# Frequently Asked Questions

## -Seskion GmbH-

Content:

- Table of Content
- General Questions
- PSI5-Simulyzer specific Questions
- DSI3-Simulyzer specific Questions
- SPI-Simulyzer specific Questions
- SENT-Simulyzer specific Questions
- Application Notes

| | |
|---|---|
| Version: | (1.0) 30.08.2022 – Creation |
| | |
| | |
| | |

# FAQ

## Table of Content

# FAQ

## General Questions

Is there an option to add second channel later on?

To add the second channel later is no problem. Therefore you don't have to send it back to us. We can implement it later with a license. Only the CAN interface has to built in before shipping.

What is the accessories inside the package? Connector, wires?

The DLL API is included and we are shipping the API documentation. You can connect to the Simulyzer-Box with a simple Sub-D Connector.

Is there a possibility to test the Simulyzer on loan and validate it for our application area?

There is of course the possibility to borrow a Simulyzer. However, there would be a rental fee per month. In return you get the Simulyzer box with two channels and the Windows software license. If you need an additional CAN interface, we can also fulfill this wish.

We need to read the serial number from a sensor and check the continuity of the wiring harness.

With the Simulyzer you can read the serial number and check the continuity of wire harness. For an end of line test we have already written software for other companies for the identical tests, which we can offer you with our know-how. For this we would of course need further details which could also be discussed in a meeting. Otherwise you can use our API for your own programmed Software.

What are my options to send serial number to PLC?

It is possible to send it via CAN. If a CAN interface is available, we can do the PLC control over the CAN bus. But therefor we need to programm our box for the specific PLC connection. So we have to offer you software development costs.

What is the option when we don’t use CAN bus?

In principle, you have all the options. You can use USB, Ethernet and more. You can then try to program your connection yourself with our API, or we can program it for you with your required connection. We have the know-how and know the API very well.

Does the CAN port work the same way as the USB port? Is the CAN port used to control communication with the PC?

The USB port cannot be exchanged with the CAN port. With the CAN port only further features like the PSI5-to-CAN communication are possible. However, this can only be done once the Simulyzer has been properly configured with the PC via USB connection. With the CAN connection data can be fed in, but this does not replace the communication via USB connection with the PC.

How long are the individual software licenses valid?

The licenses are valid indefinitely. If you have activated a license once, it is valid forever for the time being.

What is the meaning of Simulyzer-Box or PC bunded license?

You can choose between these options. If it is box bound, you can work with the Simulyzer on multiple PCs. If it is PC bound, you can work with various Simulyzers at one PC. For a better understanding there is an application note.

Do you have some training options for the Seskion-Simulyzers?

We would like to give you the opportunity to offer a training with our distinguished engineer. For which areas do you need a training? We offer the following courses:

- Simulyzer GUI Basic Overview: 2h
- Simulyzer GUI Advanced Configuration: 4h
- Simulyzer Sensor Programming: 4h

These are not all our courses, we can also create a customized program for you. The cost of the training is 500€ per hour. The number of participants is not taken into calculation of the price. We do not have staggered prices associated with the number of participants. Normally, groups with a size of 5+ participants have always participated.

How to check the software license by Custom officer?

You get a license sheet with a license code on it. This is in the package. Please do not be confused by the sentence "Not in package, has to be downloaded from Seskion web-page." The software is available for download from our website. Without the license code, the software cannot be used. Similar to a Windows license that you buy from Microsoft. This certificate is always included.

Is the Simulyzer box only functional with an AFP sample from the corresponding supplier?

In order to emulate the central sensor system, you need access to the internally installed sensors. This is only possible with the AFP patterns of the ECUs.

Are we independently responsible for obtaining an AFP sample?

Yes.

Is the Simulyzer box compatible with each manufacturer or must adapters/HW/couplers be ordered at the same time?

The box itself is compatible because we only simulate the sensors and not entire ECUs. For other ECUs the box must be reconfigured. But we offer adapters for the direct connection of the AFP-ECU's with our Simulyzer. These are also different for ZF and Conti due to the constructional differences of the ECU's.

How is the connection from an operating computer/simulator to your box?

The boxes are operated via USB by default. For the HIL stands, a so-called standalone mode was implemented in which the boxes are first configured via USB and can then be operated autonomously with a 12V supply. In this case, the sensor data can then be sent to the boxes via CAN(classic , i.e. no FD).

# FAQ

## Is additional SW/driver needed for the operation of the Seskion-Box?

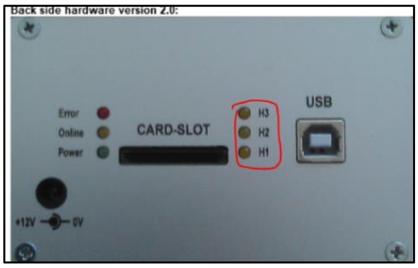
To configure the boxes, at least one PC is required on which our GUI software is installed. This software can be installed and operated free of charge for the configuration process. We recommend to purchase at least one GUI license in order to be able to quickly track down problems in case of troubleshooting.

## Is it possible to sim-/stimulate the external crash sensors of the Airbag-ECU with your solution?

For this the PSI5-Simulyzer is used which supports 2 physical PSI5 interfaces with up to 4 timeslots. So clearly, yes we can, but with different boxes.

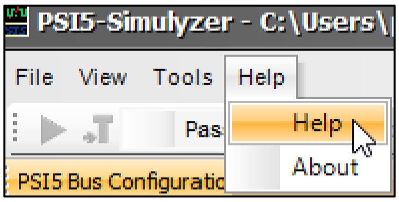
## How is the H1, H2, H3 meaning on the Box?

The H1 and H2 lamps are not configured. H3 shows the sample enabled, i.e. when you switch on power in the GUI.



## How to use the Seskion-Simulyzer?

First of all I can share some Application Note for a little bit better understanding of the software. Also you can look into the Help menu in the software. There are a lot of things explained. If both doesn't help we can maybe do an online meeting. And if you have some questions that we can clarify via e-mail, please do not hesitate to contact me.



## How to send your acquired Simulyzer data to somebody else?

It is very easy to send your acquired data to somebody else. After stoping your acquisition, just click on „File“=> „Save“. This will place several acquisition files to the directory where your .ppf project file is located. All files have the same basename but different extensions. For example:

- myProject.ppf (the project file)
- myProject.rawData (your physical acquisition of PSI5- voltage and current)
- myProject.sensorData (your decoded comunication data)
- myProject.sensDatFile0
- myProject.triggerData
- myProject.triggerData_1
- and maybe even more ….

For sending the acquisition, please zip all files with the same basename (including the .ppf) and send it via e-mail or whatever. The recipient may de-zip the files and place it in a new directory. Then he just may open the myProject.ppf file from within our GUI app and the app looks if it finds the other files. If so, the data will be available for examination, same if possibilities as if self acquired.

How to operate several Simulyzers parallel in Sensor mode?

Problem: If you have the situation to simulate lots of sensors and you need several PSI5-Simulyzers or SPI-Simulyzeres or a mix of them, you have to do lots of API calls for each Simulyzer. You have to open several connections, prepare all the configurations, start the simulation for all these Simulyzers and so on. Luckily there is a small set of API calls that operate on s so called „system configuratiom“ (described below). This is a xml file describing all the connected Simulyzers, their configurations and their Simulation stream data. Start and stop of the Simulyzesr can be in a call covering all Simulyzers.

```
SIMULYZER_API _tSIMU_retVal SimulyzerSystemCreate(HANDLE *hSystem, const char *systemConfigurationFile);
SIMULYZER_API _tSIMU_retVal SimulyzerSystemDestroy(HANDLE hSystem);
SIMULYZER_API _tSIMU_retVal SimulyzerSystemGetDevices(HANDLE hSystem, int *count, HANDLE *hDevs);
SIMULYZER_API _tSIMU_retVal SimulyzerSystemSetSignalData(HANDLE hSystem, int count, int *sampleCountArray, double
**sampleBufferArray, int resetToDefaultValue);
SIMULYZER_API _tSIMU_retVal SimulyzerSystemSetSignalDataFromCSV(HANDLE hSystem, const char *csvFileName, int
resetToDefaultValue);
SIMULYZER_API _tSIMU_retVal SimulyzerSystemStartStream(HANDLE hSystem);
SIMULYZER_API _tSIMU_retVal SimulyzerSystemStartSampling(HANDLE hSystem);
SIMULYZER_API _tSIMU_retVal SimulyzerSystemStopSampling(HANDLE hSystem);
```

Explanation of API calls:

- SimulyzerSystemCreate(); This creates a system configuration by opening the system configuration xml file passed in parameter const char *systemConfigurationFile . The call returns a HANDLE *hSystem to the system configuration. All connection to the Simulyzer boxes defined insid the system configuration xml file are tried to open.
- SimulyzerSystemDestroy(); This is the counterpart to SimulyzerSystemCreate(), . It closes all Simulyzer connections and releasess all its resources. After this call the HANDLE hSystem must not be used any more!
- SimulyzerSystemGetDevices(); Occasionally you want to do some calls to only one of the connected Simulyzers. In This case you need the device-handle of the Simulyzer. This call returns an array of device-handles in HANDLE *hDevs. The number of valid device-handles is returned in int *count. You may use hDevs[0] up to hDevs[count-1].
- SimulyzerSystemSetSignalDataFromCSV(); is used to load the signal data-streams of all simulated signal traces out of one single csv file.
- SimulyzerSystemSetSignalData(); is used to load the signal data-streams of all simulated signal traces out of two-dimensional array to be created by the application and provided to the API call. This may be useful if the data comes out of a database and your app has to create the signal data-streams by filling the array with database values.
- SimulyzerSystemStartSampling(); starts the Simulyzer sampling machine of all Simulyzers defined in the system configuration
- SimulyzerSystemStopSampling(); stops the Simulyzer sampling machine of all Simulyzers defined in the system configuration
- SimulyzerSystemStartStream(); starts the Simulation stream of all Simulyzers defined in the system configuration

Python Example:
Here is an example created in Python script language (May also serve as example how to use the Simulyzer API from Python).

```
from ctypes import *
import os
from SimulyzerConstants import *
import threading

rdy_flag = threading.Event()

def py_callcack(v_p):
    print "Stream complete"
    rdy_flag.set()
    return

CMPFUNC = WINFUNCTYPE(None,c_void_p)
cb_func = CMPFUNC(py_callcack)
callback_ctx = c_void_p()


#load the simulyzer library
os.chdir("./tmp")
hnd = cdll.LoadLibrary("../Simulyzer_64.dll")
print hnd

#create a system handle from system configuration file
syshnd = c_void_p()
sekRetVal =
hnd.StartLogging("simu.log",c_int(LoggingWhat['TRACE_APICALLS']+LoggingWhat['TRACE_ERROR']),c_int(LoggingHow['MODE_REOPEN']+Lo
ggingHow['MODE_CONSOLE_OUT']))
print "StartLogging return code: " + str(sekRetVal)

sekRetVal = hnd.SimulyzerSystemCreate(byref(syshnd),"../ExampleSystemConfigCSVImport.xml")
print "SimulyzerSystemCreate return code: " + str(sekRetVal)
#read back the actual count of initialized simulyzer devices (only for testing)
devCount = c_int(10)
devHnd = (c_void_p * 10)()
hnd.SimulyzerSystemGetDevices(syshnd,byref(devCount),pointer(devHnd))
print devCount
sekRetVal = hnd.SimulyzerSystemSetStreamEndCallback(syshnd,callback_ctx,cb_func)
print "SimulyzerSystemSetStreamEndCallback return code: " + str(sekRetVal)


#set the data for streaming
hnd.SimulyzerSystemSetSignalDataFromCSV(syshnd,"../data/2kHz.csv",1)
#hnd.SimulyzerSystemSetSignalDataFromCSV(syshnd,"../data/crash1.csv",1)
#set trigger signal
raw_input("return to continue")
rdy_flag.clear()
hnd.SimulyzerSystemStartStream(syshnd)
rdy_flag.wait(2.0)
print rdy_flag.isSet()

#cleanup all simulyzer stuff
hnd.SimulyzerSystemDestroy(syshnd)
```

Explanation of Python code

- Line 1 .. 4: all needed imports
- Line 6: rdy_flag is used for synchronization between callback (API internal thread context) and the main thread
- Line 8 .. 11: callback function being called when simulation stream is used up. It sets rdy_flag in line 10
- Line 13 .. 15: definition being needed for callback
- Line 19 .. 21: binding to Simulyzer_64.dll
- Line 24: instantiate a variable to hold the handle for the system configuration
- Line 25 .. 26: start logging (with useful debugging info in textfile simu.log)
- Line 28 .. 29: open the system configuration xml-file and place handle in variable syshnd (being instantiated in line 24)
- Line 31 .. 34: get device handle in devHnd and number of device handles in devCount. Never used in this example
- Line 35 .. 36: link „StreamEndCallback“ to callback function cb_func
- Line 40 .. 41: loading the signal data from csv file
- Line 43: just wait for a console „Return“ entry for starting the simulation stream
- Line 44: clear synchronization flag rdy_flag (will be set by callback after stream finishes.
- Line 45: real start of simulation stream
- Line 46: wait for synchronization flag rdy_flag being set at stream end
- Line 50: cleanup, close the system configuration.
- End

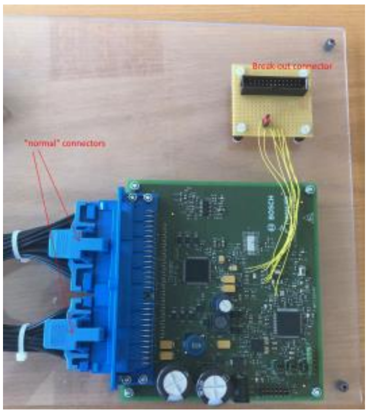
Download complete example:
Here is, as Zip-file, an example with all needed files: DOWNLOAD HERE

## How to hold several older versions of the Win-GUI app on your PC

Win-GUI apps of different Simulyzer types may be installed in parallel on your PC. But you may only have one version of the same type of Win-GUI app as an official installed version on your PC. But before de-installing an older version (in order to install a newer one), you may create a copy of your C:\Program Files (x86)\SesKion GmbH*Type*-Simulyzer directory and rename it to for example C:\Program Files (x86)\SesKion GmbH*Type*-Simulyzer_Vx.y . You need Admin rights to do this!
*Type* may be PSI5, SPI, DSI or SENT
After having the copy directory you may de-install the old version and install a newer version. To use the old version you may create a link on your desktop to C:\Program Files (x86)\SesKion GmbH*Type*-Simulyzer_Vx.y*Type*_SimulyzerGUI.exe . To get rid of such an older version, just delete the C:\Program Files (x86)\SesKion GmbH*Type*-Simulyzer_Vx.y directory and remove the desktop link.

## How to prepare a Break-out ECU?

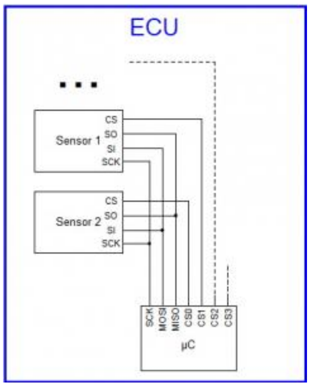
A Break-out ECU is a special case of a standard ECU (for example an Air-bag ECU) to be used in HIL (hardware-In-the-Loop) testing environments. A Break-out ECU does not go to production. The benefit of a „Break-out ECU“ is to route SPI signals that normally go to one or several SPI sensor(s) inside the ECU out of the ECU box. For this an extra connector, for example an extra D-Sub connector has to be place at a suitable position of the ECU case and an hardware expert has to make short wire connections from the SPI-sensor chip to the extra connector.



Here is a photo of a „Demo“ ECU where Seskion made its own Break-out connector.

The „Demo“ ECU has been purchased as evaluation board from freescale company for some 150 Dollar. Its intent has been for users to learn about sensors and about ECUs in Automotive application.

Seskion de-soldered the SPI-sensor chip in the mid of the board and made some connections to a post connector serving as Break-out connector. Another strategy may be to leave the SPI sensor in place on the board and only make some cuts of the SO (Slave Out) signal traces to give the sensor simulating device the chance to feed in its simulated SPI responses. The scheme of the unchanged SPI signals in relation to the SPI-sensors may look like this inside the ECU:



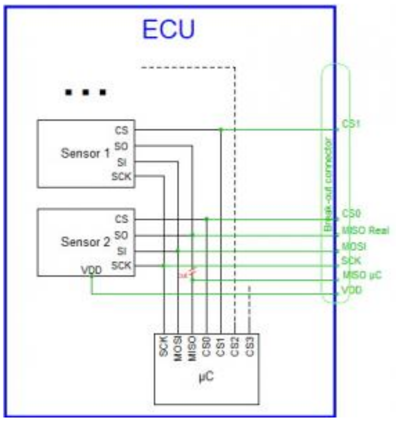
Simple Break-out scheme

Since the simulating device has to drive its simulated responses to the MISO pin of the µC, the original sensor(s) have to be inhibited to drive the MISO line. One possibility is, to cut the copper-trace of the SO outputs of the SPI sensor before going to the MISO input of the µC. Instead, the SO lines of the sensors have to be joined to the Break-out connector and go to the SPI-Simulyzer. The Simulyzer has the chance to read the real sensors response and even has the possibility to re-connect the original sensors SO lines to the MISO input of the µC. Additionally all outputs:

- CS* chip-selects
- SCLK SPI clock
- MOSI master out

have to be joined in parallel to the Break-out connector.

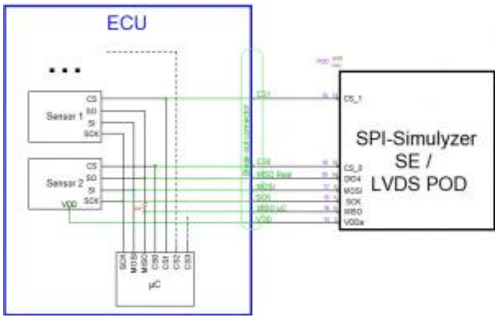
Aditionally the supply voltage of one of the SPI-sensor(s) has to be joined to the Break-out connector; this is for the SPI-Simulyzer may sense the availability of power at the sensors. As long as power is not detected, the software sensor-model in the Simulyzer will not react on any SPI master request and will be held in a Reset state. Last but not least the MISO input of the µC has also to be joined to the Break-out connector:



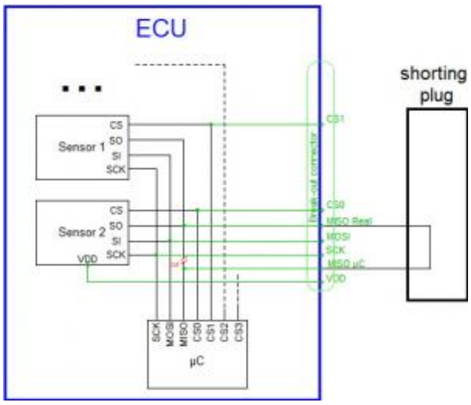
# FAQ

Connection of the Break-out connector to the SPI-Simulyzer. Shown are the pin numbers of both

- LVDS-POD, Sub-D-25, in case of a SPI-Simulyzer-LVDS and
- post connector 26, in case of a SPI-Simulyzer-Single-ended



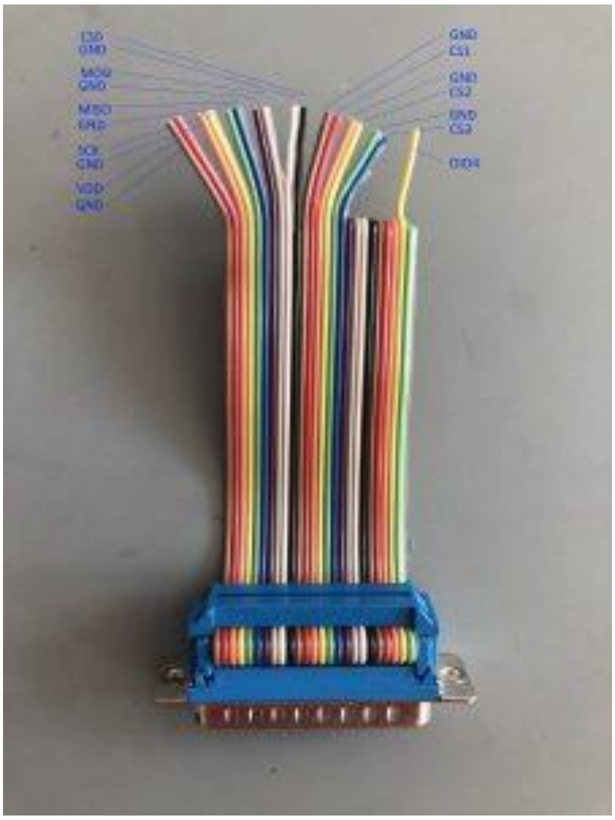
You may also prepare a „Shorting plug“ to be placed into the Break-out connector in case of not simulating the sensors. This revokes the separation cut in the MISO line inside the ECU.



Considerations about wiring of the Break-out connector

For best result, use a SUB-D-25 male connector as Break-out connector, use a flat-ribbon cable of length not exceeding 10cm (4 inch) and connect each signal together with its GND as pairs to your copper traces of Break-out ECU (signal to signal, GND to GND nearby):

- The VDD signal is used by the SPI-Simulyzer to detect if sensor-power is present.
- The SCK signal is used by the SPI-Simulyzer as SPI-Clock imput
- The MISO signal is used by the SPI-Simulyzer asa SO output
- The MOSI signal is used by the SPI-Simulyzer as SI input
- The CS0 signal is used by the SPI-Simulyzer as Chip-select-0 input
- The CS1 signal is used by the SPI-Simulyzer as Chip-select-1 input
- The CS2 signal is used by the SPI-Simulyzer as Chip-select-2 input
- The CS3 signal is used by the SPI-Simulyzer as Chip-select-3 input
- The DIO4 signal is used by the SPI-Simulyzer as input of the original sensors SO signal



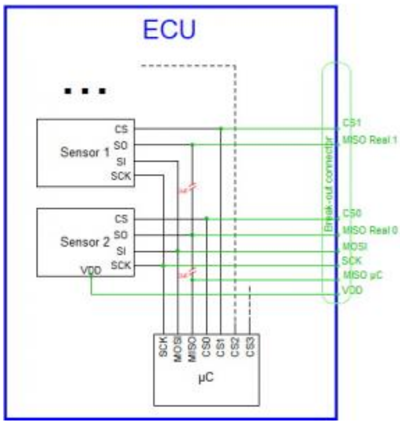
# FAQ

| Signal Name | Pin Sub-D 25 | Pin post-connector 26 | wire in flat-ribbon cable |
|---|---|---|---|
| GND | 1 | 1 | 1 |
| GND | 2 | 3 | 3 |
| GND | 3 | 5 | 5 |
| GND | 4 | 7 | 6 |
| GND | 5 | 9 | 9 |
| GND | 6 | 11 | 11 |
| GND | 7 | 13 | 13 |
| GND | 8 | 15 | 15 |
| n.c. | 9 | 17 | 17 |
| n.c. | 10 | 19 | 19 |
| n.c. | 11 | 21 | 21 |
| do not use | 12 | 23 | 23 |
| do not use | 13 | 25 | 25 |
| VDDA1 | 14 | 2 | 2 |
| CLK | 15 | 4 | 4 |
| MISO | 16 | 6 | 6 |
| MOSI | 17 | 8 | 8 |
| CS_0 | 18 | 10 | 10 |
| CS_1 | 19 | 12 | 12 |
| CS_2 | 20 | 14 | 14 |
| CS_3 | 21 | 16 | 16 |
| Reset | 22 | 18 | 18 |
| Eclock | 23 | 20 | 20 |
| DIO_3 | 24 | 22 | 22 |
| DIO_4 | 25 | 24 | 24 |
| n.c. |  | 26 |  |

# FAQ

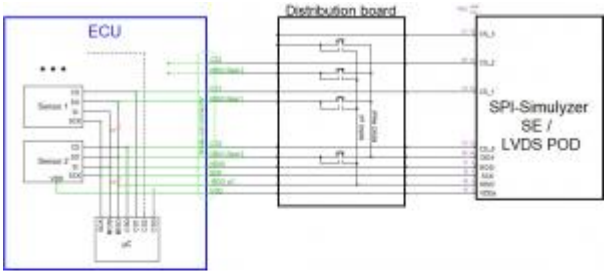
Better Break-out scheme

A more versatile approach for a Break-out ECU is to cut the SO line of each sensor and to route this SO lines individually to the Break-out connector:



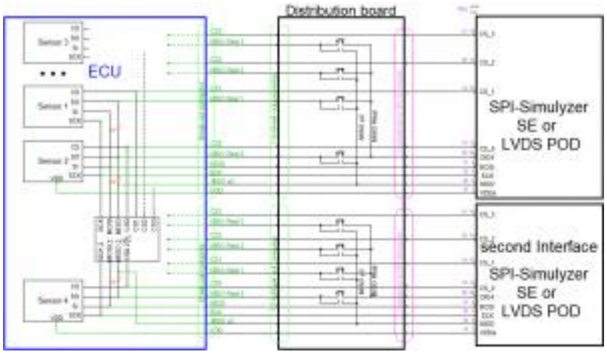
This opens the posibility decide for each sensor if it should be handled by the original chip inside the ECU or by the SPI-Simulyzer. For this, a Distribution board has to be constructed

- to allow routing of the probably different pin numbers of the Break-out connector and the SPI-Simulyzer and
- to set jumpers of each sensors MISO line for either be simulated or be the real sensor



Better Break-out scheme with 2 SPI interfaces

Here is an even more complex Break-out scheme using 2 SPI interfaces. On side of the SPI-Simulyzer 2 interfaces (and PODS in case of LVDS) hace to be used.



How large must the file-based ring buffers be configured?

To ask a bit more precisely: How large have the file-based ringbuffers of the PSI5-Simulyzer to be configured to hold the data of a given timespan

1. Sensordata

Sensordata contain the acquisition of sent and received PSI5-frames / sync pulses. The formula for calculating the size is:

```
fileSize =
(sizeSyncInfo+sizeRespFrame*numberOfSlots)*numberOfInterfaces*numberOfSynperiodsPerSecond*numb
erOfSecondsToAcquire;
sizeSyncInfo = 6; // Byte
sizeRespFrame = 12; // Byte
numberOfSlots = 4; // lets asume a worst case of 4 timeslots in P10P-500-4H
numberOfInterfaces= 2; // lets asume a worst case of both interfaces
numberOfSynperiodsPerSecond = 2000; // lets asume a typical case of 500µs Syn-periode
```

When using the values above, the formula reduces to

```
fileSize = (6 + 12 * 4) * 2 *2000 * numberOfSecondsToAcquire;
fileSize = 216000 * numberOfSecondsToAcquire;
```

In this worst case scenario, you need a ringbuffer filesize of 216000 Byte for each second to acquire. Lets say, you want 10 Minutes to acquire:

```
fileSize = 216000 * 60 * 10; // gives 129600000 Byte (124 MByte)
```

The standard WIN-GUI app can handle this and even larger file sizes if

- The used filesystem supports this file size
- There is plenty of room on the drive to hold the file
- The file has to be a local file, not a network file!

2. rawdata

Rawdata contain the acquisition of PSI5 voltage and current curves being sampled with 3MSamples/s. Only one of the two interfaces can be sampled at a given acquisition. The formula for calculating the size is:

```
fileSize = 12MB * numberOfSecondsToAcquire;
```

You need a ringbuffer filesize of 12 MByte for each second to acquire. Lets say, you want 10 Minutes to acquire:

```
fileSize = 12MB * 60 * 10; // gives 7200 MByte (7.1 GByte)
```

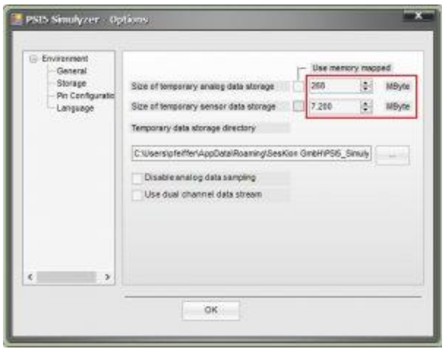
The standard WIN-GUI app can handle this and even larger file sizes if

- The used filesystem supports this file size (NTFS does support even larger files)
- There is plenty of room on the drive to hold the file
- The file has to be a local file, not a network file!

3. Conclusion

In the examples above, you would have to set this sizes:

But please consider, despite this storage size will be handled properly, it may not be so easy to find special parts inside this acquisition. Nor may it be practical to handle a csv export of such large files.



PSI5 Simulyzer - Options

□ Use memory mapped

Size of temporary analog data storage □ 268 MByte

Size of temporary sensor data storage □ 7.200 MByte

Temporary data storage directory

C:\Users\pfeiffer\AppData\Roaming\SesKion GmbH\PSI5_Simuly

□ Disable analog data sampling

□ Use dual channel data stream

## What physical errors does the PSI5-Simulyzer recognize?

The PSI5-Simulyzer recogizes following errors upon receiving a Manchester Frame:

| Error | Error bit | Meaning |
|---|---|---|
| timeslotError | 0x01 | data frame starts before start of timeslot or ends after end of time-slot |
| startBitError | 0x02 | error in requested value ’00b‘ of startbits |
| manchesterError | 0x04 | no polarity change in mid of Manchester bit or number of bits different to configured one |
| parityError | 0x08 | error in parity bit or in CRC bits |
| currentError | 0x10 | only detected if ‚CurrentCheckMode‘ is enabled; reported if „0“-current is outside 0-upper or 0-lower limit or if „1“-current is outside 1-upper or 1-lower limit |
| noDataError | 0x20 | no data (polarity change) within time-slot |
| bitTimeError | 0x40 | only detected if ‚ExtendedTiming‘ is true; the bit-times do not match the default the default tolerance of +5% and -5% |

## How to permanently store your received sensor data in files?

First of all, the files in the temp-directory are not intended to be processed after terminating the your application. They just are intended as a file-based ring-buffer where received binary data is temporary stored. While you application is still running, there are API calls to get data out of these temporary files (see discussion further down).

If you want your acquired data to be available after terminating the your application, you should call

```
StoreData(HANDLE hSensor,unsigned char mode,char *rawDataFile, char *sensorDataFile);
```

if your .ppf project files is named *myProject.ppf* and is located in */your/Path/to/Config/* you should give following parameters:

```
StoreData(yourSensorHandle, 3, "/your/Path/to/Config/myProject.rawData",
"/your/Path/to/Config/myProject.sensorData");
```

Of course, you may chose different path, different file-basename and different extension, but doing it this way has the advantage you could use our standard WIN-GUI app to open */your/Path/to/Config/myProject.ppf* and the GUI will automatically find and open the *myProject.rawData* and *myProject.sensorData* files and also open them for showing them as graphs and tables.

## Is there any limitation to connect 5 or more Simulyzer with only 1 laptop?

There are no problems. You only need a appropriate USB-Hub for that.

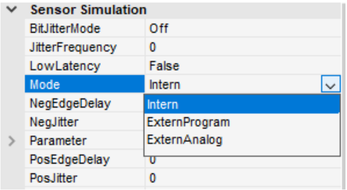
## How to configure sensor simulation with PSI5-Simulyzer?

Basic:
In sensor simulation there are 3 modes what signal data is used:

- Intern, using an internal cyclic incremental signal (Sawtooth). Warning: The saw-tooth spans the complete data range of -512 .. +511 (at 10 bit signaldata). This may interfer with init-data being in the area of -512 .. -481 or +481 .. +511. Sow the sawtooth is nice but gives problems if init-data are important!
- Extern Program, using configurable constant data or imported signal sequence
- Extern Analog, using selectable analogue inputs to be the source of signal data.

The selection box for mode can be found in our WIN-GUI app in the „PSI5 Bus Configuration“ panel beneath section „Sensor Simulation“

In this example we will use Extern Program and prepare an excel sheet for import. Please note, you have to do this independently for each channel (interface) by selecting the appropriate „channel“-tab.



Sensor Simulation

| | |
|---|---|
| BitJitterMode | Off |
| JitterFrequency | 0 |
| LowLatency | False |
| Mode | Intern |
| NegEdgeDelay | |
| NegJitter | |
| Parameter | |
| PosEdgeDelay | 0 |
| PosJitter | 0 |

Download the example:
Please download the files being used here from ExampleSensorSim.zip. Unzip the downloaded file to get these files extracted:

- P10P-500_3L.ppf the configuration file
- Sinus_6.xlsx an Excel sheet for constructing some signal traces
- Sinus_6.csv file to be imported to the configuration
- P10P-500_3L.simulationData not really needed. Will also be created in this exercise

and copy the files to some test directory on your Win-PC.

Preparing the Excel sheet:
The Excel sheet „Sinus_6.xlsx“ is only a tool to create a csv file that can be imported to the configuration of a PSI5-Simulyzer. There are 6 columns for signal traces, implemented as some sinusoidal curves. You may play around with different traces.

Creating the csv file and import:
When done, the Excel sheet should be „save as“-ed to create a Sinus_6.csv. The latter file is also part of the zip example.

Importing the csv-file into your Simulyzer project:
Start the PSI5-Simulyzer GUI app and „File“ => „Open“ the P10P-500_3L.ppf file extracted from the zip-file. Then click to „File“ => „Import“ and in the appearing open-panel select the just created „Sinus_6.csv“ file. In the appearing „Import Sensor Data“ panel just click „Ok“.

Run the simulation:
Just click the green „Run“ button and then the „Power“ button. After some seconds of running you may stop and power-off. You should see in the „Sensor Signals panel“ the sinusoidal curves. Finally click to „File“ => „Save“. this will lead to a „P10P-500_3L.simulationData“ file besides the „P10P-500_3L.ppf“ config file. It is a binary representation of the imported signal traces. When later on the GUI app will re-open the „P10P-500_3L.ppf“ config file, it will also find the „P10P-500_3L.simulationData“ an you do not have to do the import every time.

Important settings of the config file:
Operating mode in this example is „Simulation", so you do not need any external components to be connected to the Simulyzer. The Simulyzer takes both roles, the ECU as well as the senso(s). In panel „PSI5 Bus Configuration", section „Sensor Simulation" the Mode has to be „ExternProgram" to use the imported signal traces. In panel „PSI5 Bus Configuration", section „Sensors" ech sensor should have „External" set to false, elsewise the sensor is assumed to be physical present external to the box and will not be simulated.

How to get sensor data inside your app?
This is about getting sensor data while your app is running. There are two situations when you may want to get the data:

- Immediately after starting the data acquisition you want to look at the data while it continuously streams from the box or
- after stopping of the data acquisition you may want to look at the data.

In case 1, you have to to hurry to get the data fast enough to keep up with new arriving data. This may be the preferred method if you have a long acquisition time, even with no predefined end.
In case 2, you have all the time you want to get the data. his may be the preferred method if you have a short acquisition time, lets say some seconds. Then you start acquisition, stop it after some seconds and then look at the data.
You will find examples below your installation directory in C:\Program Files (x86)\SesKion GmbH\PSI5-Simulyzer\API\
You also will find a complete Visual-Studio project PSI5-Simulyzer4AnsiC.vcxproj in C:\Program Files (x86)\SesKion GmbH\PSI5-Simulyzer\API\ANSI-C\PSI5_Simulyzer4AnsiC\PSI5_Simulyzer4AnsiC

## PSI5-Simulyzer specific Questions

Is there a possibility to record existing sensors to simulate them later?

The possibility to recorde existing PSI5 sensors (satellites) and simulate them later is one of many use cases our PSI5 Simulyzer can cover.

Can we plug the PSI5-Simulyzer from one setup to another without additional costs?

For the PSI5-Simulyzer there is the option to bind the software to a PC. However, if you want to jump with the Simulyzer from one setup/test bench to another, I recommend the box bound license. This allows you to run the Simulyzer on any PC.

Do we need the external power supply in ECU mode?

To cover all voltage ranges you need the 12V external power supply. With the USB connection a maximum of 5V is covered. In general it is always recommended die externe Stromversorgung mit anzuschließen.

Have you got a solution to read more than 4 sensors simultaneously? For example 10 or 20.

For a fast prototyping our SimulyzerRT system would fit perfectly. The individual components are as follow:

- 3x PSI5 SimulyzerRT Cards (each Card has 8 PSI5 Channels)
- Gateway CPU Card (communication between external software and PSI5 Cards)
- chassis (including power-supply)

The communication can be done with our GUI Software or can be called per API, so you can integrate it in your application. Typical it is possible to operate 4 airbag accelerometers on one channel. In this case you should be fine with only 1 PSI5 SimulyzerRT card.

Is it possible to use Profinet with dedicated gsdml or connection with PLC via RS232?

A direct programming with Profinet or RS232 is not possible. We can only create the direct connection to the PSI5.

Does PSI5-Simulyzer support LabView?

The PSI5-Simulyzer supports LabView. For this purpose a LabView Libary can be ordered. (Art.No.: 9.0011)

Does PSI5- Simulyzer support CAN control?

With the optional CAN interface (Art.No.: 1.2001), which can be ordered with each Simulyzer, CAN communication can be used.

Is USB communication possible with Linux?

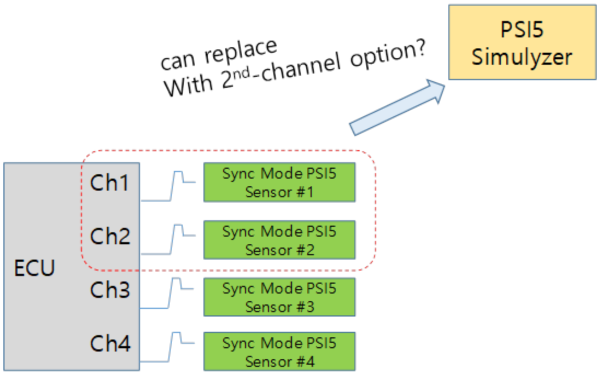
USB communication is also possible via Linux with the provided DLL.

# FAQ

How many sensor signal available simulated by PSI5-Simulyzer?

We need 4EA PSI5 Sensor simulation for one equipment. Is 4EA PSI5 Sensor Simulation = 4 PSI5-Simulyzer?:

If you need 4 interfaces you can buy 2x PSI5-SImulyzer with 2x Second Channel.



Is it possible that we configure PSI5-Simulyzer using LabView library without Windows software package? Or Windows software package is mandatory option?

The Windows software package is not mandatory. The Simulyzer Box is always delivered with the DLL for self programming. The Software is good for the implementation and for error searching which can be occur during wiring. The LabView library is not associated with the Software.

How much latency does the translation of the messages from PSI5 to CAN generate?

The latency you requested for the PSI5-Simulyzer is 20 µs.

What would be requirements for connecting of 3x PSI5-Simulyzer with second channel to one PC, if we want to have this high sampling rate (2000Hz)?

The 2kHz sampling rate is standard for us. We have other setups with up to 6x PSI5-Simulyzer with second channel running and have no problems at all. You need 3 USB connections with your PC, for every Simulyzer one connection.

Is it possible to measure the correct electrical characteristics and operate with PSI5 devices on the bus with the PSI5-POD or did we need the PSI5-Simulyzer?

With the POD no current or voltage can be displayed. Also no passive mode or sensor simulation is possible, which would be necessary for transceiver validation. Unfortunately, this means that the POD would not be able to cover your needs.

For PSI5 sensor, OTP need to be programmed with PSI5 bus voltage at about 10V(9V to 11V), is that possible to do OTP programming with PSI5-POD?

So at the POD the Supply Voltage is max 7.7V. The 11V refer then to the Sync Pulse height. But the POD cannot communicate bidirectionally and therefore of course no OTP programming.

# FAQ

We usually use Linux. Is it possible to work with this or did we need Windows?
It is no problem to get the API for Linux. We already have several customers running the Linux version without any problems. The only thing that can't be offered then is a GUI.

In which condition, we select ECU mode, and in which we select sensor mode?
If you have sensors to measure -> you have to select the ECU modus. If you have an ECU and you will replace the sensors with the Simulyzer -> you have to select the Sensor modus. If you have both (ECU and Sensors) and you only want to sniff the communication -> you have to select the Passive modus.

What is the "API-ANSI-C (included in the box)"?
We deliver header files .lib and .dll files to write your own programs free of charge. Included in the box is somehow misleading here! The files are part of the software installation.

What is an EMC Test?
EMC Test is a software program to read out sensor data in parallel to EMC tests. The software check for communication or sensor signal errors

What is the Sensor Programmer-Software?
Sensor programmer-Software is a command line tool to program serveral PSI5 sensors via bidirectional PSI5 communication. Typically used for end of line programming in production

What did we need for Sensor test software for up to 5 PSI5 boxes?
Sensor test software is a program to continuously measure sensors which are typically in a climate chamber when measured. The sensors could be switched on/off periodically and voltage parameters could be changed with a simple scripting interface. With up to 6 boxes up to 12 PSI5 interfaces could be measured in parallel. The software check for communication or sensor signal errors. Typically used for QM validation.

Is it possible to make a communicate to PC using SCPI?
No, there is no support for SCPI (Standard Commands for Programmable Instruments).

Is it possible to control PSI5-Simulyzer only with LabVIEW program?
Yes it is possible to control the Simulyzer with the LabView library from us.

Can I use only one trigger signal for all PSI5s to make sure the same T0 for collecting data?
Yes, you can use only one trigger signal for all PSI5 to make sure the same T0 for collecting data.

<u>I plan to configure 4 Channel in the future. Whether to purchase duplicate window software package and LabVIEW library?</u>

For the 4 channels you need 2x PSI5-Simulyzer hardware boxes with second channel. So you don't have to order 4x the Hardware Box. The software you need is either box or PC-ID bound. So if you have one test PC it would be enough if you have one PC-ID bound license.

<u>I need to simulate synchronized PSI5 data on all my Test ECU's channels. Is there a way to synchronize multiple Simulyzers so that they start transmission at the same time?</u>

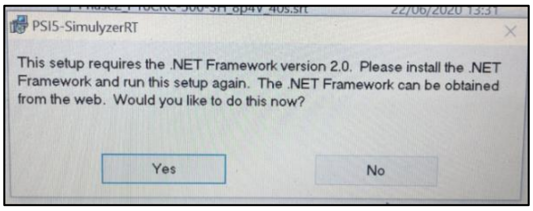
- I will have sensor data from a real-world source which I will need to "replay" in simulation with the Simulyzer tool. Is this possible?
- Multiple Simulyzers will need to be linked together in parallel to achieve simulation on all my Test ECU's channels. How many can I link together?
- I need to control through LabVIEW or C-APIs. Do you make all the above requirements possible with your library and API support?

Answer: All Simulyzer's have digital inputs/outputs which either generate or receive a trigger signal. On reception of this trigger signal simulation data is streamed on the PSI5 interfaces

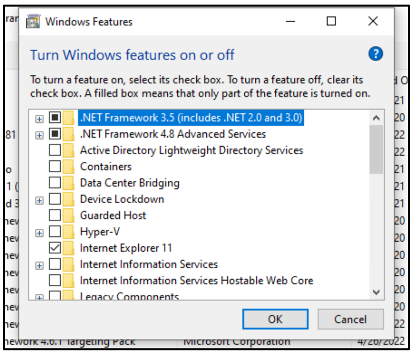
- This data is already queued via USB to the boxes, so there is no delay in between trigger signal and data stream start. The stream data could be imported form .cvs files or provided as binary data on the API interface.
- There is no limitation of parallel used Simulyzer's on the PC. We recommend dedicated USB host controller cards in the PC instead of USB connection on motherboards when using multiple Simulyzer's in parallel.
- Our API provide an interface to group more Simulyzers into a so called SimulyzerSystem. This system could be configured with one API call and a configuration file where the Simulyzer's of the system are defined. One additional API call is needed to provide the simulation data to system. That's all. After receiving the external trigger signal the simulation data is streamed on the PSI5 interface.

A typical setup we have running at several OEM's and TIER1 for Airbag functional verification consists of up to 6 PSI5-Simulyzer + 1 SPI-Simulyzer. In these setups the SPI-Simulyzer is used to simulate the internal sensors of the Airbag ECU. So the complete crash situations could be simulated to the ECU for functional verification. Additional we provide a so called SQUIB box to detect the fire pulses from the ECU.

<u>I try to install the software but I got the warning .NET Framework.</u>



This setup requires the .NET Framework version 2.0. Please install the .NET Framework and run this setup again. The .NET Framework can be obtained from the web. Would you like to do this now?



Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ■ .NET Framework 3.5 (includes .NET 2.0 and 3.0)
- ■ .NET Framework 4.8 Advanced Services
- □ Active Directory Lightweight Directory Services
- □ Containers
- □ Data Center Bridging
- □ Device Lockdown
- □ Guarded Host
- □ Hyper-V
- ☑ Internet Explorer 11
- □ Internet Information Services
- □ Internet Information Services Hostable Web Core
- □ Legacy Components

It seems you have not enabled .Net 2.0 support on your machine.

## DSI-Simulyzer specific Questions

Is it possible to simulate the ultrasonic sensors of a vehicle, respectively their control unit, in HiL setup with the Simulyzer?

Physical - protocol level:
If you want to communicate directly with an ECU, the simulator must of course be able to speak the desired protocol. Meaning: without knowing it, no simulation can work.

Sensor Specific:
Generally, ECUs use drivers to configure the specific protocols or the sensors to be addressed via them or to get measurement data in the second step. Therefore the simulating instance has to emulate a certain part of the sensor, so that the ECU believes that it is connected to the correct unit.

I want to simulate 12x Ultrasonic sensors via DSI3 interface, can you make a quotation for me?

It depends on how your sensors are connected. Are they connected with a point-to-point connection or are there several sensors on one interface? With the DSI3-Simulyzer up to 6 sensors can be timed on one interface with different timeslots. Therefor two Simulyzers are enough to simulate the sensors.

Can it change how DSI messages are sent ?

Yes, basically you can specify which error should be generated. CRC, invalid MLSC, timing and frame ignore errors can be generated. In addition, we have an MLSC Sequenzer where freely defined frames can be constructed.

Can you explain the API of DSI3-Simulyzer Software and the bandwidth of USB interface?

For the API interface we provide a share library which can be integrated into any c++/c#/python application. When using it, you can load a previous created configuration from the GUI application and start the simulation. The box has an USB2.0 Interface.

Will there be a Gigabit Ethernet interface on DSI3-Simulyzer box and multiple channel DSI3-Simulyzer box in the future?

Currently we are in developing of a simulation card which has USB 3.0 and ethernet connectivity. This device will provides up to 8 interfaces. However because of the current component shortage we are behind schedule.

Would it include end to end software solution or also software?

We can provide the Boxes and a GUI Software. In addition, you could write your own application with the API (DLL) we provide.

Bad format can it be identified and simulated?

Yes

Lost CRC can it be identified and simulated?

Yes

What are the mature use cases in Seskion for DSI3-Simulyzer?

I like to name 2 off them:

1) using the internal simple sensor simulation, where basic read/writes and streaming methods used
2) using the streaming functionality where you replay a dataset you specify in a CSV file. Here you have also the possibility to inject errors (like CRC, missing dataset, timing errors).

In addition we provides services to develop an specific sensor simulation for customer needs. This point is typical needed for more complex sensor simulation environments.

Half DSI msg format is received can it be detected?

Yes

Can the box send incorrect formatting?

YES

What is the flexibility the box has in sending manipulative messages?

You can specify a streaming file (in CSV-Format) to send any frame you like, including the described errors.

Is the software locked to a fixed PC?

You can choose between Box or PC-ID bound license.

Could you provide a detailed solution and material of DSI3-Simulyzer box , there are any cooperation with OEM and Tire1, such as ZF and so on?

Our DSI3 Simulyzer has been used so far by ASIC manufacturing companies to verify their new chips.

The actual transmission speed of USB 2.0 generally does not exceed 60M/s?

Yes that’s right. USB 2.0 is limited by 53MBytes/s

I have a request to monitor and log DSI3 bus with 6 channels of DSI3 bus, each DSI3 channel connecting two Ultrasonic sensors. These 6 DSI3 channels connected to my SoC with DSI3 controllers via SPI.

- How many DSI-Simulyzer boxes do I need?
- I want to integrate DSI-Simulyzer box in my APP, could you offer DLL API? Free or cost?

Currently our DSI-Simulyzer Box supports 1 Channel with up to 15 Timeslots. For your application you need 6 boxes and one PC bound Software-License for initial configuration. The DLL/API is included and can be used from C++ or C#.

## SPI-Simulyzer specific Questions

Is it possible to simulate an acceleration sensor with the SPI-Simulyzer? Does a simulation model exist?

In general, we have already implemented many sensors. If we don't have a sensor yet, it can be set up using the sensor specification provided. Normally we estimate about 2 weeks for this.

I want to use it for Simulation of SPI position sensor. Is this device suitable for sending 32bit frame as SLAVE ? And can be programmed in python or c#?

Because each sensor is different, the sensors and their parameters must be implemented so that the ECU still thinks that the correct sensor is connected. SPI and DSI3 are not standardized bus protocols and differ in their communication. PSI5 for example is a standardized bus protocol and therefore does not require a Sensor Model. For the implementation of the sensor we need the sensor specification to create the sensor model. For this we need about 4-6 working days from the time of sending the specification. The device is suitabale for sending 32bit frames as slave and can be programmed in python or C# with the .Net DLL which we always supply.

## SENT-Simulyzer specific Questions

Could you confirm that the SENT-Simulyzer supports following requirements?

- Sequential SENT Description
  - The devices are configured to use the Sequential SENT (SSENT) output mode. This is a “shared SENT” mode with up to 4 devices sharing the same output line. Each device has an address hard-wired to the IC via the SA0 and SA1 pins. On power-up, each device has an internal ID counter set to 00. When the devices receive an F_OUTPUT trigger pulse (15-19 ticks low triggered by the controller), they respond with a SENT message if their internal ID counter matches their hard-wired sensor ID. The internal ID counter is also incremented. If the internal ID counter is incremented beyond the maximum sensor ID configured by the MAXID parameter, the internal ID counter is reset to 0. In this way, each device responds in sequential order on the same output line.
- slow serial message
  - up to 16 different messages of 1 byte

Answer: Yes, both requirements are handled by our SENT-Simulyzer.

## Application Notes

AN Box or PC-ID bound License

AN DSI3 Software Licenses

AN Error Analysis

AN HiL-Setup with PSI5-, SPI- and SQUIB-Box

AN How many Simulyzer do I need

AN How to install Seskion License

AN Instructions Simulyzer LabView-Library

AN Optional 2nd-Channel & CAN-Interface

AN PC-ID bound License

AN How to inject PSI5-Sensor Data via CAN

AN PSI5 Sensor Programming

AN PSI5 Using LabView to Collect and Export Data

AN Replacing fuse in the Simulyzer-Box

AN Simulyzer-Software Operating Help