SAIA S-Bus AX + N4
Driver Guide

Version V1.4

Document information

Document name:	Niagara AX SAIA SBus Driver Guide-en.docx
Directory:	P:\4 Intern\04 Produkte\02 saturnMIS\11 Dokumentation\01 Leitsystem
Status	Freigegeben
Author:	RDS / R

Changes

Version	Date	Abbr.	Change
V1.0	03.11.2010	fch	Document created
V1.2	28.05.2015	fch	Complete to match German version
V1.3	27.06.2016	fch	New address
V1.4	12.08.2019	fch	Supported Niagara Version added

Content

+41 (0)44 261 00 70 info@alvasys.ch / www.alvasys.ch alvasys automation ag / Hermetschloostrasse 75 / CH-8048 Zürich

Figures

Tables

+41 (0)44 261 00 70 info@alvasys.ch / www.alvasys.ch alvasys automation ag / Hermetschloostrasse 75 / CH-8048 Zürich

1 Saia S-Bus Driver installation

1.1 License requirement

To use the Saia S-Bus driver, you must have a target Niagara host (JACE or Supervisor) that is licensed with the feature `saiaSbusIp`.
In addition, note that other limits on devices and proxy points may exist in your license.
The following parts are available:

- PI-DR-SBUS-500 Driver for S-Bus IP 500 Points
- PI-DR-SBUS-ADD500 Driver for S-Bus IP Additional 500 Points
- PI-DR-SBUS-DEMO Driver for S-Bus IP Demo

1.2 Niagara version requirements

The followed Niagara versions are supported and tested:

- AX 3.7
- AX 3.8
- N4 4.4

Other versions may work but are not tested.

1.3 Installation

From your PC, use the Niagara Workbench 3.*n.nnn* to install the module `saiaSBusDriver` in the host.
For details, see "Software Manager" in the *Platform Guide*.
Following this, the remote JACE is now ready for Saia S-Bus configuration in its running station, as described in the rest of this document.

2 Saia S-Bus Driver Quick Start

2.1 Configure the Sbus Driver Network

To configure the Sbus Driver Network, perform the following tasks:

Add a Sbus Driver Network

Note: *As an alternative to the procedure below, you can simply copy (drag and drop) a* `SbusDriverNetwork` *from the* `saiaSBusDriver` *palette in Workbench, placing it in the station's Config, Drivers container.*

Use the following procedure to add a `SbusDriverNetwork` under the station's Drivers container:

Step 1 Double-click the station's Drivers container, to bring up the `Driver Manager`.
Step 2 Click the `New` button to bring up the New Device Network dialog. For more details, see "Driver Manager New and Edit" in the Drivers Guide.
Step 3 Select "`SbusDriverNetwork`", number to add: 1, and click `OK`. This brings up a dialog to name the network.
Step 4 Click `OK` to add the `SbusDriverNetwork` to the station.

You should have a `SbusDriverNetwork` named "`SbusDriverNetwork`" (or whatever you named it), under your Drivers folder.

Configure the Sbus Driver Network

If the used Saia station use another port than 5050, you must define the used UDP port.

To configure this property, use the following procedure:

Step 1 Right-click the `SbusDriverNetwork` and select `Views > Property Sheet`. This produces the network's property sheet.
Step 2 Expand the node `Udp Config` and expand the node `Address`.
Step 3 You can leave the `Port` field to its default value (5050) or set the port you want to use.

+41 (0)44 261 00 70 info@alvasys.ch / www.alvasys.ch alvasys automation ag / Hermetschloostrasse 75 / CH-8048 Zürich

2.2 Add Sbus Driver Device

After adding a Sbus Driver Network, you can use the network's default device manager view to add the appropriate Sbus Driver devices.

To discover Sbus Driver Devices

The Sbus Driver doesn't support discovering devices.

To add a Sbus Driver device in the network

Use the following procedure to add a Sbus Driver device in the network.

Step 1 In the Nav tree or in the Driver Manager view, double-click the network, to bring up the device manager.
Step 2 Click the `New` button to bring up the New device dialog.
Step 3 Select for number to add: 1 (or more, if multiple) and click `OK`.
This brings up a dialog to name the device(s), enter the IP-address and the UDP Port. If you know it, you can enter the station address.
Step 4 When you have a Sbus Driver device component configured properly for your usage, click `OK`.
The Sbus Driver device is added to the station and appears listed in the Database pane.
Step 5 Bring up the property sheet for the added device and review the properties.

Discovering the station address

Maybe you don't know the station address. You can discover it, providing that the station is available. Use the following procedure to discover the station address:

Step 1 In the Nav tree or in the Driver Manager view, right-click the device, to bring up the context menu.
Step 2 Click the `Actions` menu.
Step 3 Click in the `Actions` submenu the menu item `Sync Station Address`.
This will send a broadcast request to the station and refresh the station address.

2.3 Create Sbus Driver proxy points

As with device objects in other drivers, each client Sbus Driver device has a `Points` extension that serves as the container for proxy points. The default view for any Points extension is the Point Manager (and in this case, the "`Sbus Driver Point Manager`"). You use it to add Sbus Driver proxy points under any Sbus Driver device. For general information, see the "About the Point Manager" section in the *Drivers Guide*.

Online Discover to add Sbus Driver proxy points

The Sbus Driver doesn't support online discovering of points.

Discover Visi+ Symbol file to add Sbus Driver proxy points

The `Sbus Driver Point Manager` supports discovering points from Visi+ Symbole File. This file can be exported with the software PG5. A Visi+ Symbole File is a CSV file with the following structure:

```
SYMBOL;MEDIA;ADDRESS;COMMENT;SCOPE;TAGS
BlinkerSymEin;F;1000;Blinker symetrisch aktivieren;0;
BlinkZeitSym;R;1000;Blinkzeit symetrisch;0;
BlinkSignalSym;F;1100;Blinksignal symetrisch;0;
Heizkurve;R;2000 [7];Heizkurve;0;
```


+41 (0)44 261 00 70 info@alvasys.ch / www.alvasys.ch alvasys automation ag / Hermetschloostrasse 75 / CH-8048 Zürich

Discover Vision Symbol File to add Sbus Proxy points

The Sbus Driver Point Manager supports discovering points from Visi+ Symbol File. This file can be exported with the software PG5. A Visi+ Symbol File is a CSV file with the following structure: (Separator is <TAB>):

```
SYMBOLNAME      →   TYPE    →   ADDRESS   →   TAGS   →   COMMENT
BlinkerSymEin   →   F       →   1000      →         →   Blinker symetrisch aktivieren
BlinkZeitSym    →   R       →   1000      →         →   Blinkzeit symetrisch
BlinkSignalSym  →   F       →   1100      →         →   Blinksignal symetrisch
Heizkurve       →   R       →   2000 [7]  →         →   Heizkurve
```

Notices for discovering points

The columns SYMBOL, MEDIA and ADDRESS must contains a value to get the points discovered. The value of the column SYMBOL is serving as point name. The value of the column MEDIA is needed to determine the „Variable Access Type“. Only supported types are discovered. The column ADDRESS has to contain a number or a number followed with “[x]”. This is an address range. In the sample above the line “Heizkurve” will discover 7 points. The string “[x]” will be append to the names, in which x begins with 0.
Basically all types can be writable except the type “input”. “Text” points cannot be written. The information Die Information, if a point can be written or not depends from the software in the SAIA controller and is not included in the Symbol Files. Therefore, it is the responsibility of the user to select the correct type.

The discovered points can be filtered by name and comment. This can be useful with larger import files. The filter has to be specified as a regular expression. Regular expressions allow complex filtering operations that are nor accessible by normal wildcards.

Sample:

```
.*Heiz.*
→ All names containing the string „Heiz“.

.*Data([1-9]|(1[0-6]))
→ All names ending with Data and a number from 1 to 16.
For example GugusData12 or Data1.
Data0 or Data17 don’t match the filter.

Blink.*
→ All names beginning with Blink.
```

For other examples or tutorials, see the literature and on the Internet.

Manually adding Sbus Driver Proxy points

You can manually add Sbus Driver proxy points, using the New-button in the Sbus Driver Point Manager, or by dragging from the saia SBus Driver palette.

Step 1	Brings up the Sbus Point Manager.
Step 2	Click the New button. The New points dialog appears, in which you select a point “Type” and “Number to Add”. Click OK.
Step 3	Another New dialog appears, where you can name and configure the new points. ***Notice**: You can choose every point type. But not every type is supported by Sbus.*
Step 4	Click OK to add the proxy point(s) to the Points extension (or to the current points folder), where each shows as a row in the point manager.

+41 (0)44 261 00 70 info@alvasys.ch / www.alvasys.ch alvasys automation ag / Hermetschloostrasse 75 / CH-8048 Zürich

3 S-Bus and Niagara

The Saia Sbus Driver supports IP communication over UDP. A serial communication is not implemented.

3.1 Floating Point Format

The floating point numbers are based on 32 Bits and have the following format:

Byte																															
0								1								2								3							
m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	s	e	e	e	e	e	e	e

Legend:

m	→	Mantissa
s	→	Sign
e	→	7-bit exponent in excess-64 notation

In Niagara number are stored as double. The type double in Java is similar, but is based on 64 bits. The Saia floating point format is also different from Java Float format.
The conversion from Saia-floating point numbers to double so can cause perceived inaccuracy losses, although a Double is in theory "precisely" as a float.

Sample:

Number 1.255
Binary representation as „Saia Floating Point Format“:

```
10100000 10100011 11010111 01000001
```

Binary representation as „Java Float Format“:

```
00111111 10100000 10100011 11010111
```

Binary representation as „Java Double Format“:

```
00111111 11110100 00010100 01111010 11100000 00000000 00000000 00000000
```

By aligning the float and double formats, you realize what is happening during the conversion. The Double is "topped up" with 0 bits. It thus follows, depending on the number, a supposed loss of accuracy.

```
       s eeeeeeeeeee mmmmmmmmmmmmmmmmmmmmmmmmmmmmmm mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
Saia   0     1000001 10100000101000011110101111
Float  0    01111111  01000001010001111010111
Double 0 01111111111  01000001010001111010111 00000000000000000000000000000000
```

The number above is as Java Double Format `1.2549999952316284`
This difference is normal and not an error from driver.

3.2 Variables access

The variables are read from the Saia station in a polling process. Write accesses are executed immediately.

+41 (0)44 261 00 70 info@alvasys.ch / www.alvasys.ch alvasys automation ag / Hermetschloostrasse 75 / CH-8048 Zürich

3.2.1 Media types

In the tables below are the possible media types (Saia types) that can be read or written from a Saia station via S-Bus. The last column indicates whether these types can read grouped. Write accesses are never performed grouped.
The media type must be set in the SBus Driver Proxy Ext component with the **`Variable Access Type`** property.

Legend:

X → Supported
(X) → Supported, but not rational
empty → Not supported

Saia Type	Format	Niagara Type	Read	Write	Group
Register	32 Bits Integer	Numeric	X	X	X
		Boolean	X		
		Enum	X	X	
		String	X		
Register Float	32 Bits Saia Floating Point	Numeric	X	X	X
		Boolean	X		
		Enum	X	(X)	
		String	X		
Counter	32 Bits Integer	Numeric	X	X	X
		Boolean	X		
		Enum	X	(X)	
		String	X		
Timer	32 Bits Integer	Numeric	X	X	X
		Boolean	X		
		Enum	X	(X)	
		String	X		
Flag	1 Bit Boolean	Numeric	X	X	X
		Boolean	X	X	
		Enum	X	X	
		String	X		
Input	1 Bit Boolean	Numeric	X		X
		Boolean	X		
		Enum	X		
		String	X		
Output	1 Bit Boolean	Numeric	X	X	X
		Boolean	X	X	
		Enum	X	X	
		String	X		
Real Time Clock	8 Bytes (64 Bits)	Numeric	X	(X)	
		Boolean			
		Enum			
		String	X	(X)	
Display Register	32 Bits	Numeric	X		
		Boolean	X		
		Enum	X		
		String	X		
Text	ASCII-Chars	Numeric			
		Boolean			
		Enum			
		String	X		

Table 1: Media Types und supported formats

3.2.2 Conversion between different types

Since different Saia-types can be read or written by various types of points, the driver converts the values according to the following rules:

+41 (0)44 261 00 70 info@alvasys.ch / www.alvasys.ch alvasys automation ag / Hermetschloostrasse 75 / CH-8048 Zürich

Saia → Niagara

Number	→	Numeric

The Media Register, Register Float, Counter, Timer, Display Register, Flag, Input and Output are converted to Double.

Number	→	Boolean

Die Media Register, Register Float, Counter, Timer, Display Register, Flag, Input and Output are converted to Boolean. For that the `long` value of Number is evaluated.

```
Value != 0   →    true
Value == 0   →    false
```

Number	→	Enum

Die Media Register, Register Float, Counter, Timer, Display Register, Flag, Input and Output are converted to Enum. For that the `int` value of Number is evaluated.

Number	→	String

For this conversion the `toString()` function is used.

Real Time Clock	→	Numeric

The result is the number of milliseconds since 01.01.1970 (compare. `BAbsTime.millis()`).

Real Time Clock	→	String

The result is the same as the function `BAbsTime.encodeToString()`.

Niagara → Saia

Numeric	→	32 Bits Integer

The double value is casted in a long value. This means that the decimals are truncated.

Numeric	→	32 Bits Saia Floating Point

The double value is first casted in a float value (Java Type Cast) and then converted in a Saia Floating Point. This can result in a loss of accuracy (see above).

Numeric	→	1 Bit Boolean

The double value is casted in an integer value. Only the first bit is written.

Numeric	→	Real Time Clock

It is possible to set the Real Time Clock. The number in the Numeric is the number of seconds since 01.01.1970.

Boolean	→	1 Bit Boolean

```
true   →    1
false  →    0
```

Enum	→	32 Bits Integer, 32 Bits Saia Floating Point

No conversion. The value is the same as in an Enum (Ordinal).

Enum	→	1 Bit Boolean

The Ordinal value is used. Only the first bit is written.

+41 (0)44 261 00 70 info@alvasys.ch / www.alvasys.ch alvasys automation ag / Hermetschloostrasse 75 / CH-8048 Zürich

String	→	Real Time Clock

Die String is evaluated by the function `BAbsTime.make()`. See the help of `BAbsTime` for more information.

3.2.3 Important notes for write access

In the Niagara Framework writable points perform under certain conditions a write command.
Certain conditions can be adjusted by `Tuning Policies`. These are:

- Write On Start
- Write On Up
- Write On Enabled

Other conditions cannot be selected:

- Changing the points Facets (e.g. unit from the point)
- Changing the device Facets (e.g. unit in the Saia-device)
- Changing the conversion (linearization, inversion, etc.)

This is a default behaviour, and applies to all writable points. However, this has important implications for media such as `timer` or `real time clock`. Suppose the user performs the "Set Action" to set a value in a timer. The value is written and the timer starts and runs some time off. If the point facets are then changed, the last written value is rewritten in the timer. It will then start again.

***Note**: to set the time in the Saia-station, the actions on the device can be used (instead of a writing spot). See section 3.3.*

3.2.4 Grouping

Settings of Sbus Driver Proxy Ext

To enable grouping the property `Grouping Enabled` of the `Sbus Driver Proxy Ext` has to be set to `true`. The media supporting grouping are listed in the Table 1. If the property `Grouping Enabled` is set to `true` on a not supported type, this setting is simply ignored.

Poll Group Entry

To read points grouped you have to insert `Poll Group Entries` in the `Poll Group Config` component of the device. You can insert one or more components from the palette.

These properties have to be set on Poll Groups:

Property	Meaning	Default
Enabled	Set the grouping group enabled or disabled. All subscribed points are resubscribed on change.	false
Variable Access Type	This is the Variable Access Type for the group. Only proxy extension with this variable access type are read from this group.	Register
Address	Start address of the group	0
Count	Number of the variables to be read minus 1.	0
Poll Frequency	Poll frequency of this group. Only proxy extension with this poll frequency are read from this group.	normal

Table 2: properties of Poll Group Entry component

+41 (0)44 261 00 70 info@alvasys.ch / www.alvasys.ch alvasys automation ag / Hermetschloostrasse 75 / CH-8048 Zürich

Allocation of the extensions to the Poll Groups

The proxy extensions looking off into the Poll Group listing of the group to which they belong.
Die criteria for the search am mapping are:

- Variable Access Type
- Address (address of the point must be within the range that is read by the Poll Group)
- Poll frequency
- Poll Group enabled (`enabled` property must be `true`)
- Poll Group Status (must be `{ok}`)

If, based on these criteria, no Poll Group are found, the point is quite read normal (i.e. not grouped).

The following illustration is intended to show how the group membership based on the address takes place.

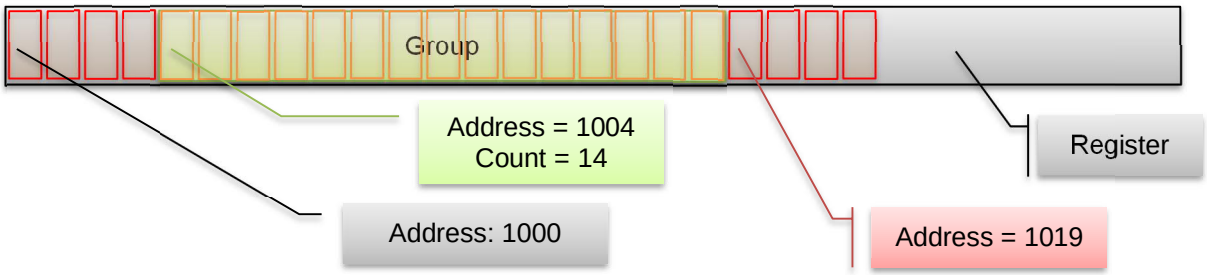

Figure 1: group membership by address

***Notes to the figure above**: points with addresses from 1004 to (and including) 1018 (= 1004 + 14) are read on the group. A point with address 1019 is read individually and not of the group.*

Limitations

The Poll Group component has a status that is information about the condition of the group.

Possible reasons for a `{fault}` Status:

- Media type does not support grouping
- The maximum number of variables (Count property) has been exceeded
  - For Input, Output and Flag → Max Count = 127
  - For Register, Register Float, Timer and Counter → Max Count = 31

Optimization

By grouping the reading can be accelerated because several variables are packed into a telegram. It can therefore make sense to group variables, that are "together", or which are repeated on these screens.

Examples:
In the examples below it is demonstrated that a group does not necessarily mean an optimization. In the first example 32 variables are read, that are located next to each other in the register area. In the second example, only 2 variables to read, but with a gap between two variables.

- 32 Register, from address 1000 to 1031
- Poll Group settings: Address = 1000, Count = 31

	Grouped	**Not grouped**	**Saving**
Request	1 telegram x 16 Bytes	32 telegrams x 16 Bytes = 512 Bytes	496 Bytes
Response	1 telegram x 139 Bytes	32 telegrams x 15 Bytes = 480 Bytes	341 Bytes
Total	155 Bytes	992 Bytes	837 Bytes 84.375 %

Table 3: Grouping vs. individually read example 1

- 2 Register, address 1000 and address 1031
- Poll Group settings: Address = 1000, Count = 31

	Grouped	Not grouped	Saving
Request	1 telegram x 16 Bytes	2 telegrams x 16 Bytes = 32 Bytes	16 Bytes
Response	1 telegram x 139 Bytes	2 telegrams x 15 Bytes = 30 Bytes	-109 Bytes
	155 Bytes	62 Bytes	-93 Bytes 150 %

Table 4: Grouping vs. individually read example 2

In the second example, the amount of data in an array increases by a total of 150%. In this case, it would be more efficient to read the two variables individually.

3.3 S-Bus Device

3.3.1 System Information

The `System Information` of the PCD can be read with an action (`Read System Information`). The following values are read and set this in the property `System Information` in the device:

Program Information

The property Program Information is a multiline text that contains various information about the program. Example:

```
[Program]
PG5Licensee=DEMONSTRATION VERSION
PG5DeveloperID=CH_ChFr1070
Originator=DEMONSTRATION VERSION
PG5Version=V2.1.310
ProjectName=Test-SBus
DeviceName=Station
PcdType=PCD1.M2120
ProgramVersion=1.0
ProgramID=7225FBDAD786E3DC
ProgramCRC=30980039
BuildDateTime=2014/07/24 13:19:17
DownloadDateTime=2014/07/24 18:16:18
```

Memory Information → System Information 1

Eigenschaft	
Ct Size	Is size of the Code/Text memory device (in kbytes)
Ex Size	Is size of extension memory (in kbytes)
Memory Type	Is memory type: 0 = no memory 1 = RAM 2 = EPROM 3 = FLASH
Memory Continuous	(PCD2.M170,…)
Code Text Password Protected	
Extension Memory Ram Present	
Onboard Backup Size	Is the onboard backup partition size in Kbytes
Rem Code Text Mem Size	Is the remaining Code/Text memory size in bytes
Rem Extension Mem Size	Is the remaining Extension memory size in bytes
RemPrecompMemSize	Is the remaining Pre-Compiler memory size in bytes
Program Block Header Size	Program Block HeaderSize in bytes in the FW
Rom Text Db Header Size	ROM Text/DB(x) HeaderSize in bytes in the FW
Ram Text Db Header Size	RAM Text/DB(x) HeaderSize in bytes in the FW

Table 5: System Information 1

Trace Buffer Information → System Information 2

+41 (0)44 261 00 70 info@alvasys.ch / www.alvasys.ch alvasys automation ag / Hermetschloostrasse 75 / CH-8048 Zürich

Eigenschaft	
Trace Buffer Size	is the size in bytes of the trace buffer

Table 6: System Information 2

F5/7 EEPROM Information → System Information 3 + 4

Eigenschaft	
Eeprom Cs	EEPROM_CS
Length Of Data	length of data (without CS+length)
Eeprom Type	Type (in ASCII)
Version	Version (in ASCII)
Modification	Modification
Fab Date	Fab. Date (Jahr/Woche)
Fw Version	FW Version
Mac Address	MAC Address

Table 7: System Information 3 + 4

Possible Baud Rates → System Information 5

Eigenschaft	
Baudrate xxxxx	Baudrate

Table 8: System Information 5

Onboard Information → System Information 6

Eigenschaft	
Serial Number	Seriennummer
Mac Address	MAC Address
Cpu name	CPU Name
Fw Version	FW Version
Version Modification	Version/Modifi
Fab Date	Fab. Date (Jahr/Woche)

Table 9: System Information 6

SAIA NT Information → System Information 7

Noch nicht unterstützt.

Slot (IO , Ext, M1, M2) Information → System Information 8

Noch nicht unterstützt.

3.3.2 Ping

The Sbus Driver Devices have a Ping function. The Ping function read the status of the Saia PCD. The Ping result is ok if a valid answer is received. This is independent of the read status. A NAK answer is also a valid answer.

3.3.3 PCD-Status

The PCD status is polled to and the property `Pcd Status` set.
Possible status are:

Status	Bedeutung
Serial Interface	??? Serial Interface, Serial Port
Halted	Halted
Running	Running
Conditional Running	Wait breakpoints
Disconnected	Processor doesn't exist
Exceptional Intermediate Running	??? For more details see the documents of the project 'MODEMS+'
Cannot Get Status	This is not a "real" PCD-Status. Set on read errors or on ACK/NAK answer.

Table 10: PCD-Status

3.3.4 System Program Version

The `System Program Version` of the PCD can be read with an action (`Read System Program Version`).
The following values are read and set this in the property `System Program version`:

+41 (0)44 261 00 70 info@alvasys.ch / www.alvasys.ch alvasys automation ag / Hermetschloostrasse 75 / CH-8048 Zürich

- Module Type
- Version
- CPU Number

3.3.5 Write System Time

The Sbus Driver Devices have two actions to write System Time in the Saia station.

Set Clock

With this action the user is promped to enter a time. The time of the PCDs is set with this time.

Set Clock From System Time

With this action the current system time of the Niagara station is set in the PCD.

+41 (0)44 261 00 70 info@alvasys.ch / www.alvasys.ch alvasys automation ag / Hermetschloostrasse 75 / CH-8048 Zürich