Software Manual – Programming and Integration

SEA 9510

EnDat Interface Module

Doc. no.: HB/SEA 9510 SoftwareManual/2.3.b/Feb-2023

s·e·a Science & Engineering Applications Datentechnik GmbH

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	2.0.0	Update release for new LabVIEW driver architecture: Module Development Kit 2.0, MDK2 (NI cRIO 9951).
2	2.1.x	Major bugfixes: • SEA-9521_GetPosition (RT) example didn't work with all encoders • GetPosition I/O node time out increased as it was too low at 100 kHz bus speed and also for certain encoders and could lead to malfunction. Main new features: • R-Series Expansion Chassis support • MXI-Express RIO Chassis support • EtherCAT Chassis Support • Extended “EnDat 2.2 Additional Info” support
3	2.2.0	Minor changes: • A4 page size • Getting Started chapter • Installation chapter • Deployment chapter
4	2.2.1	Minor update: • Acquisition rates are slightly higher with driver versions 2.2+
5	2.3.a	Changes: • Support for NI CompactRIO 904x targets added. Oldest supported LabVIEW version is now 2017.
6	2.3.b	Minor changes: • Document formatting • Images replaced

2 Getting Started

2.1 General

Before starting to work with the SEA 9510 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 9510 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 9510 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™ `Help` 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. 1: Structuring elements/symbols

2.4 Additional Sources

Technical Information: EnDat 2.2 – Bidirectional Interface for Position Encoders, *DR. JOHANNES HEIDENHAIN GmbH*

3 Installation

SEA 9510 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 9510 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 describes how easy the SEA 9510 module can be integrated into an existing CompactRIO system.

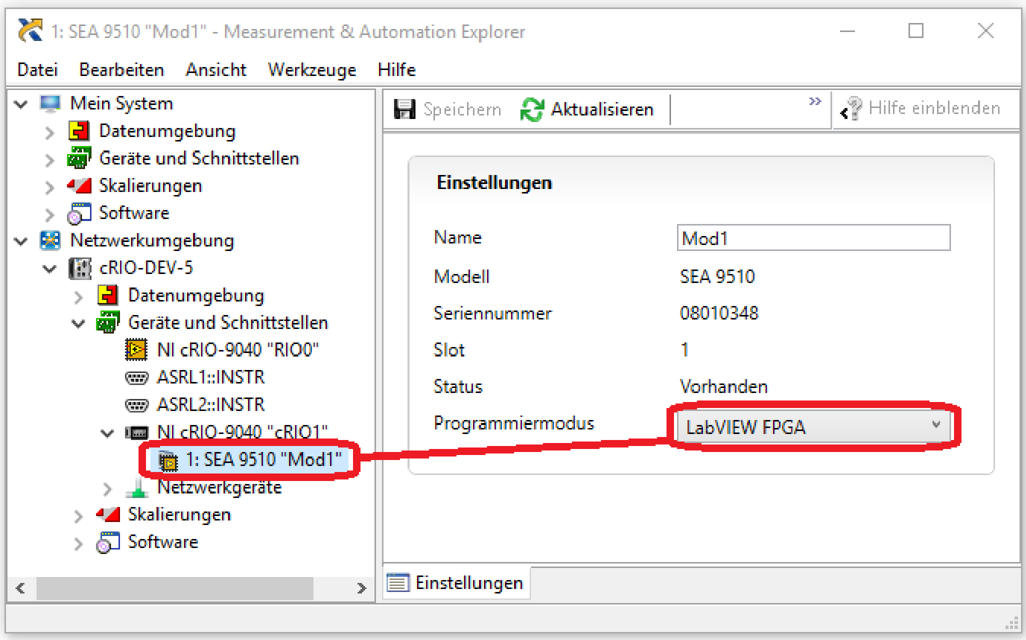
Refer to the hardware manual for proper installation of the hardware.

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 9510* module and run it on a NI CompactRIO system (for this tutorial the NI cRIO-9040 has been used exemplary):

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 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 9510 driver software. For this, the VI Packet Manager (VIPM) is required. If VIPM is not yet installed on your system please start LabVIEW and select *Tools → Find LabVIEW Add-ons...*. Complete the VIPM installation following the instructions.
5. Detect/add the NI CompactRIO system in the NI-MAX (*Measurement & Automation Explorer*) *Remote Systems* (German: *Netzwerkumgebung*). In the *Devices and Interfaces* settings, ensure that the SEA 9510 is configured to the Program Mode *LabVIEW FPGA*, refer to Fig. 1. Close NI-MAX afterwards.



Einstellungen

Name	Mod1
Modell	SEA 9510
Seriennummer	08010348
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:

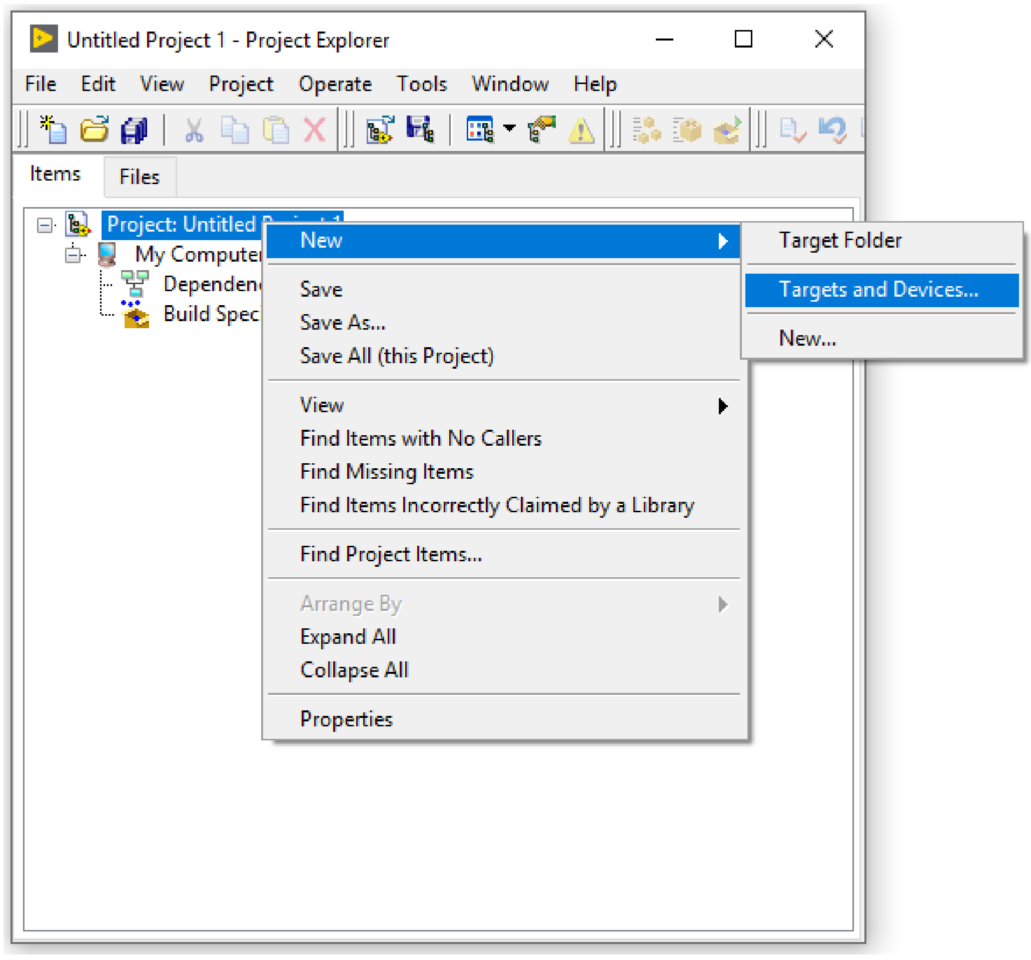
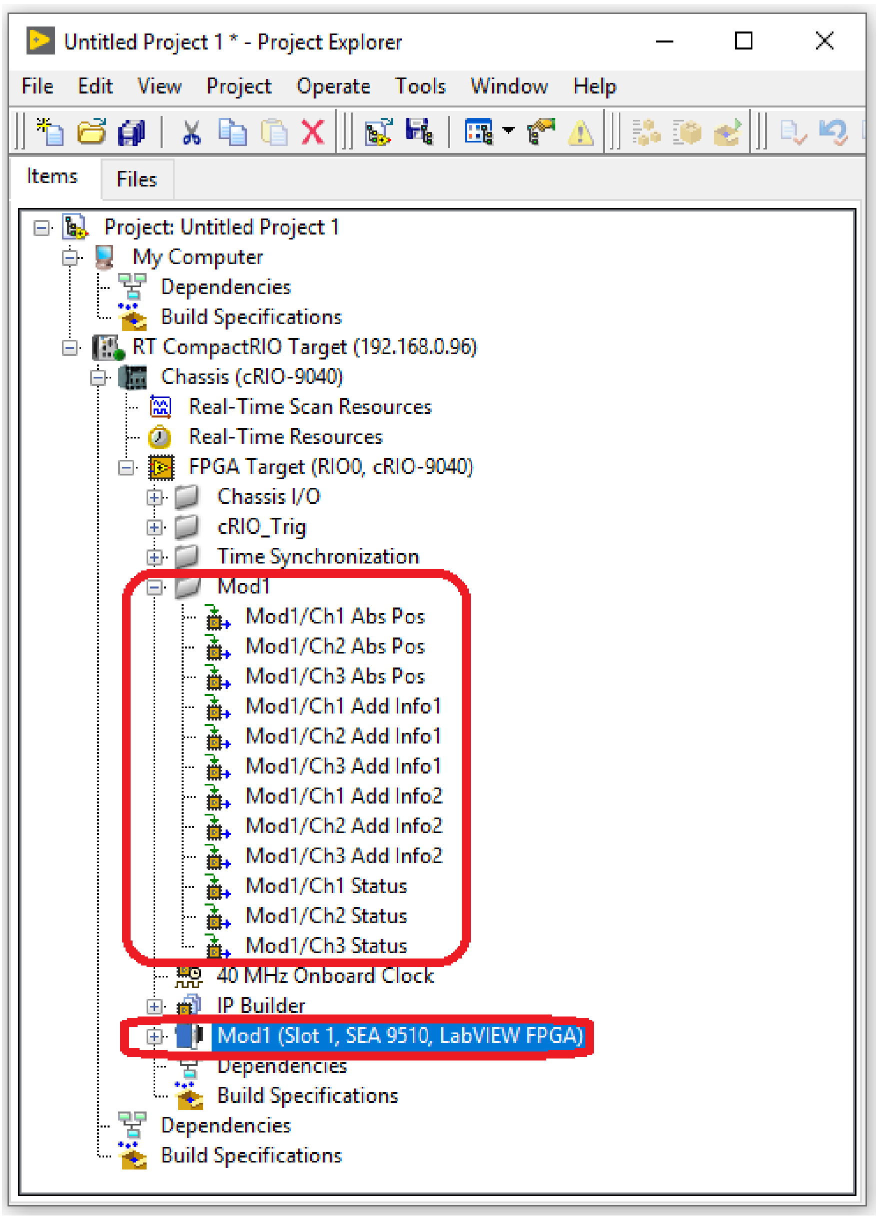


Fig. 2: Quick Start – Add new target

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.


Untitled Project 1 * - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files

```
Project: Untitled Project 1
  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/Ch1 Abs Pos
          Mod1/Ch2 Abs Pos
          Mod1/Ch3 Abs Pos
          Mod1/Ch1 Add Info1
          Mod1/Ch2 Add Info1
          Mod1/Ch3 Add Info1
          Mod1/Ch1 Add Info2
          Mod1/Ch2 Add Info2
          Mod1/Ch3 Add Info2
          Mod1/Ch1 Status
          Mod1/Ch2 Status
          Mod1/Ch3 Status
        40 MHz Onboard Clock
        IP Builder
        Mod1 (Slot 1, SEA 9510, 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 9510 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 9510 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:

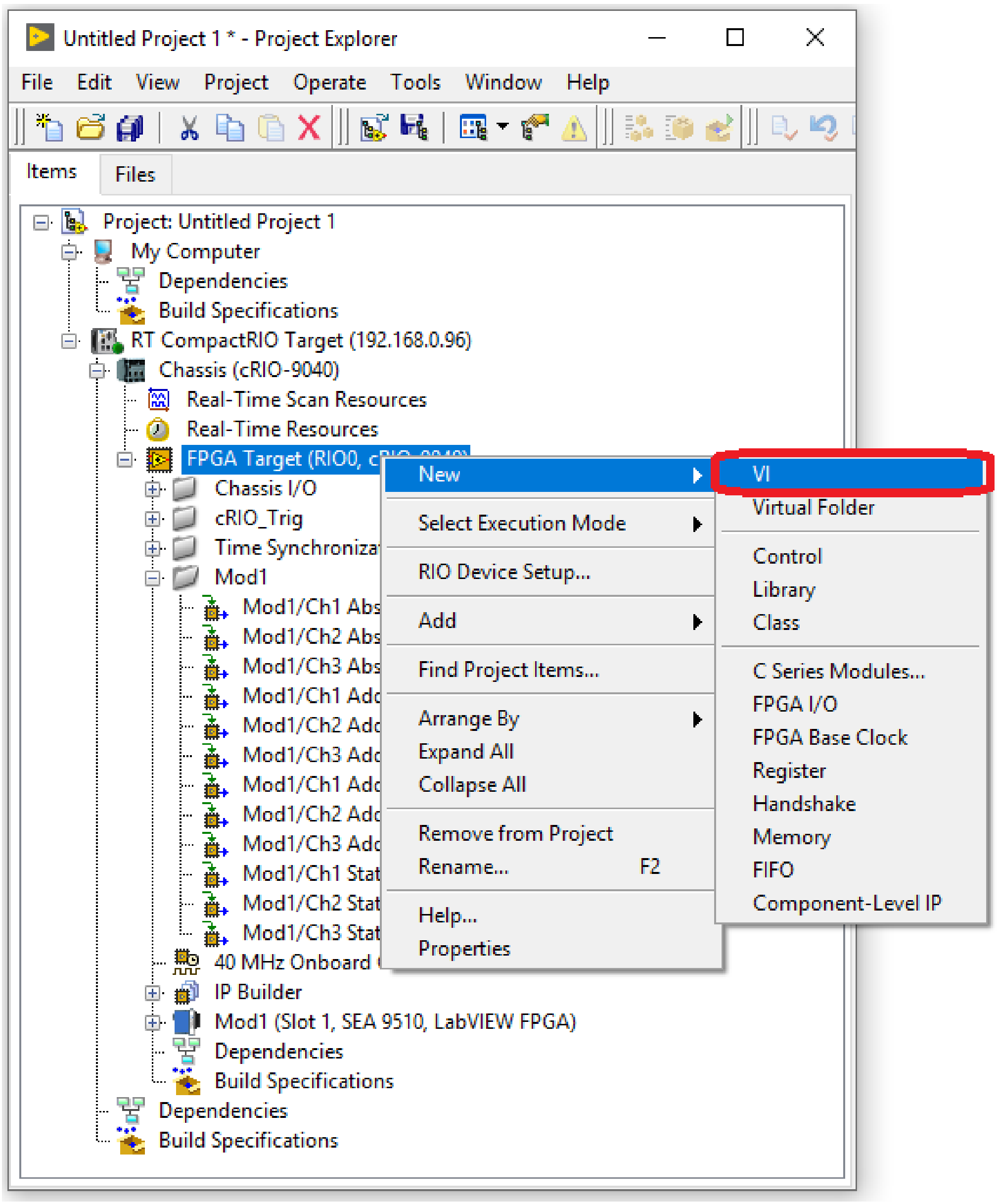


Fig. 4: Quick Start – Add new FPGA VI

12. Open the block diagram of the VI just created and place a method node from the functions palette (*FPGA I/O* –> *I/O Method*) on the block diagram. Afterwards, select e.g. the item *Mod1* (depending on the cRIO slot used).
13. Finally select the method to be executed from the list of available methods for the selected item. For this tutorial, select the *Init* method as shown in Fig. 5.

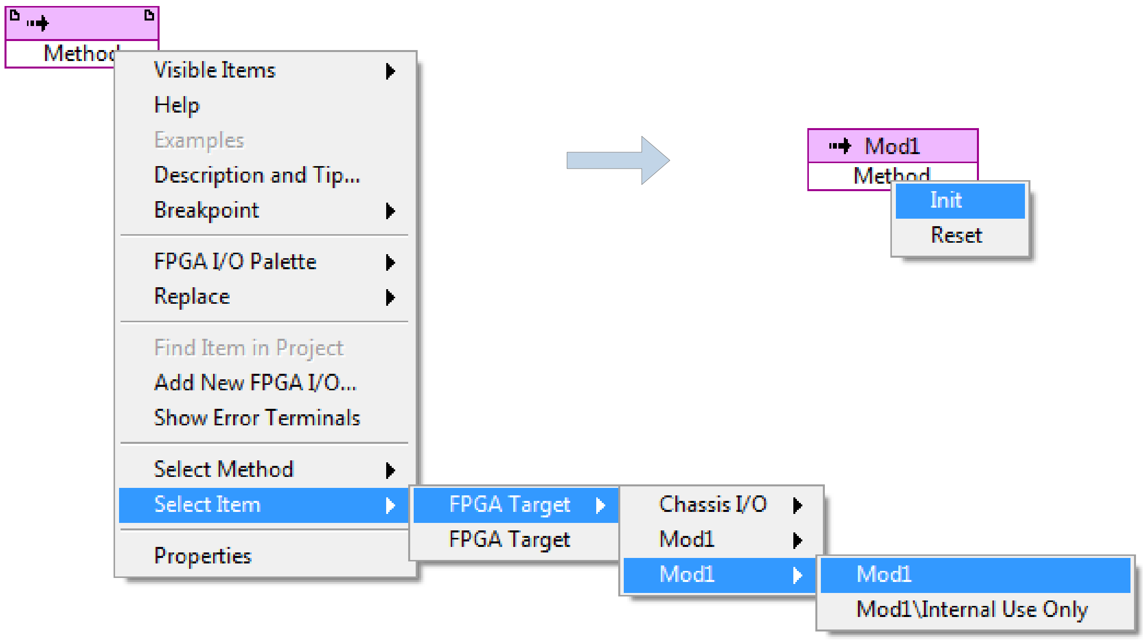


Fig. 5: Quick Start – Create initialization method

14. Repeat steps 11 and 12 with an IO node and select the absolute position input of channels 1 to 3. Complete the example placing inputs/outputs for the nodes as well as a sequence to ensure that the initialization method node gets executed before the read operation. For the inputs, create constants as shown in the figure below (an encoder attached to at least one channel is expected):

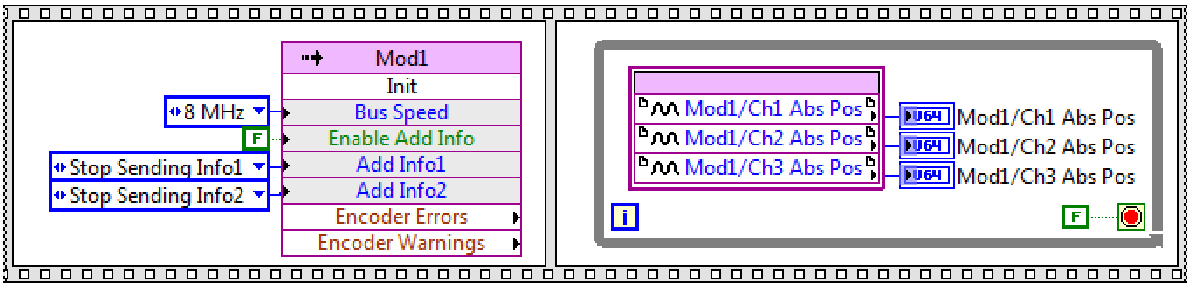


Fig. 6: Quick Start – Final block diagram

15. Save the created FPGA VI, create a build specification and compile it.
16. The *Mod1/Ch1 Abs Pos* indicator will display the absolute position which was read from the encoder attached to channel 1.

All functions are accessible via either IO, property or method nodes. For a complete list of function nodes, please refer to section 6.2 *Application Programming Interface (API)*.

5 Functional Overview

5.1 cRIO Platform

The SEA 9510 is a cRIO compatible module that can be used in a wide range of carriers from NI. All CompactRIO systems with a programmable (FPGA) backplane are currently supported. SEA 9510 is not supported in systems without an FPGA programmable backplane like CompactDAQ.

5.2 EnDat Protocol Standard

The EnDat interface from *HEIDENHAIN* is a digital, bidirectional interface for encoders. It is capable of both, transmitting position values from incremental and absolute encoders as well as transmitting or updating information stored in the encoder, or saving new information. The type of transmission (position values, parameters, diagnostics, etc.) is selected through mode commands that the subsequent electronics sends to the encoder.

5.2.1 EnDat 2.1 Mode

The SEA 9510 module supports the EnDat 2.1 protocol standard. This mode allows only one type of data to be transferred from the encoder, either position information or additional parameters. This mode is also required during initialization of the module to preserve downward compatibility.

5.2.2 EnDat 2.2 Mode

The EnDat 2.2 standard supports the transportation of the position information bundled with either one or two additional pieces of information. This mode is intended for reading and writing parameters during time critical loops.

All SEA 9510 drivers starting from version V2.1.0 fully support the concurrent retrieval of position and additional information. Switching between different types of information is possible, but cannot be done without interrupting the data acquisition for a short period of time.

5.3 SEA 9510 Functionality

The SEA 9510 cRIO module allows communication between the CompactRIO backplane and *HEIDENHAIN* EnDat encoders.

The SEA 9510 cRIO module is suitable for a wide variety of applications, including position monitoring and position control loops. The EnDat technology offers a serial differential bus protocol with added safety checks for secure position values. The module is capable of capturing three independent channels simultaneously. Cable delays are considered and measured during configuration to guarantee equal signal transport delays. Position values are read with FPGA reliability and speed at a maximum data rate of about 24 kHz per channel for absolute position values *without* additional information (the exact value is also dependent on the length of the longest cable and on the *Bus Speed*. It can be easily determined placing the IO node (position value) in a while loop and measuring the loop time.

When reading *EnDat 2.2 Additional Information* the maximal data rate reduces to about 16 kHz per channel.

The SEA 9510 cRIO modules offer:

- three independent channels.
- support for EnDat 2.2.
- full backward compatibility with EnDat 2.1.
- configurable encoder bus speed between 100 kHz and 8 MHz as defined by the EnDat standard

- integrated cable delay compensation.
- Full access to encoder memory in the host target.

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

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

Encoder bus speed is **not** the positional data acquisition rate. It describes the bit rate of transmissions between EnDat encoder and SEA 9510 module. Naturally, it will influence the actual data acquisition rate which is much lower.

5.4 Power Up Behavior

At power up, a hardware reset of the SEA 9510 module is executed. However, before reading positions from the attached encoders, an initialization has to be performed using the *Init* method node as described in section 6.2 *Application Programming Interface (API)*.

6 Programming

6.1 Examples

The SEA 9510 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 values.

The examples are available via the example finder. Use search keywords like *EnDat* or *9510* to find the related examples.

6.2 Application Programming Interface (API)

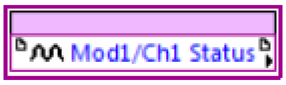
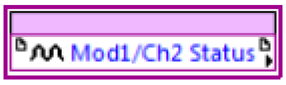
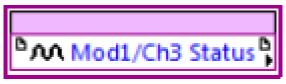
The Application Programming Interface (API) provides the user with access to the underlying EnDat protocol functionality. This interface exposes nodes that can be used within a LabVIEW FPGA application. The following different node types are available:

- IO nodes
- property nodes
- method nodes

6.2.1 IO Nodes

For each channel the IO nodes deliver the absolute position (Abs Pos), two additional information from the encoder hardware (Add Info1 + Add Info2) and the status register (Status).

Node	Direction	Description
Mod1/Ch1 Abs Pos	read only	Retrieves the absolute position of the encoder attached to channel 1.
Mod1/Ch2 Abs Pos	read only	Retrieves the absolute position of the encoder attached to channel 2.
Mod1/Ch3 Abs Pos	read only	Retrieves the absolute position of the encoder attached to channel 3.
Mod1/Ch1 Add Info1	read only	Additional information 1 of channel 1 according to EnDat 2.2 standard.
Mod1/Ch2 Add Info1	read only	Additional information 1 of channel 2 according to EnDat 2.2 standard.
Mod1/Ch3 Add Info1	read only	Additional information 1 of channel 3 according to EnDat 2.2 standard.
Mod1/Ch1 Add Info2	read only	Additional information 2 of channel 1 according to EnDat 2.2 standard.
Mod1/Ch2 Add Info2	read only	Additional information 2 of channel 2 according to EnDat 2.2 standard.
Mod1/Ch3 Add Info2	read only	Additional information 2 of channel 3 according to EnDat 2.2 standard.

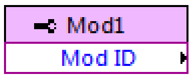
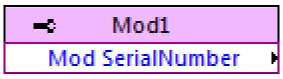
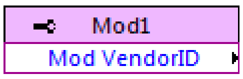
Node	Direction	Description
 Mod1/Ch1 Status	read only	Status information about the encoder attached to channel 1. For details please refer to chapter 6.3.4 *Status*.
 Mod1/Ch2 Status	read only	Status information about the encoder attached to channel 2. For details please refer to chapter 6.3.4 *Status*.
 Mod1/Ch3 Status	read only	Status information about the encoder attached to channel 3. For details please refer to chapter 6.3.4 *Status*.

Tab. 2: SEA 9510 IO Nodes

It is recommended to read the position and/or additional data of all channels using a single IO node in order to ensure proper synchronization between all channels (or between current position and additional information of a single channel). Any cable propagation delay is taken into account by propagation delay compensation featured by the SEA 9510 module.

6.2.2 Property Nodes

The property nodes deliver information about the module type and identity.

Node	Direction	Description
 Mod1 Mod ID	read only	Retrieves the module ID from the module. For SEA 9510 module it retrieves 0006 (0x0006).
 Mod1 Mod SerialNumber	read only	Retrieves the serial number from the module. The serial number is an 8 digit number.
 Mod1 Mod VendorID	read only	Retrieves the Vendor ID. This value identifies the manufacturer of the module. S.E.A. modules always return 0x4711.

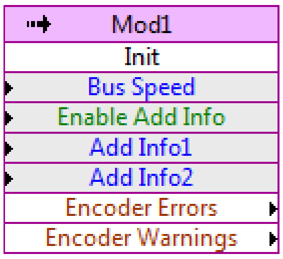
Tab. 3: SEA 9510 Property Nodes

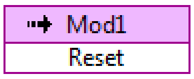
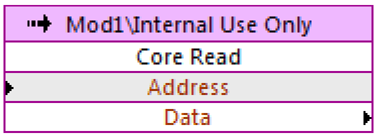
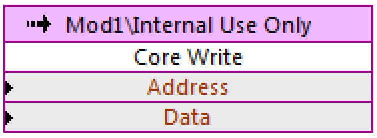
To reduce FPGA slice usage and compile time, the property node functionality can be disabled like shown in the examples, refer also to the section 6.5 *FPGA Usage and Optimization*.

Reading property nodes may not be used in parallel with IO Nodes or while executing method nodes. This will vastly reduce sampling rate or even lead to malfunction. Please use all nodes sequentially anyhow.

6.2.3 Method Nodes

Method nodes provide mandatory initialization, as well as auxiliary functionality.

Name	Direction	Description
 Mod1 Init Bus Speed Enable Add Info Add Info1 Add Info2 Encoder Errors Encoder Warnings	read and write	The *Init* method node is **mandatory** and needs to be called before reading any data from the attached encoders. This node requires some input parameters. They are described in greater detail below the table. Finally two status registers are returned: *Encoder Errors* and *Encoder Warnings*. Please refer to the respective chapters 6.3.5 and 6.3.6 for details. The execution of this node may take several seconds.

Name	Direction	Description
 Mod1 Reset	write only	The *Reset* method node resets the driver (e.g. if an irretrievable error occurs). After reset, the *Init* method node has to be called again in order to re-establish proper data transfer.
 Mod1\Internal Use Only Core Read Address Data	read only	This node provides direct **read** access to the encoder hardware. This node is not designated to be used directly by the user, instead it acts as a data transportation pipe from the encoder hardware to the host application. To host application typically concatenates multiple Core Read/Write node calls to access a single memory register, refer to the example *ReadEncoderMemory* to learn how this node is used.
 Mod1\Internal Use Only Core Write Address Data	write only	This node provides direct **write** access to the encoder hardware. This node is not designated to be used directly by the user, instead it acts as a data transportation pipe from the host application to the encoder hardware. To host application typically concatenates multiple Core Read/Write node calls to access a single memory register, refer to the example *ReadEncoderMemory* to learn how this node is used.

Tab. 4: SEA 9510 Method Nodes

The *Init* method node is essential for receiving positional data from the encoders. Without proper initialization, receiving positional data is impossible. Calling this method will:

1. detect all EnDat encoders attached to the SEA 9510 module,
2. return all module errors and warning occurred since lats initialization,
3. clear all module errors and warnings which previously occurred,
4. set the desired EnDat bus speed (default 8 MHz),
5. optionally initiate the transmission of additional information,
6. perform a propagation delay measurement,
7. initiate a cable delay compensation, and
8. start the desired EnDat operation mode (default: Send Position).

Bus Speed, *Encoder Mode*, *Add Info1* and *Add Info2* are enums, which should be created using the *Create Constant* functionality from the input context menu (right-clicking on the input). The desired value can be then selected from the list of available values.

6.3 Data Formats

The information exchanged between the SEA 9510 module hardware and the application software is organized in words with different length and meanings. The subsequent sections describe their data formats.

6.3.1 Absolute Position

Each channel of the SEA 9510 module provides an IO node with the absolute position (Abs Pos). The data width of this node is 64 bit, however, depending on the encoder only up to 48 bits are used (the remaining upper bits are zero).

The *Abs Pos* comprises single- and multiturn data bits which are concatenated and have to be separated by the user. Please refer to the examples to learn how to accomplish this.

Refer to the *ReadEncoderMemory* example, or your encoder's data sheet or to get information about the single- and multiturn values for the particular encoder you use.

Make use of the *GetPosition (FPGA)* or *GetPosition (RT)* examples to learn how to separate the absolute position into single- and multiturn position values.

6.3.2 Additional Information 1

Using the *Create Constant* context menu when right-clicking on the *Add Info 1* input of the *Init* node will create an enum giving access to various encoder memory areas being read out every cycle. Which kind of information is supported by the respective encoder can be found out reading the encoder's data sheet or by reading out the electronic data sheet (EDS) and interpreting the AddInfo1 Status of the manufacturer parameters. More information about available information can be found in *HEIDENHAIN*'s technical information on EnDat 2.2 (search for *EnDat 2.2 – Bidirectional Interface for Position Encoders* on the internet).

Temperature 2 is the internal encoder temperature whereas temperature 1 reads out an external temperature sensor.

6.3.3 Additional Information 2

By right-clicking on the *Add Info 2* input of the *Init* node you can choose what kind of additional information 2 is transmitted alongside the positional data.

Additional information will only be transmitted if the *Enable Add Info* input is set to TRUE.

6.3.4 Status

The Status is a 16 bit word, that provides operational information about the encoder like position validity, warnings or errors. The default Value is (binary): 0b0000010000000001 (=0x0401).

Bit	Function	Description
0 (LSB)	Rcv Reg 1	0 = new position data is **not** yet available (since the last reset of the flag). 1 = New position data is available.
1	Error 1	0 = Error 1 **has not** occurred. 1 = Error 1 **has** occurred. Read encoder memory for details.
2	CRC	0 = Position value is correct. 1 = Position value is **not** correct.
3	E-Type I	0 = Error Type I[1] **has not** occurred. 1 = Error Type I **has** occurred. Reinitialize encoder.
4	E-Type II	0 = Error Type II[2] **has not** occurred. 1 = Error Type II **has** occurred. Memory red/write error, check technical information about memory access.
5	MRS/Adr	0 = Acknowledge/Addressing Error **has not** occurred. 1 = Acknowledge/Addressing Error **has** occurred. Reinitialize encoder.
6	/IR6	Not used.
7	/IR7	Not used.
8	Rcv Reg 2	0 = Additional Information **2** is **not** yet available (since the last reset of the flag). 1 = Additional Information **2** is available.
9	Rcv Reg 3	0 = Additional Information **1** is **not** yet available (since the last reset of the flag). 1 = Additional Information **1** is available.
10	/Error 2	0 = EnDat 2.2 related error **has** occurred. Read encoder memory for details. 1 = EnDat 2.2 related error **has not** occurred.
11	CRC AI 1	0 = Additional Information 1 value is correct. 1 = Additional Information 1 value is **not** correct.
12	CRC AI 2	0 = Additional Information 2 value is correct 1 = Additional Information 2 value is **not** correct.
13	Busy	Not used.
14	RM	The RM bit indicates whether the reference run has been completed. In incremental systems, this is required in order to establish the absolute reference to the machine reference system. The absolute position value can then be read from the additional information 1. On absolute encoders the RM bit is always HIGH.
15	WRN	0 = Warning **has not** occurred. 1 = Warning **has** occurred. Read encoder memory for details.

Tab. 5: Status Register

1 Error Type 1 according to EnDat 2.2 specification.
2 Error Type 2 according to EnDat 2.2 specification.

6.3.5 Encoder Warnings

Encoder warnings denote the exceeding of tolerance limits for particular internal encoder values. In the long therm errors can arise from warnings. Therefore warnings have a preventive intention.

The *Encoder warnings* word has 16 bit width. Not all encoders support all following warnings:

Bit	Function	Description
0 (LSB)	Frequency Collision	0 = Frequency collision warning has not occurred. 1 = Frequency collision warning has occurred.
1	Temperature Overrun	0 = Temperature overrun warning has not occurred. 1 = Temperature overrun warning has occurred.
2	Control Reserve Lighting	0 = not reached. 1 = reached.
3	Battery Load	0 = O.K. 1 = too low.
4	Reference Position	0 = reached. 1 = not reached.
5	—	Not used.
6	—	Not used.
7	—	Not used.
8	—	Not used.
9	—	Not used.
10	—	Not used.
11	—	Not used.
12	—	Not used.
13	—	Not used.
14	—	Not used.
15	—	Not used.

Tab. 6: Encoder Warnings

6.3.6 Encoder Errors

Encoder errors occur when a malfunction of the encoder may lead to an erroneous position value.

The *Encoder Errors* word has 16 bit width. Not all encoders support all following errors (read the (electronic) data sheet for details about supported errors):

Bit	Function	Description
0 (LSB)	Lighting	0 = O.K. 1 = Failure.
1	Signal Amplitude	0 = O.K. 1 = Faulty.
2	Position Value	0 = O.K. 1 = Faulty.
3	Over Voltage	0 = No (O.K) 1 = Yes.
4	Under Voltage	0 = No (O.K.) 1 = Yes.
5	Over Current	0 = No (O.K.) 1 = Yes.
6	Battery	0 = O.K. 1 = Change necessary.
7	—	Not used.
8	—	Not used.
9	—	Not used.
10	—	Not used.
11	—	Not used.
12	—	Not used.
13	—	Not used.
14	—	Not used.
15	—	Not used.

Tab. 7: Encoder Errors

6.4 Error Codes

In case of unexpected behavior the driver software returns an error within the error cluster. The option S*how Error Terminals* of a node (right-clicking on the node) must be enabled to obtain error information.

Adding error terminals will increase FPGA usage and compilation time.

The following errors and warnings are possible:

Error Code	Description
Module Errors	
65536	No module or invalid module type found. Please insert SEA 9510 module into the correct slot. If a SEA 9510 module is recognized as invalid, please contact S.E.A. As a firmware (eeprom) update may be required.
65537	Incorrect module type found. Only SEA 9510 modules will work with this driver version. Please check if all modules are inserted into the right slot.
Encoder Errors	
358600	SEA 9510 module not (properly) initialized. Please execute the *Init* method node before reading any data from IO nodes. Check encoder errors and warnings and init status for details.
358601	No encoder connected/detected during initialization phase. Check connection, encoder type (only *HEIDENHAIN* EnDat encoders are allowed) and the additional power supply to the SEA 9510 module.
Encoder Warnings	
358615	At least one channel of SEA 9510 module reports timeout during position reading. Check connections and/or the encoder. Reset the SEA 9510 module and re-initialize the data transfer.

Tab. 8: Error Codes

6.5 FPGA Usage and Optimization

The SEA 9510 driver software consumes approx. 25–28% of available FPGA space, when using the NI 9104 backplane. Note that the FPGA space consumption for the driver code is not linear in any way. It varies with the chassis types and number of total modules installed.

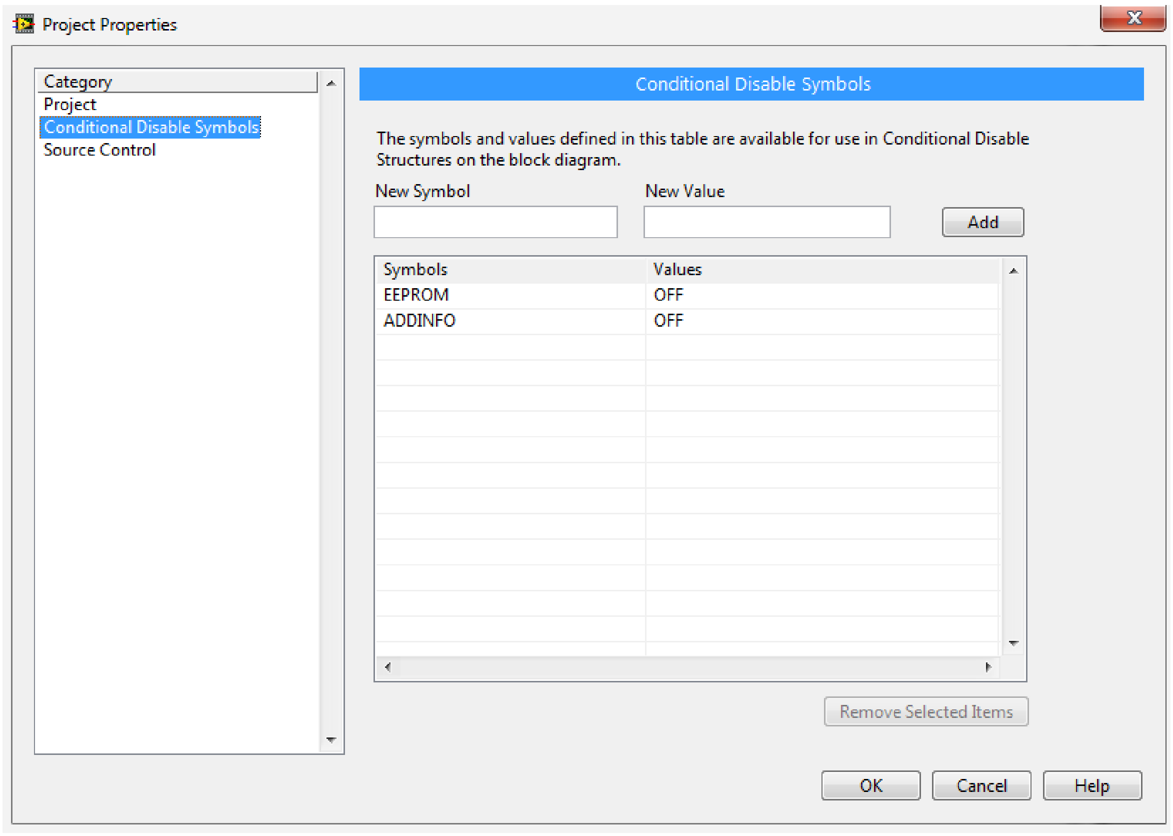
However, the current driver software architecture offers a way that can help to reduce the FPGA space further if FPGA space is precious. Within the SEA 9510 driver core certain functional blocks can be disabled using *Conditional Disable Symbols*. A disabled functional block is not compiled into the bitfile and therefore does not consume FPGA space. When using *Conditional Disable Symbols* some API functions become non-executable and must not be used to prevent unexpected behavior.

Please refer to the table below for details on using this optimization feature:

Conditional Disable Symbol	**Value Range**	**Description**
EEPROM	OFF ON (default)	OFF disables the following nodes: • Mod ID • Mod SerialNumber • Mod VendorID
ADDINFO	OFF ON (default)	OFF disables the complete Additional Information functionality

Tab. 9: Conditional Disable Symbols

The *Conditional Disable Symbols* can be defined in the project property dialog box like shown in Fig. 7.


Project Properties

Category
Project
Conditional Disable Symbols
Source Control

Conditional Disable Symbols

The symbols and values defined in this table are available for use in Conditional Disable Structures on the block diagram.

New Symbol
New Value

Symbols	Values
EEPROM	OFF
ADDINFO	OFF

Fig. 7: Project Properties – Conditional Disable Symbols

This dialog box is accessible by right-clicking on the project root item “Project: *‹project name›*” in the LabVIEW project explorer. If no conditional disable symbols are defined the default values are used (EEPROM=ON; ADDINFO=ON).

7 Deployment

If a user application containing the SEA 9510 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 detected within LabVIEW	1. Make sure you have the CompactRIO system powered and connected to your PC. 2. Make sure you have installed the current EnDat driver software (≥ 2.0.0) after de-installing the previous driver. 3. Make sure that the module is detectable in the LabVIEW project tree. Contact S.E.A. if detection fails to check if a firmware (eeprom) update is required.
2	The **Init** method node returns an error	1. Check if you are using fully functional EnDat encoders from *HEIDENHAIN*. 2. If you are using self confectioned cables check the pin-out and contact.
3	The **Init** method node only works with low *Bus Speed* setups	Check your cables and the overall cable length. The specified max. cable length is only possible with officially classified *HEIDENHAIN* cables and connectors.
4	The IO nodes suddenly stop delivering the position data	Check all connectors and encoders. If at least one connector stops responding properly, the Init method node has to be called again in order to recover. If the error occurs again, enable the error terminals in the IO node and indicate the error status. Additionally, the encoder memory can be read to look at the errors and warnings, refer to the *ReadEncoderMemory* example.
5	No additional information can be read	Make sure that the *Enable Add Info* input of the *Init* node is set to true during initialization.
6	Upgrading from driver version 2.0 to 2.1 throws internal error #65221 when trying to build the FPGA code.	Due to an LabVIEW bug, the *Init* method node is not properly updated. Re-select the *Init* method by choosing "Select Method –> Init" in the context menu. Afterwards, properly re-connect all input nodes.

A Figures

B Tables