 Period (µs)	U32	write	Controls the time pulse period in microseconds. For most accurate timing information a pulse period of 1000000 µs (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)
Pulse Length µs)	U32	write	Controls the length of the time pulse (high-level phase) in microseconds. Default: 100000 (corresponds to 100 ms)
OK	Boolean	read	Indicates whether the configuration change has been accepted.

Tab. 12: 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 GNSS 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 commonly used S.E.A. GNSS antenna 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.

6.3.3.6 Read RAW

This node reads the 'raw' GNSS telemetry data stream. The stream contains the GNSS messages as described in the section 6.2 *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 data FIFO had happened. This is most likely due to a too slow execution rate of this method node, or too much enabled messages
Backlog	U32	read	Indicates how many bytes are still in the data FIFO (unread data).

Tab. 13: I/O Method – Read RAW

6.3.3.7 Reset

This node enforces a reset of the GNSS 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, almanac, health, klobuchar, position...)

Tab. 14: I/O Method – Reset

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

6.3.3.8 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 POS property 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 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. 15: I/O Method – Wait For POS

6.3.3.9 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 STATUS property 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 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. 16: I/O Method – Wait For STATUS

6.3.3.10 Wait For TIMEGPS

This node waits until a new TIMEGPS 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 TIMEGPS property 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 new TIMEGPS 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. 17: I/O Method – Wait For TIMEGPS

6.3.3.11 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 function was successful or has timed out.

Tab. 18: I/O Method – Wait For Timepulse

6.3.3.12 Write RAW

This node writes data to the GNSS receiver. This is an advanced function for special purposes and not intended to be used directly by the user. Improper usage can lead malfunction or hardware damage.

Parameter	Data Type	Direction	Description
Byte	U8	write	Data byte to write
Timed Out	Boolean	read	Indicates whether the function was successful or has timed out.

Tab. 19: I/O Method – Write RAW

6.3.3.13 Write RAW2

This node writes data to the GNSS receiver. This is an advanced function for special purposes and not intended to be used directly by the user. Improper usage can lead malfunction or hardware damage.

Parameter	Data Type	Direction	Description
Byte	U8	write	Data byte to write
Timed Out	Boolean	read	Indicates whether the function was successful or has timed out.

Tab. 20: I/O Method – Write RAW2

6.4 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 *Show 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. 21: Error codes

6.5 Conditional Disable Symbols

The driver software utilizes the LabVIEW Conditional Disable Symbols to enable/disable advanced features of the module. In regular operation these switches are not required and should not be used. For SEA 9406 modules two symbols are available:

Symbol	Description
TxUART1RAW	Enables the *Write RAW* node if value set to *EN*. This is only required, if sensor features are used.
TxUART2RAW	Enables the *Write RAW2* node if value set to *EN*. This is only required, if RTK (NTRIP) data should be provided through the UART2 on the backplane. Currently this feature is only supported through the *COM* connector.

Tab. 22: Conditional Disable Symbols

7 Deployment

If a user application containing the SEA 9406 is deployed to a run-time device (i.e. 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 automatically detected within a LabVIEW project	1. Automatic cRIO module detection is not supported on some NI CompactRIO targets (i.e. NI cRIO-904x/905x. Add modules manually instead. For details see: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000YSbmCAG&l=en-GB
2	The module is not found within LabVIEW project	1. Make sure you have installed the LabVIEW development environment like described in section 3.2 *Software Requirements*. 2. Make sure you have installed the driver software (module support) for the used SEA cRIO module.
3	GNSS Data is not provided	1. Wait until LED 2 on the module blinks. The GNSS receiver needs some time to collect data from at least 4 satellites until the first data is returned. 2. Check if the GNSS 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.
4	GNSS data is not returned or some data is missing	1. Disable SBAS. 2. Deactivate some messages. 3. Reduce update rate (default is 1 sec).

Tab. 23: Trouble shooting

A Figures

B Tables