



New Features in ibaPDA v6.33.0

Author: iba Gent

Date: 11/09/2014

Table of contents

1	S7-Xplorer	4
1.1	S7-200 support.....	4
1.2	LOGO! support.....	5
1.3	ibaPDA-S7-Xplorer Proxy: PLCSIM in Windows Vista and later	9
1.4	Timer and counter operands.....	11
2	Codesys-Xplorer	12
3	HiPAC Request	22
4	ibaHD server changes	27
4.1	Text channels	27
4.2	Diagnostic data.....	28
5	ibaBM-DP	30
5.1	S7 request compatible with ibaCOM-L2B	30
5.2	L2B real format supported for FM458 and TDC request	32
6	Non-compressed data.....	33
7	Technostring changes	34
8	Installer changes	35
9	User management changes.....	36
10	Expressions: navigation to signals	38
11	InSpectra.....	39
11.1	Universal CM module	39
11.1.1	Settings and signals	39
11.1.2	Snapshots	41
11.2	Fan module	43
11.2.1	Settings and signals	43
11.2.2	Snapshots	45
11.3	InSpectra Expert.....	48
11.3.1	Averaging	48
11.3.2	Copy / paste calculation settings	49
11.4	InSpectra interface: toggle snapshots	50
12	Improved import / export (text files)	52
12.1	Groups	52
12.2	InSpectra profiles.....	53
12.2.1	Text format	53

12.2.2 Import / export via InSpectra profiles dialog	55
13 FFT view	57
13.1 Slave organization	57
13.2 New Slaves.....	59
13.2.1 Slice slave	59
13.2.2 Marker-spectrum slave	63
13.3 FFT Calculations.....	66
13.3.1 Order calculations.....	66
13.3.2 Integration / differentiation, units and scaling	68
13.4 Markers.....	69
13.4.1 Sideband tooltip	69
13.4.2 Marker styles	69
13.4.3 Harmonic marker dragging.....	70
13.4.4 Copy / paste of markers.....	70
13.5 Printing	71
13.6 Slave visualization	71
13.7 ibaPda preferences.....	72

1 S7-Xplorer

S7-Xplorer is the new name of the S7 Analyzer interface. It is now part of the PLC-Xplorer product. This product supports access to different PLCs including Siemens S7, Codesys-based PLCs and Allen-Bradley PLC5 and SLC500 PLCs.

1.1 S7-200 support



S7-200 class CPUs are now supported in S7-Xplorer. It is possible to communicate with the CPU through an add-on Ethernet module or using a PC/CP connection. As with other S7-Xplorer modules, the communication parameters need to be configured in the Connection tab after adding an S7-200 module in the S7-Xplorer node in the I/O Manager.

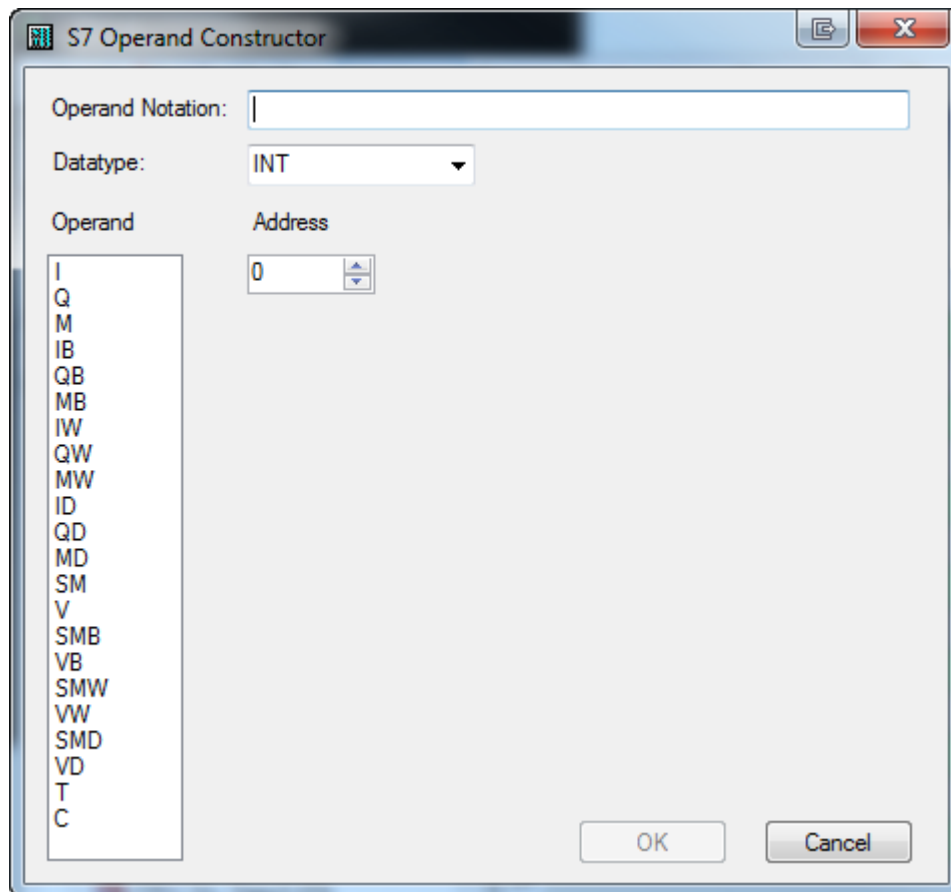
General Connection Analog Digital Diagnostics

Connection mode: TCP/IP Timeout (s): 15

Address: 192.168.123.242 Test

☐ Use projected connections

Note that it is also possible to make use of projected connections.

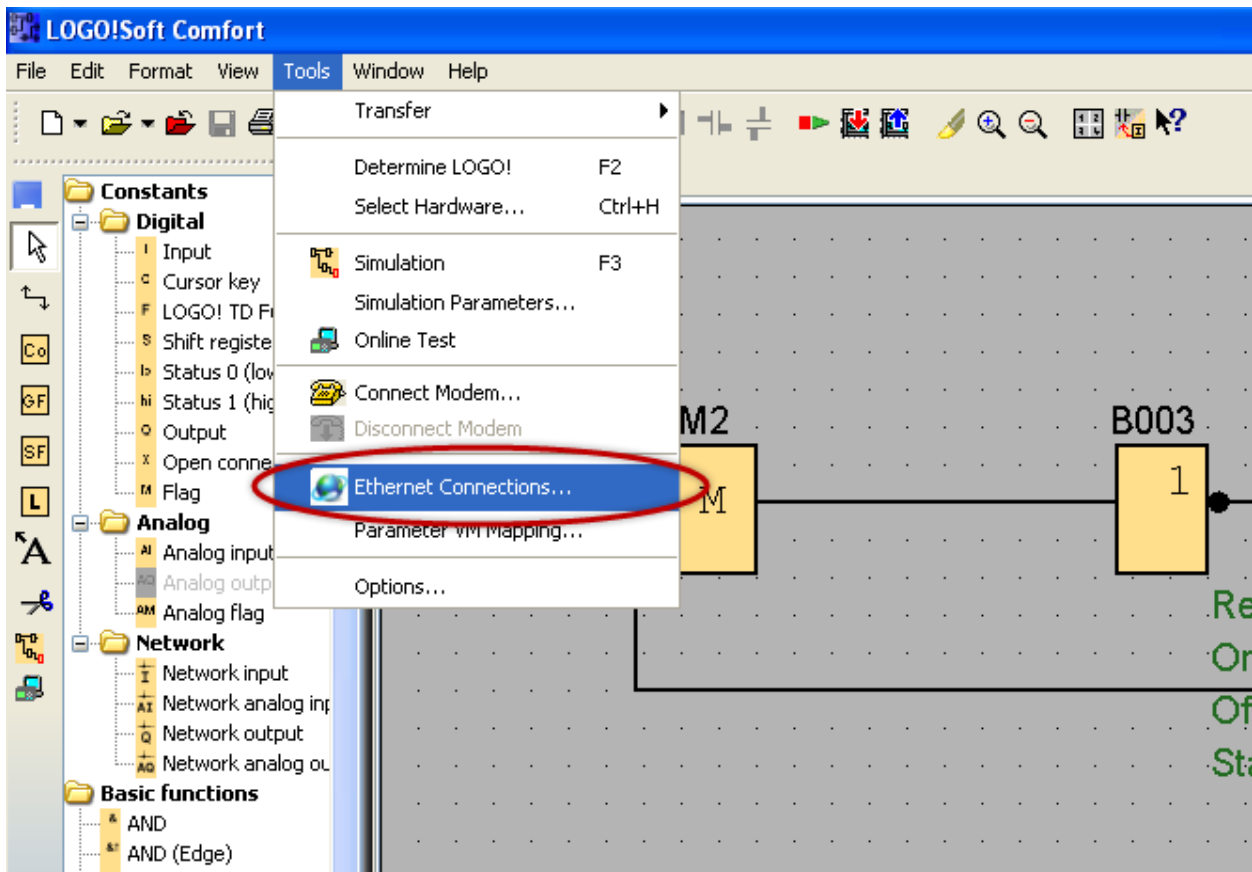


Since S7-200 CPUs do not support symbol address books, the requested operands have to be configured using the S7 Operand Constructor. The following standard memory areas can be requested for S7-200 CPUs (a bit value, byte, word or double word): I, Q, M, SM and V. Furthermore, it is also possible to request Timer and Counter variables (see also 1.4).

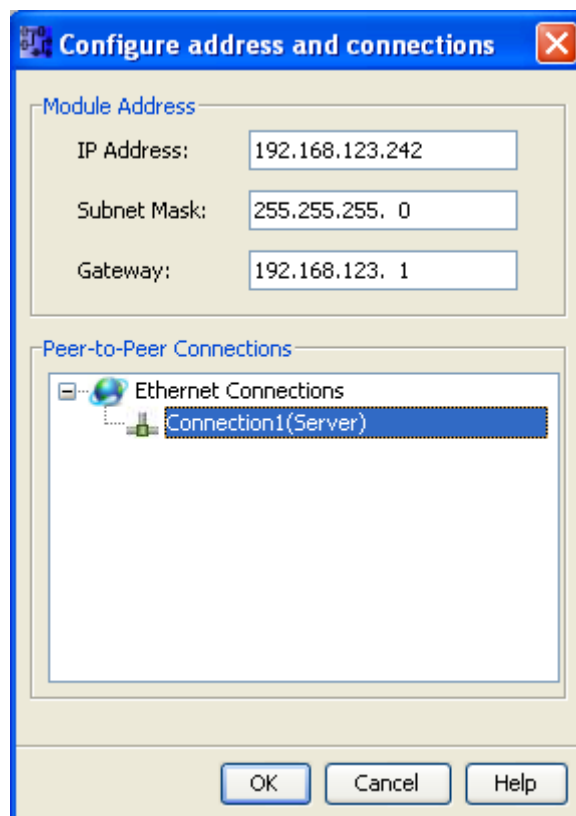
1.2 LOGO! support



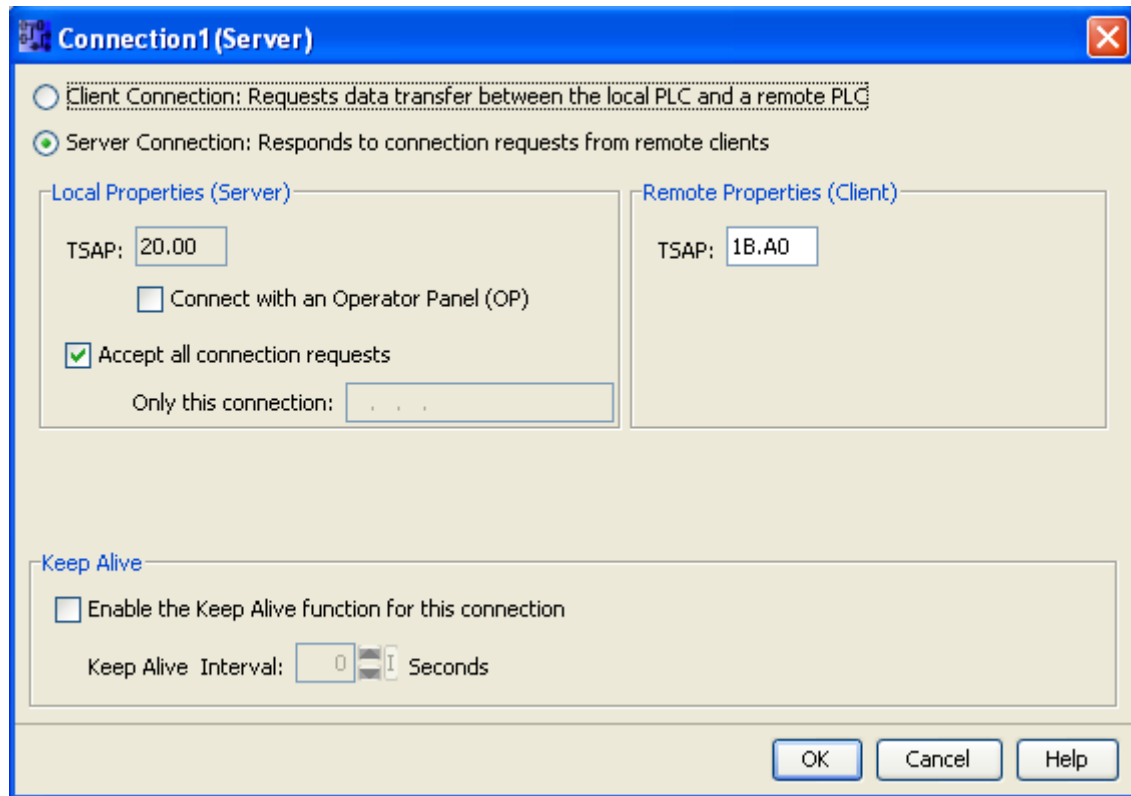
Another new type of CPU that is supported is Siemens LOGO! (provided that they contain an Ethernet module).



After downloading the desired program to the PLC, the network settings have to be configured to allow incoming connections from S7-Xplorer. This can be done in LOGO!Soft by opening the Tools > Ethernet Connections... menu.

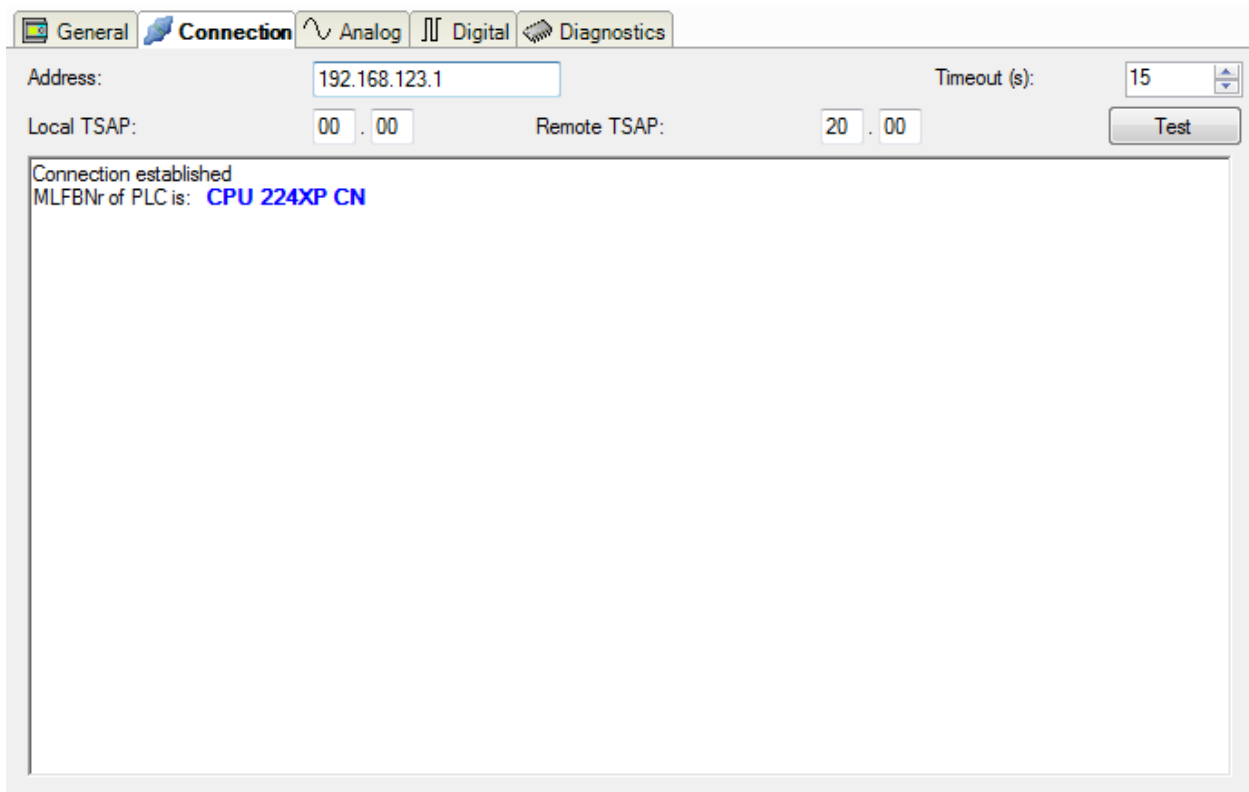


First, we have to configure the IP Address, Subnet Mask and Gateway of the LOGO! CPU. Next, we have to add a new Peer-to-Peer connection by right-clicking on Ethernet Connections and adding a new connection. When double-clicking the newly created connection, we see another dialog.

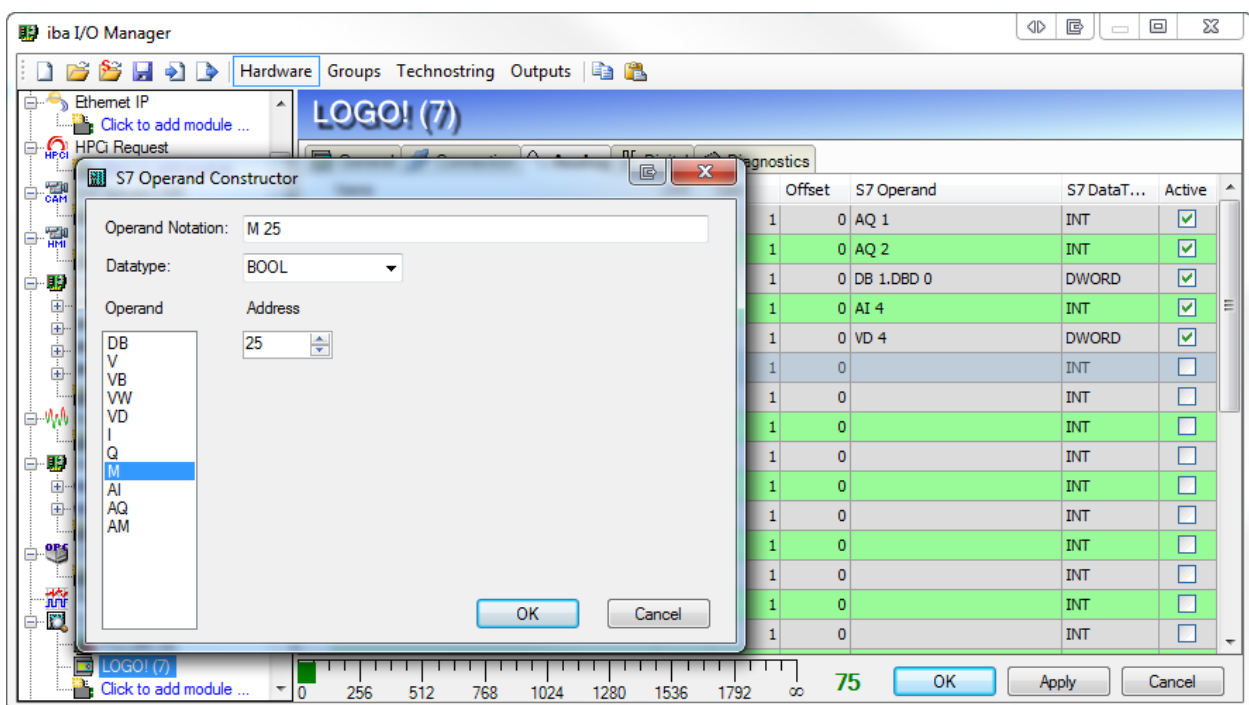


In this menu, we have to select Server Connection (since ibaPDA acts as a client here). It is possible to restrict the allowed client IP addresses but in this example we have chosen to allow all incoming connections. Finally, we have to defined a TSAP that will be used at the client (ibaPDA) side. This number consists of 4 hexadecimal digits, separated by a dot, and can be chosen freely.

After downloading this configuration to the PLC, we can start configuring ibaPDA. First, we need to add a new module in the S7-Xplorer node of the type LOGO!. The connection tab of this module is as follows:



We can either chose to communicate directly using TCP/IP or by using PC/CP access point. In this example, we will be using TCP/IP. Apart from the IP address of the LOGO! CPU we have to enter the information we configured in the network configuration of LOGO!Soft. Note that here, “local” represents ibaPDA while “remote” refers to the PLC. After inserting the correct values, we can test the connection by clicking the appropriate button.

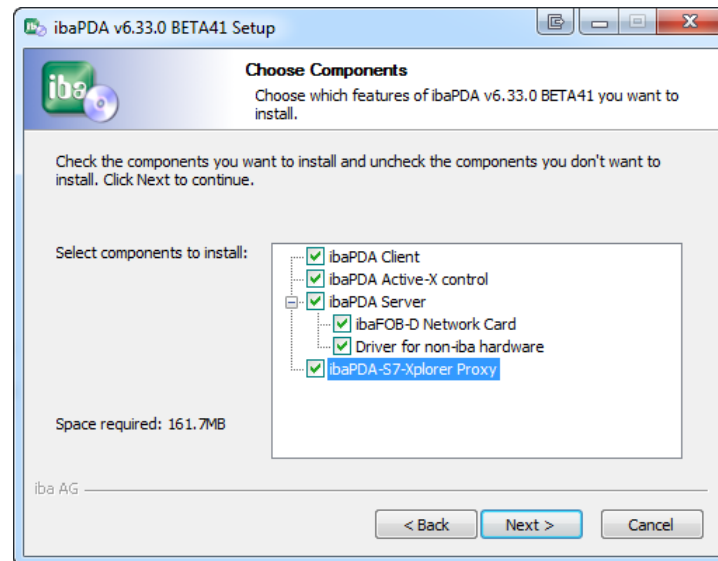


Operands can be selected in a similar way to standard S7-Xplorer modules although there is no address book functionality.

1.3 ibaPDA-S7-Xplorer Proxy: PLCSIM in Windows Vista and later

As of Windows Vista, it is not possible anymore to establish a proper connection between the ibaPDA service (more specifically S7-Xplorer) and PLCSIM. Since this is due to a fundamental restriction in Windows Vista, it is not possible to solve this issue directly. Therefore, we have created a separate proxy application that can facilitate a connection between PLCSIM and S7-Xplorer: ibaPDA-S7-Xplorer Proxy.

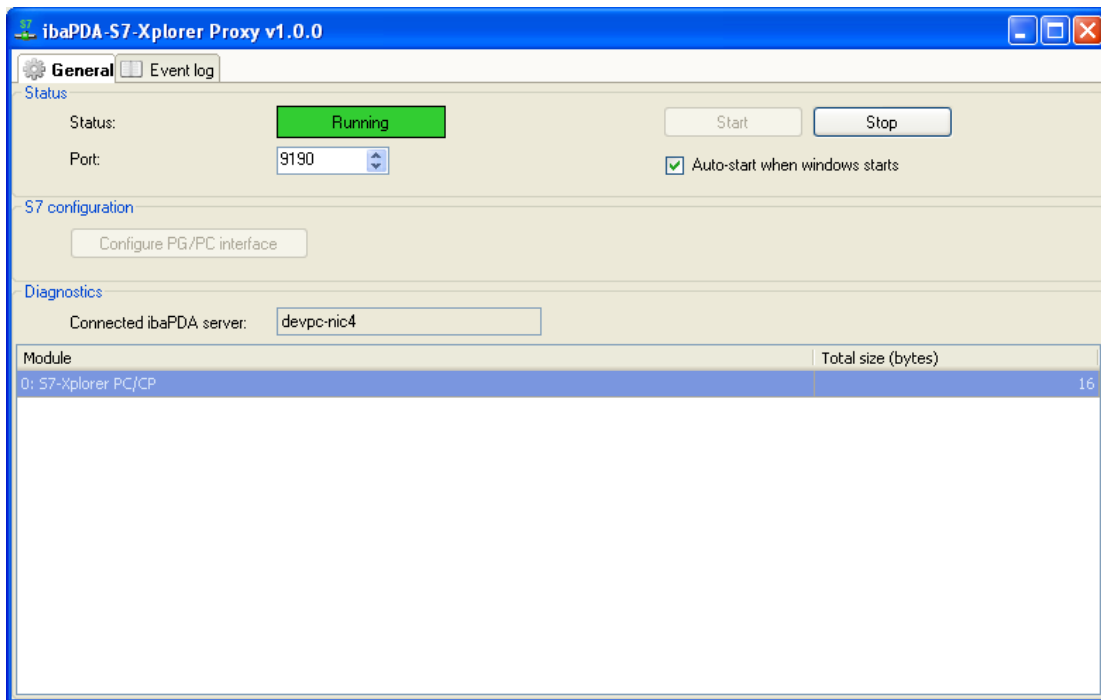
This implies that, when using PC/CP access points, it is no longer required that ibaPDA is running on the host controlling the S7 hardware.



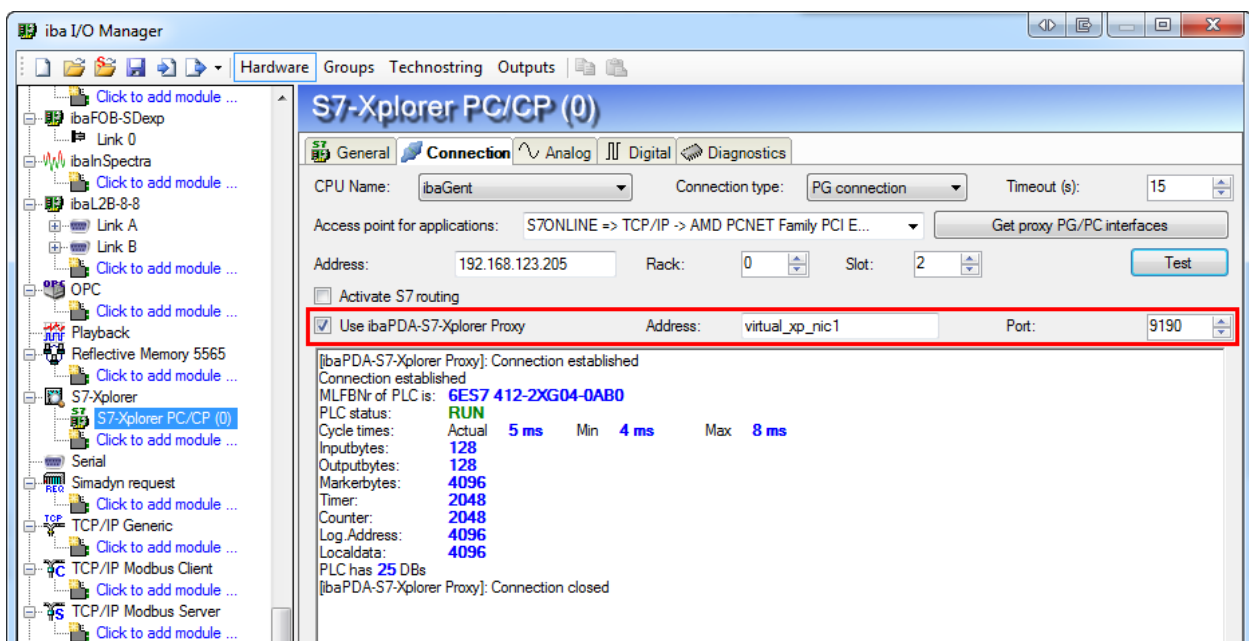
The ibaPDA-S7-Xplorer Proxy is part of the ibaPDA installer. It can be installed separately or together with the ibaPDA client and/or server.



The ibaPDA-S7-Xplorer Proxy runs in the system tray. The icon changes color depending on the status of the application. It is green when an ibaPDA server is connected and measuring data. It is orange when it is waiting for an ibaPDA server to connect and it is red when it couldn't start waiting for an ibaPDA server connection.



When you double-click the icon the status window opens. Here you can configure the TCP port number the application should listen to for ibaPDA connections.

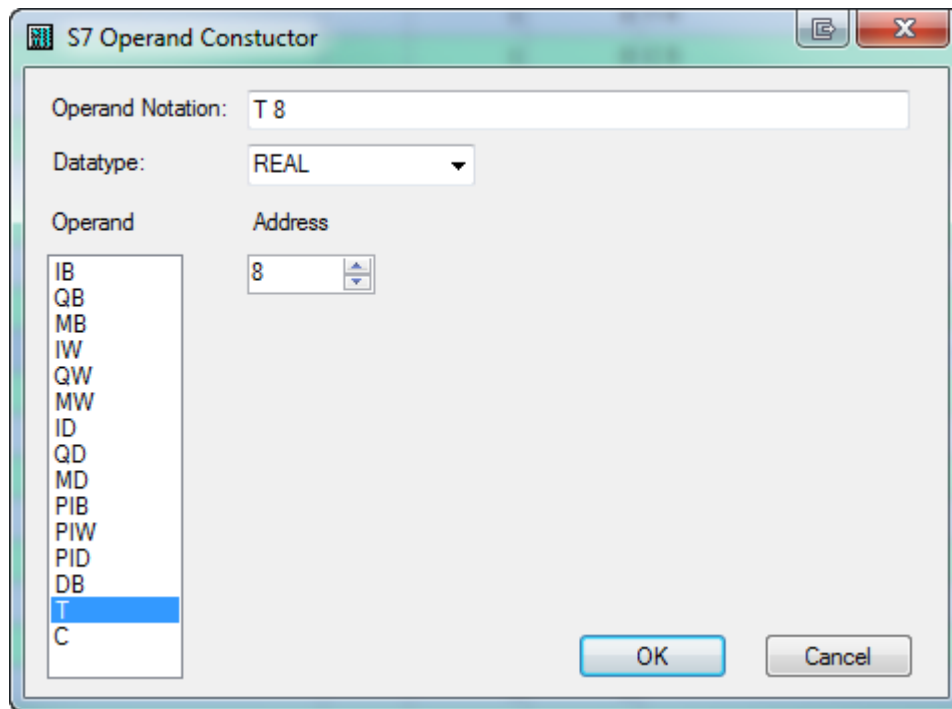


Once the ibaPDA-S7-Xplorer Proxy is properly configured, we can try connecting to it using S7-Xplorer in ibaPDA. The connection tab of the S7 PC/CP modules now has a new checkbox “Use ibaPDA-S7-Xplorer Proxy”. When this option is enabled, it is possible to enter the IP address or hostname of the host running the ibaPDA-S7-Xplorer Proxy and the port at which the proxy application is listening for connections.

Apart from this setting, nothing else has to be changed to request variables from PLCSIM.

1.4 Timer and counter operands

For both S7-200 and S7-300/400 CPUs, the timer and counter operands are now supported.

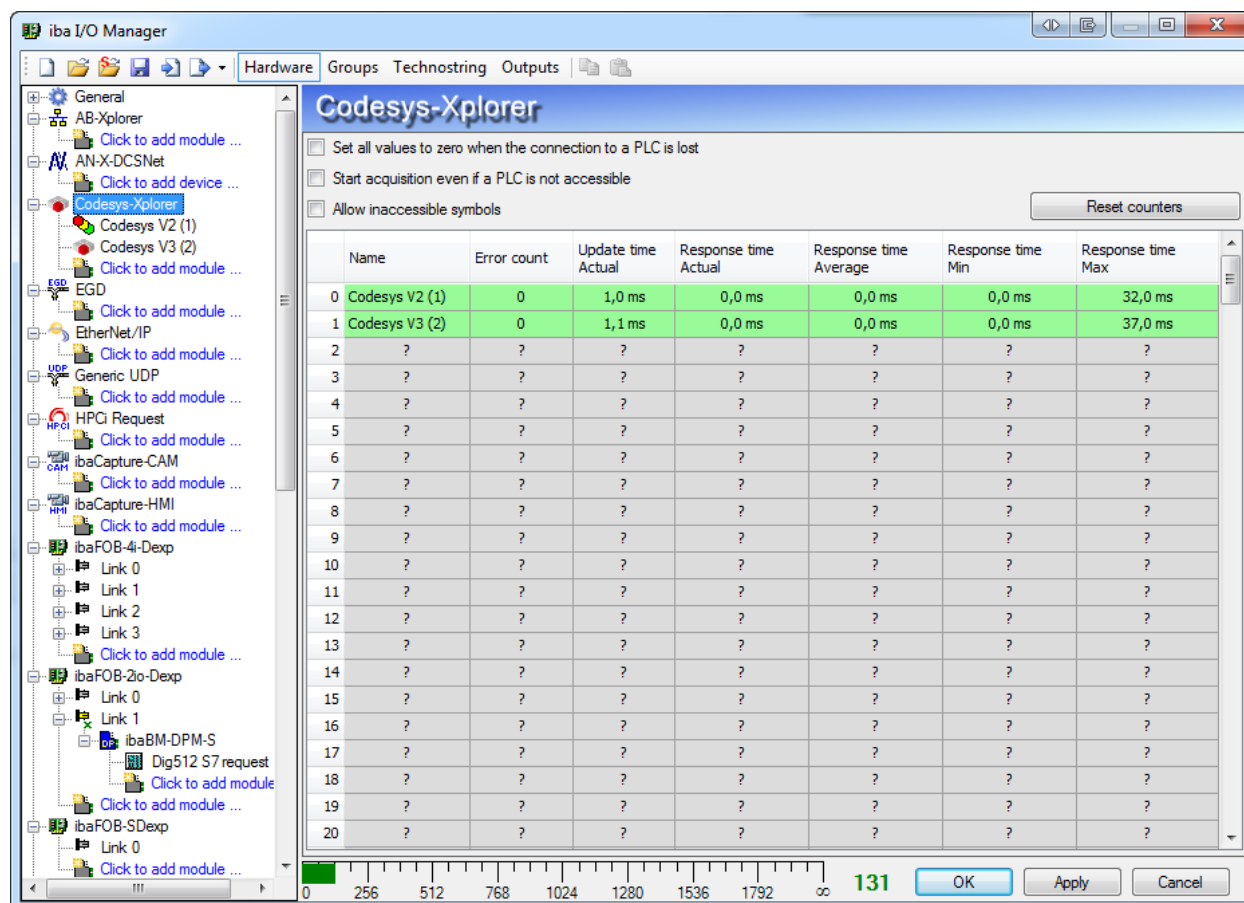


Timers are associated with the operand **T** while counters are represented by the operand **C** (or **Z** in German). The actual value that is displayed in PDA is the 16-bit integer retrieved from the S7 CPU. Note that the *Address* does not refer to a byte address in the case of a timer or counter but merely indicates which timer or counter number is requested.

In S7-200 CPUs, the timer and counter values can be requested as signed or unsigned 16-bit integers; in the case of S7-300/400, counters can only be unsigned 16-bit integers (with a maximum value of 999) while timers are converted in PDA to floating point numbers in units of second.

2 Codesys-Xplorer

The Codesys-Xplorer interface can be used to measure data from Codesys-based PLCs. It is an Xplorer interface which means that the data is cyclically read by ibaPDA instead of being sent by the PLC. The communication path used by ibaPDA is the same as the one the Codesys OPC server uses. It is a low priority communication that can be used to read the configured symbols.



The Codesys-Xplorer interface shows a table of the available connections. Per Codesys-Xplorer license you get 16 connections. A maximum of 240 connections is allowed. This means that maximum 15 licenses can be used. Each connection corresponds with a row in the table. The row is green when the connection is ok and data is being read. The row is orange when the connection is ok but the data is coming in slower than the configured update time. The row is red when the connection could not be established. The row is grey when there is no connection configured. The response time is the time it takes to read the data for a connection. The table shows the actual, average, minimum and maximum of the response time. The update time is the time between 2 reads. You can use the “Reset counters” button to clear the counters for all connections.

On the interface you can also decide how to handle some error conditions:

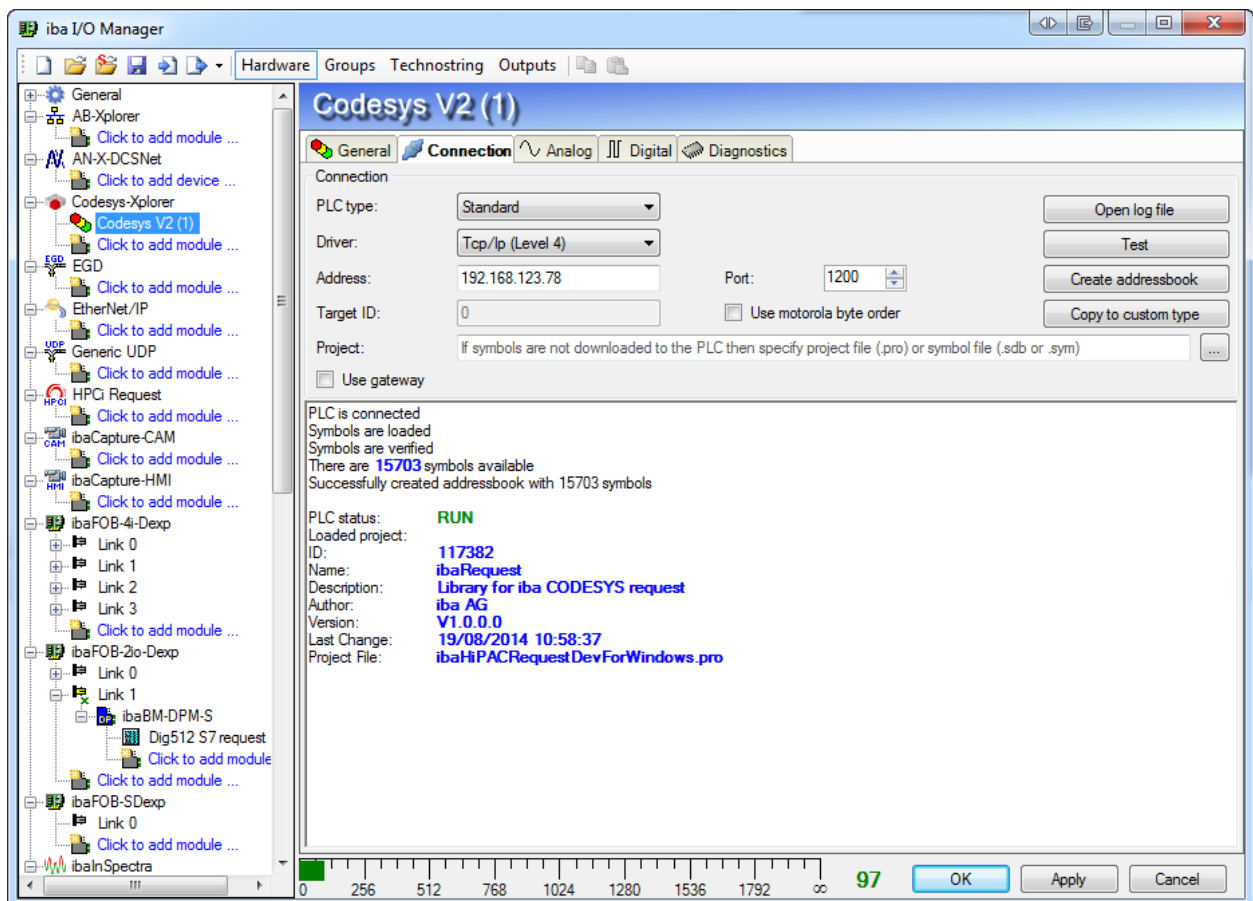
- When the connection to a PLC is lost during the acquisition then you can choose if the values stay at the last read value or if they are set to zero.
- When a PLC is not accessible during the start of the acquisition then you can choose if the acquisition starts without this PLC or if the acquisition is not started. When the acquisition is started without the PLC then ibaPDA will periodically try to connect to the PLC during the acquisition. As long as the PLC is disconnected the values will remain at zero.

- When the addressbook isn't up to date then it can contain a variable that is no longer available. When ibaPDA then tries to read the data for this variable an error will be returned by the PLC. If the option "Allow inaccessible symbols" is enabled then ibaPDA will ignore this signal and start the acquisition without this signal. If the option is not enabled then the acquisition will not start.

There are 2 module types that can be mapped to the Codesys-Xplorer interface:

- Codesys V2 module for PLCs based on Codesys version 2.x
- Codesys V3 module for PLCs based on Codesys version 3.x

They only differ in the way you configure the connection to the PLC. The configuration of the connection to a Codesys V2 based PLC is a bit complicated since many manufacturers of Codesys based PLCs have made changes to the communication part.



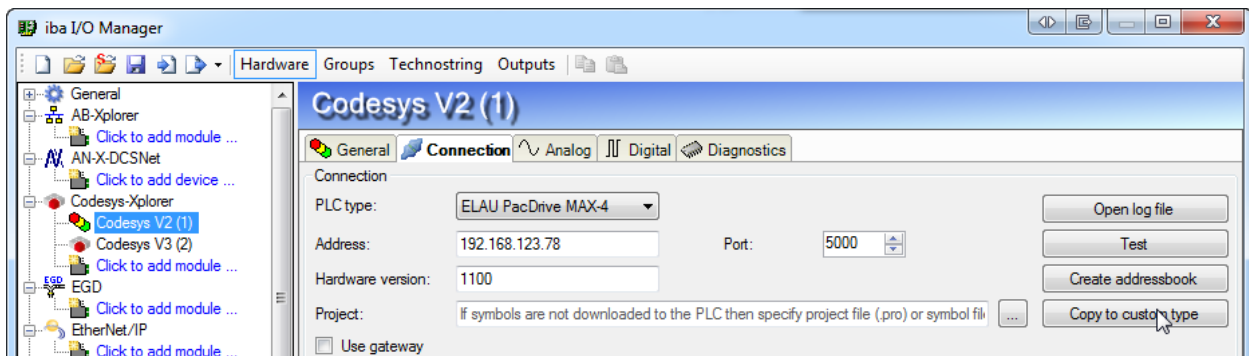
You first have to choose the PLC type. There are 6 options:

- Standard
- Standard without login
- ABB
- ELAU Standard
- ELAU PacDrive MAX-4
- Custom

The standard type can be used for most Codesys V2 PLCs. The connection to the PLC is made via Ethernet. There are 3 possible drivers:

- Tcp/Ip (Level 2)
- Tcp/Ip (Level 2 Route)
- Tcp/Ip (Level 4)

Which driver you have to choose will depend on the PLC you want to connect to. The Codesys windows PLC supports Tcp/Ip (Level 2 Route) and Tcp/Ip (Level 4). You will have to enter the IP address or host name of the PLC and the port number. Standard the port number is 1200 but many manufacturers have chosen a different port. You also have to set the “Use Motorola byte order” option correctly. Check it if the PLC uses big endian byte order. The target ID is only used when TCP/IP (Level 2 Route) is selected as driver. You should enter the ID of the target PLC and the address and port number should be from the PLC that acts as a router. If the symbol configuration is not stored on the PLC then you can enter the path to the symbol file (.sdb or .sym) or the project file (.pro).

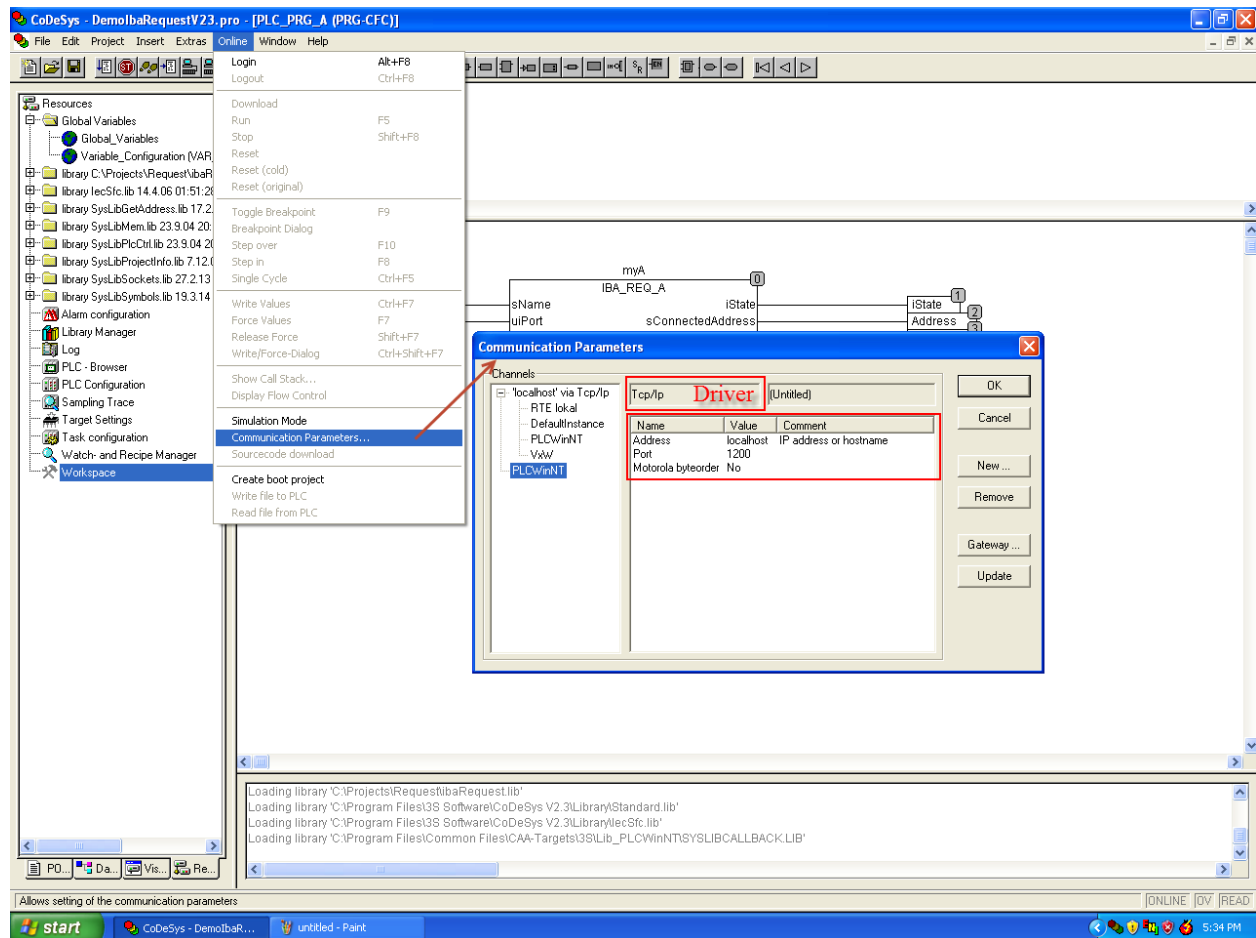


The standard without login PLC type is completely the same as the standard one except that ibaPDA won't try to login to the PLC. Depending on the connected PLC this is required or not. The ABB PLC type looks the same as the standard PLC type. The ELAU types are a bit different. Here you just have to specify the address and port number. In case of the PacDrive MAX-4 you also have to specify the hardware version.

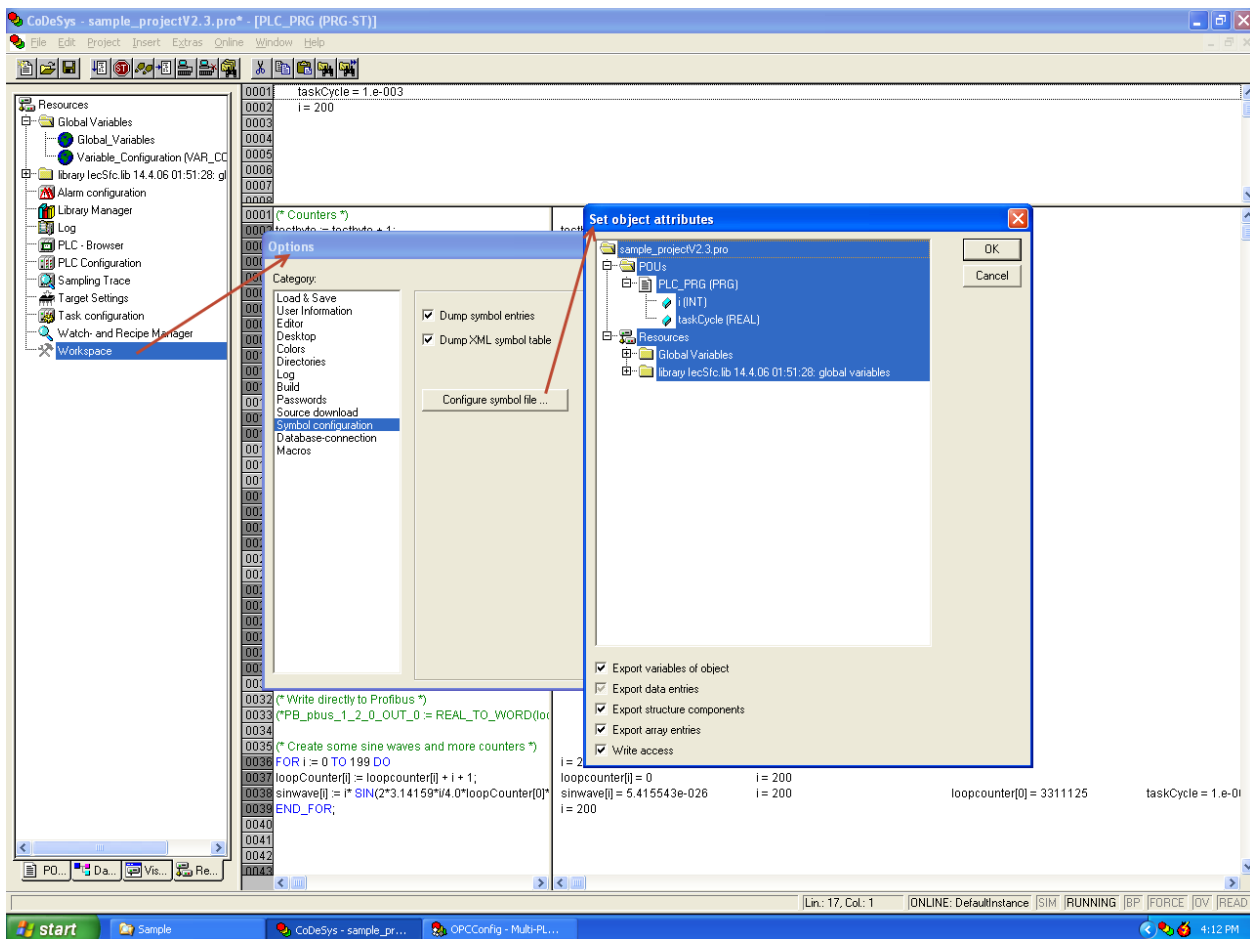


Internally ibaPDA will generate an INI file that contains all the settings for the connection. These INI files are stored in the Codesys subdirectory of the ibaPDA service installation directory. The custom PLC type allows you to define the contents of the INI file yourself. The easiest way is to start from another PLC type and then clicking the “Copy to custom type” button. This will copy the settings from that PLC type to the custom configuration file. This could be used by iba support to experiment with settings in order to try to connect to a new PLC type.

The easiest way to get the correct parameters is to look at the programming software and see which communication parameters are used there to go online. The following screenshot shows an example.

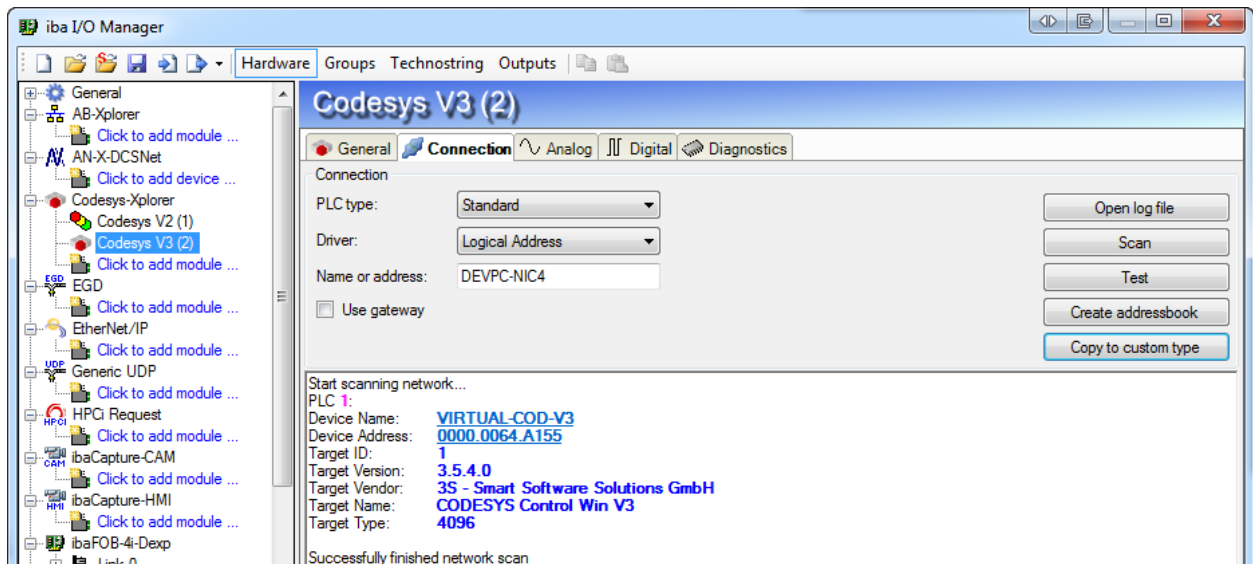


You can use the “Test” button to try to establish a connection. If there are no symbols configured then the connection will not work. In the Codesys programming software you can configure symbols via Workspace -> Symbol configuration -> Configure symbol file...



You will have to select the variables in the tree and then set the desired check boxes. You need to set “Export variables of object”. “Export data entries” is not needed. “Export structure components” is needed when you wish to measure members of a structure and “Export array entries” is needed when you wish to measure elements of an array. “Write access” is not required for ibaPDA.

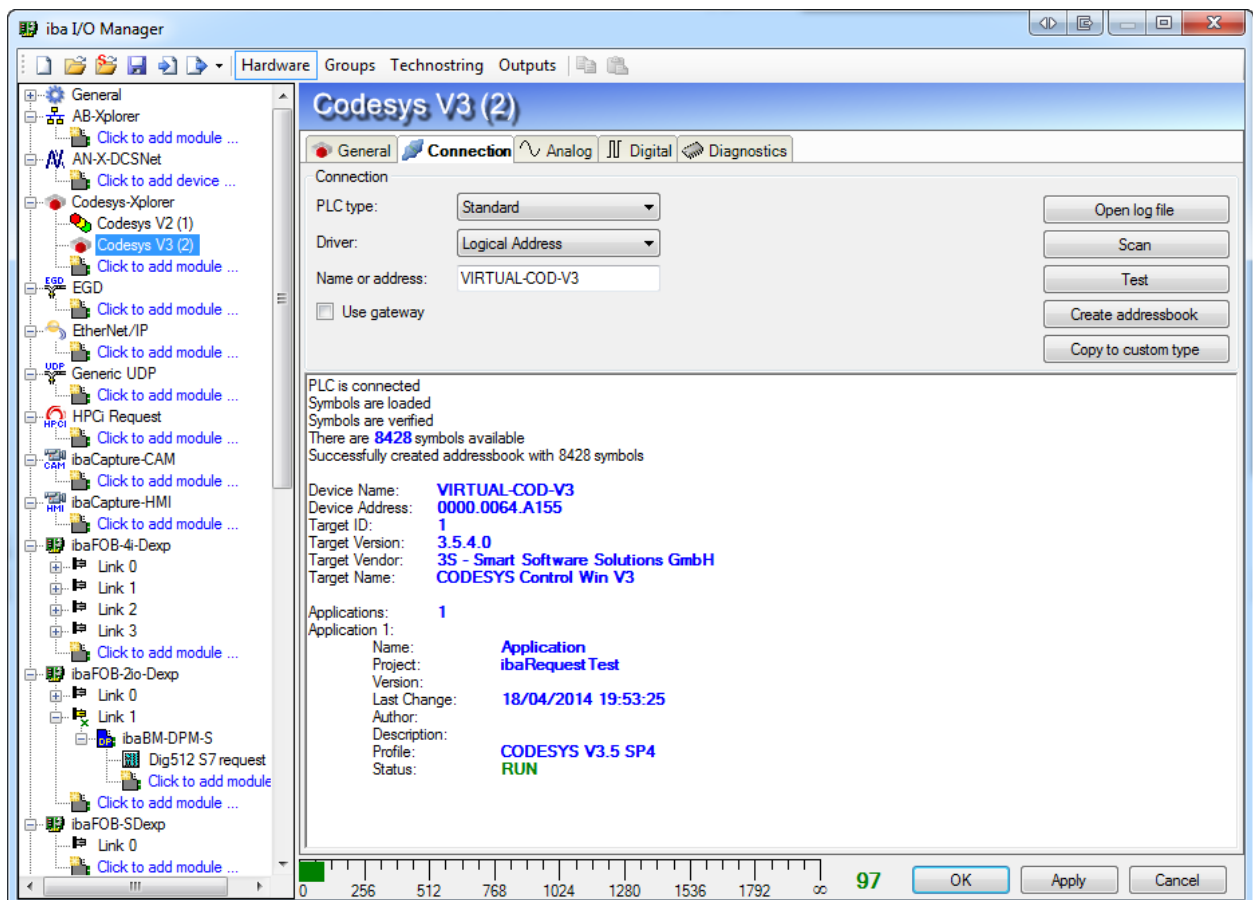
The Codesys V3 module is easier to configure. There are only 2 PLC types: standard and custom. Custom allows you again to manually create an INI file. In the standard type you can scan the network for available PLCs. If a PLC is found then you can click on its name or address to try to connect to the PLC. In order to be able to connect you will need to have TCP access to the PLC. This means that you either have to be in the same subnet or there should be a router in between.



The driver type is used to switch between 2 connection modes:

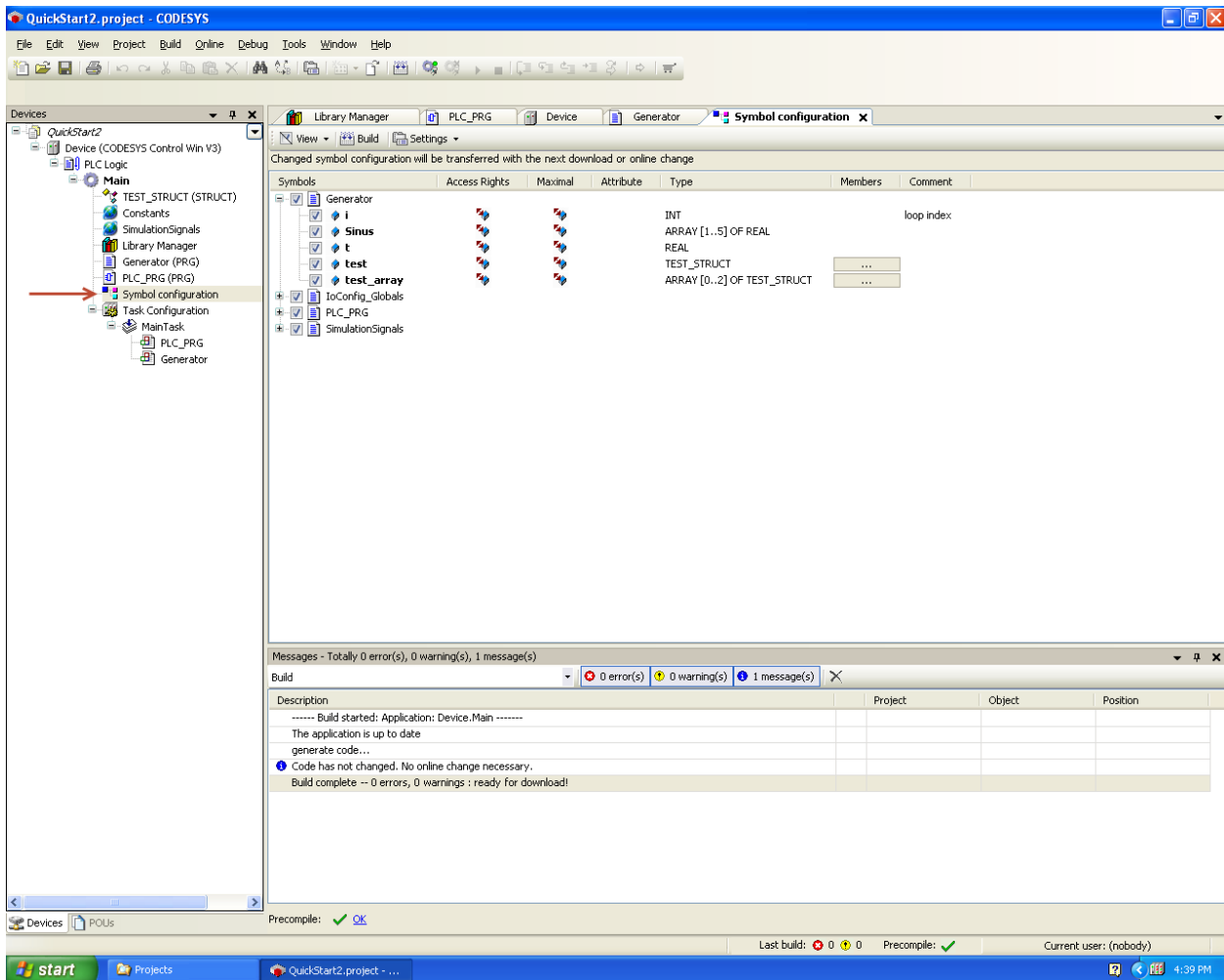
- Logical address: PLC name or address is used to connect
- CmpBlkDrvTcp: IP address and port number are used to connect

Normally the logical address mode will be used for Codesys V3 based PLCs. You can also use a gateway to connect to the PLC. It can be either a local gateway or a remote gateway. For a remote gateway you will have to enter its IP address and port number.

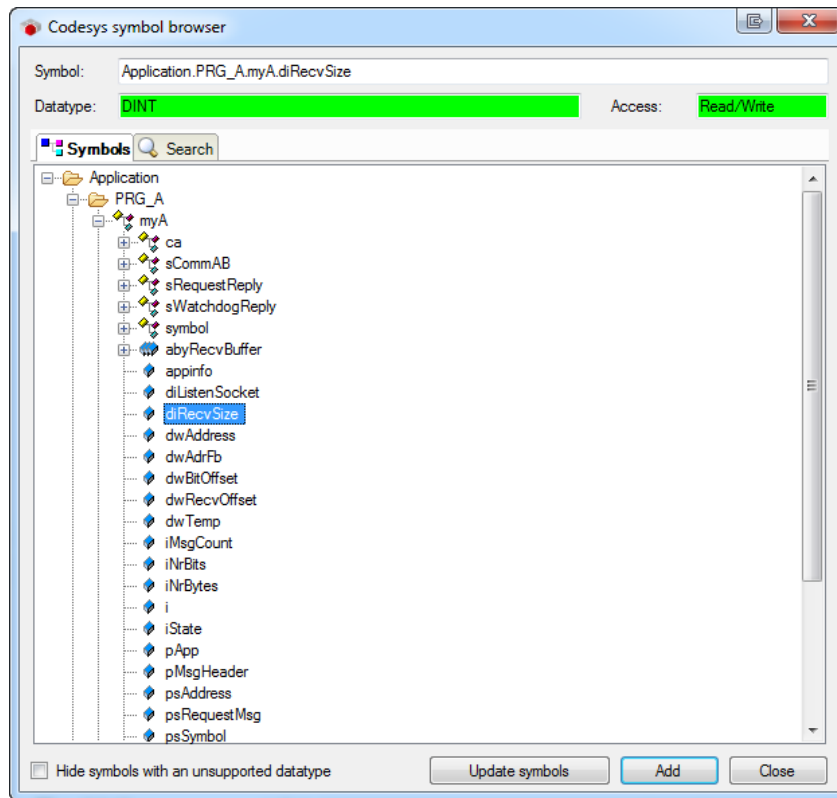


When you test the connection you get some information about the connected PLC. If there are no symbols configured in the PLC then the connection will fail. You can configure symbols in the

Codesys software. You have to add a symbol configuration to the project and then select which symbols you want to make available to ibaPDA. The screenshot below shows an example.



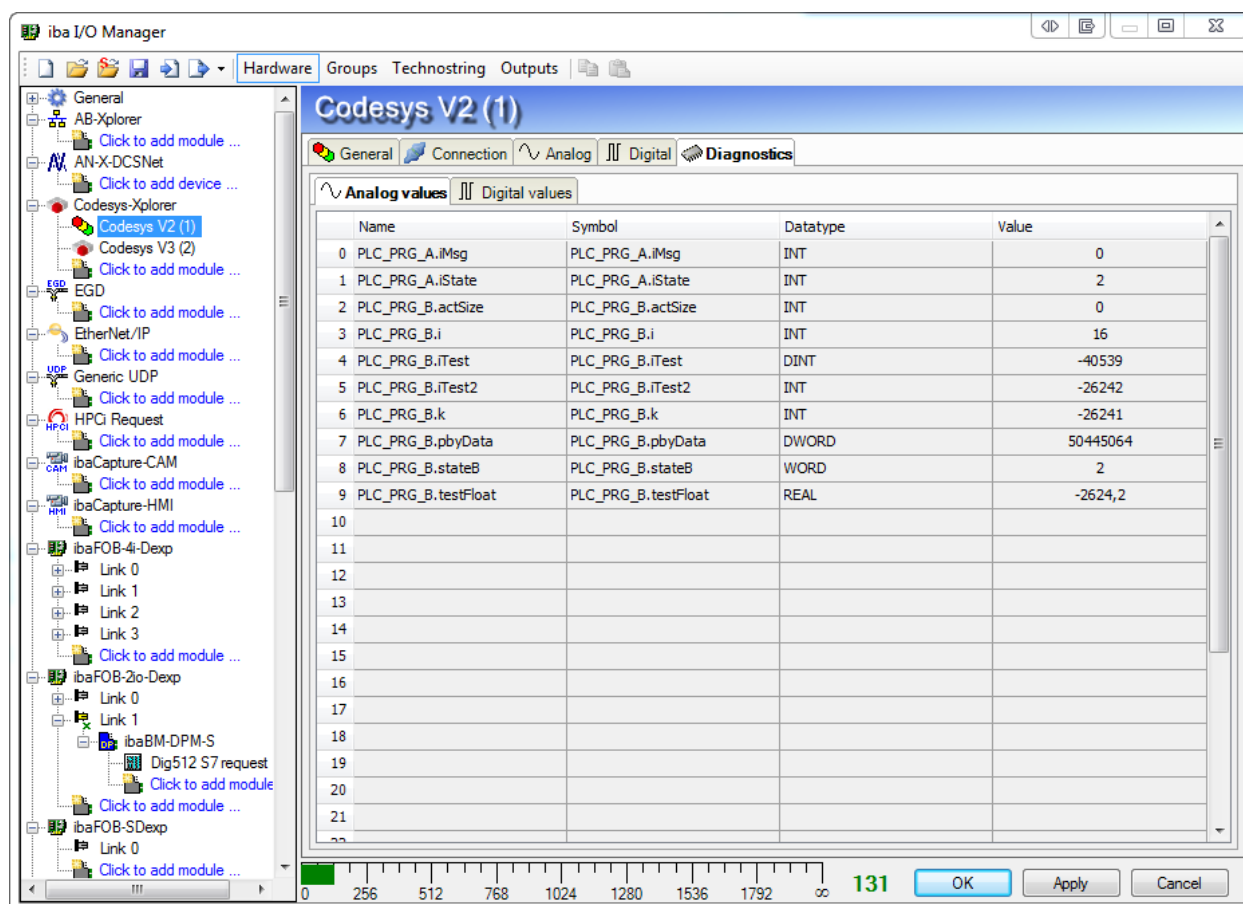
Once a connection to a V2 or V3 PLC is established you can create the addressbook. On the general tab of the module there is a hyperlink "Select symbols" to open the symbol browser.



You can select single or multiple symbols in the tree. Click the add button to add them to the corresponding analog or digital signal grid. If you selected a single symbol then the next symbol will be selected after you clicked the add button. This allows you to hit add multiple times in order to add consecutive symbols. You can also double click a symbol to add it to the signal grid. Use the update symbols button to read the addressbook from the PLC again.

On the search tab you can search symbols by name. The search result tree works in the same way as the complete symbol tree.

On the general tab of the module you can also configure the update time. This determines how fast ibaPDA tries to retrieve data from the PLC. The actual update time might be higher if the PLC is unable to deliver the data fast enough. You can check the connections table on the Codesys-Xplorer interface to see how fast the data is actually coming in.



When you have selected all the symbols you want to measure and set the update time then you can apply the configuration. IbaPDA will then make a connection to the PLC and do periodic reads. The module has a fifth tab called Diagnostics that shows the values of all the signals that are currently being read from the PLC. The grey rows are signals that are inactive.

Tested PLCs:

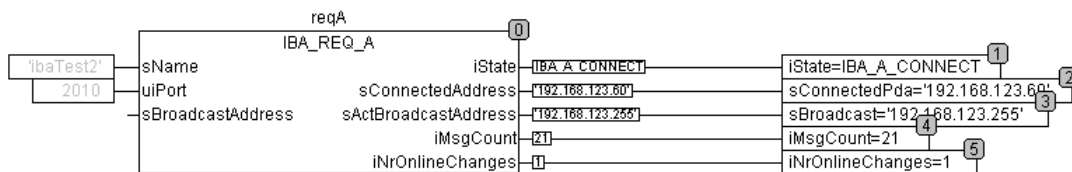
PLC	Codesys Version	Driver	Port	Remarks
3S Codesys SP PLCWinNT V2.4	V2.3	Standard TCP/IP (Level 4)	1200	
3S Codesys SP RTE	V2.3	Standard TCP/IP (Level 4)	1200	
Wago 750-808	V2.3	Standard TCP/IP (Level 2)	2455	Symbols were not stored in PLC but read directly from SDB file.
ELAU PacDrive C600	V2.3	ELAU Standard	5000	PLC Status can't be read out
Danieli HiPAC	V2.3	Standard TCP/IP (Level 4)	1200	
ABB AC500 PM554-TP-ETH	V2.3	ABB TCP/IP (Level 4)	1201	
ifm CR1051	V2.3	Standard without login	1200	Symbols were not stored in PLC but read

		TCP/IP (Level 4)		directly from SDB file.
Schneider Electric LMC 101C	V3	-	-	
Schneider Electric M258	V3	-	-	

3 HiPAC Request

The Danieli HiPAC system is a PLC based on a quad core Intel Core i7 CPU running VxWorks with the Codesys V2.3 runtime. Iba has developed a request system for this PLC. A request system consists of an agent running on the PLC that communicates with ibaPDA via a control path. TCP/IP is used as the control path. The user of ibaPDA uses a browser of all the available symbols in the PLC to select a list of symbols that he wants to measure. IbaPDA then sends this list of selected symbols to the agent. The agent validates this list and then cyclically copies the values of the signals on to the data path. The data path can be a reflective memory board or a generic UDP connection. In the future more data paths can be supported. IbaPDA will then measure the data from the data path.

In the PLC you have to add the ibaHiPACRequest library to your project. This library requires some additional libraries that should be available when the HiPAC runtime image is up to date. Please contact Danieli in case libraries are missing. The ibaHiPACRequest library contains the agent. The request agent is split into 2 function blocks: IBA_REQ_A and IBA_REQ_B. The A block will be put in a (slow) low priority task and is responsible for the TCP communication with ibaPDA. It will also check the list of symbols. The B block will be put in a faster higher priority task and is responsible to gather the signal data and pack it together in the data buffer that will be copied onto the data path.



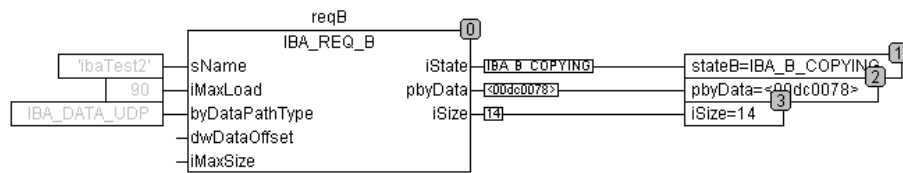
These are the inputs and outputs of the IBA_REQ_A block:

Name	Type	In/Out	Description
sName	STRING(20)	IN	Name of the function block. The same name must be used for the corresponding IBA_REQ_B function block. The name must be unique across all HiPAC CPUs connected to the same ibaPDA.
uiPort	UINT	IN	Port number of the TCP listen socket
sBroadcastAddress	STRING(20)	IN	Optional IP address where the function block sends broadcast messages to. If this is left open then the block will try to find the address automatically by retrieving its own IP address and assuming a subnet mask of 255.255.255.0.
iState	IBA_STATE_A	OUT	State of the function block
sConnectedAddress	STRING(20)	OUT	IP address of the connected ibaPDA
sActBroadcastAddress	STRING(20)	OUT	IP address that is being used for broadcast messages
iMsgCount	INT	OUT	Counter of the messages sent to ibaPDA
iNrOnlineChanges	INT	OUT	The number of online changes that have been detected by the function block.

The IBA_REQ_A block can have the following states (IBA_STATE_A):

State	Description
IBA_A_INIT	Initial state before the block has registered itself with its name
IBA_A_OPEN	Block is trying to open a listen socket on port uiPort

IBA_A_WAIT_FOR_CONNECT	Listen socket is opened and block is now waiting for an incoming connection of ibaPDA
IBA_A_CONNECT	A connection with an ibaPDA is established and messages are being exchanged

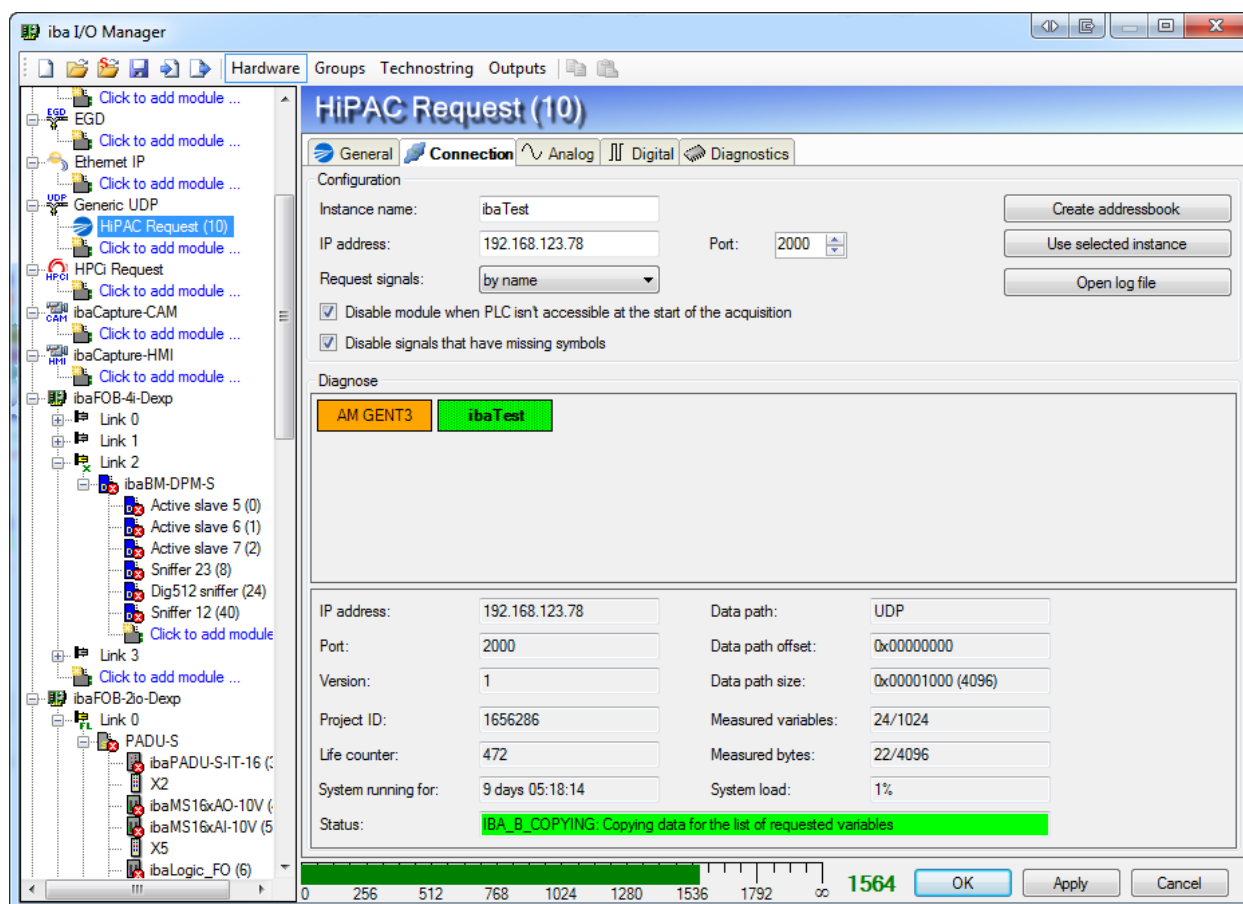


These are the inputs and outputs of the IBA_REQ_B block:

Name	Type	In/Out	Description
sName	STRING(20)	IN	Name of the function block. Must be the same as the corresponding IBA_REQ_A function block
iMaxLoad	INT	IN	Maximum CPU load in %. If the CPU load is over this maximum then the copying is stopped
byDataPathType	IBA_DATA_PATH_TYPE	IN	The data path type used. There are 2 possible options: <ul style="list-style-type: none"> IBA_DATA_RM: Reflective memory. The offset and size of the buffer reserved on the reflective memory board for request on this CPU is automatically retrieved via functions from the RFMPDA library. IBA_DATA_UDP: UDP connection. The destination address and port number are automatically retrieved from the ibaPDA instance connected to the A block at the start of the acquisition.
dwDataOffset	DWORD	IN	An optional extra offset within the data path buffer. This only needs to be used in case there are multiple B blocks on the same CPU writing to the same data path.
iMaxSize	INT	IN	Set to 0 to use the complete data path. If there are multiple B blocks on the same CPU writing to the same data path then set the maximum size this block can use.
iState	IBA_STATE_B	OUT	State of the function block
pbyData	POINTER TO BYTE	OUT	Pointer to the data buffer
iSize	INT	OUT	Actual size of the data in the data buffer. Valid when iState is IBA_B_COPYING

The IBA_REQ_B block can have the following states (IBA_STATE_B):

State	Description
IBA_B_INIT	Initial state. Searching for the IBA_REQ_A block with the same name.
IBA_B_NO_DATA_PATH	Connected to A block with the same name but no data path is available
IBA_B_READY	Connected to A block and data path found. Empty variable list
IBA_B_VALIDATE	Validating new list of variables
IBA_B_COPYING	Copying data for the list of variables
IBA_B_OVERLOAD	A PLC overload was detected while validating or copying. Copy has been stopped.
IBA_B_ONLINECHANGE	An online change has occurred and waiting for block A to react to it



In ibaPDA there are 2 module types for HiPAC request:

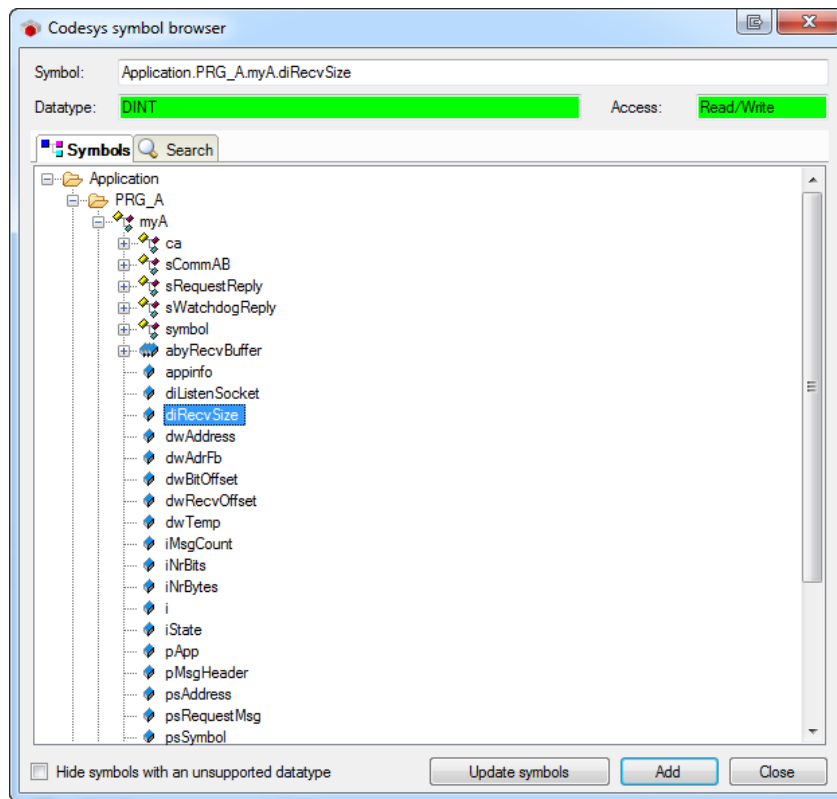
- HiPAC request on the Generic UDP interface corresponding with IBA_DATA_UDP on the IBA_REQ_B block
- HiPAC request on the Reflective Memory interface corresponding with IBA_DATA_RM on the IBA_REQ_B block

They both have the same connection tab. You have to configure the instance name, IP address and port number of the IBA_REQ_A block you want to connect to. In the diagnose section you can see a box for each A block that we are receiving broadcast messages from or that is configured in the current I/O configuration. The color of the box depends on the state of the connection:

Appearance	Configured	Receiving broadcast	TCP connection ok	Data path ok
Orange		x		
Red	x			
Red blinking	x	x		
Yellow	x	x	x	
Yellow with exclamation mark	x		x	
Green	x	x	x	x
Green with exclamation mark	x		x	x

When you select a box by clicking on it then you see some more information about this instance. You can double click a box to fill in its instance name, IP address and port number in the configuration section. Alternatively you can select the box and then click the “Use selected instance” button.

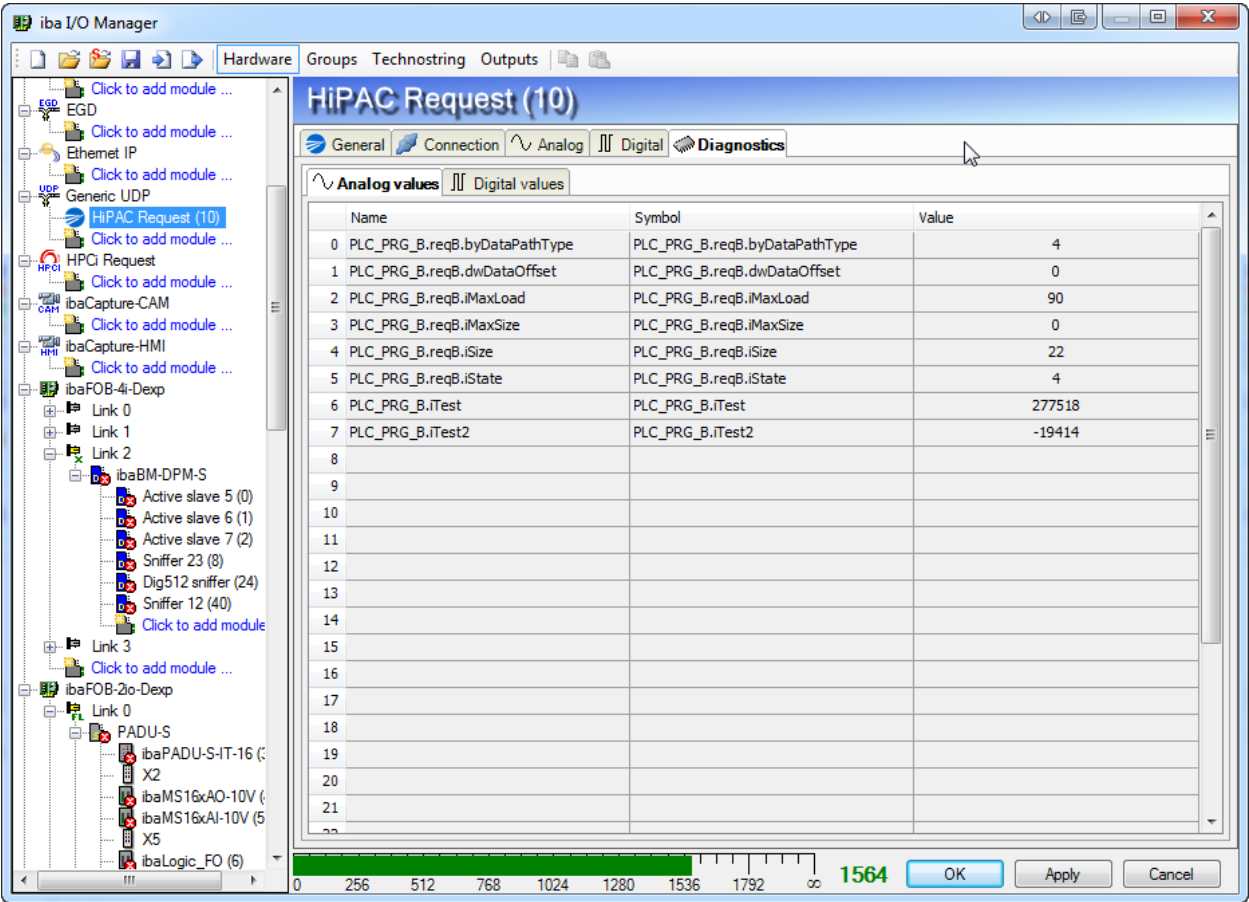
When you have configured an instance then you can use the “Create addressbook” button to retrieve the addressbook. This uses the same mechanism as Codesys-Xplorer. All the symbols that are part of the configured symbol configuration in the PLC are available for selection. The symbol browser is exactly the same as the one from Codesys-Xplorer.



In the configuration section you can also decide how to handle some error conditions:

- When a PLC is not accessible during the start of the acquisition then you can choose if the acquisition starts without this PLC or if the acquisition is not started. When the acquisition is started without the PLC then ibaPDA will periodically try to connect to the PLC during the acquisition. Once the PLC reconnects then the acquisition is automatically restarted.
- When the symbol configuration has changed then the module can contain a symbol that is no longer available. When ibaPDA then tries to read the data for this variable an error will be returned by the PLC. If the option “Disable signals that have missing symbols” is enabled then ibaPDA will ignore this signal and start the acquisition without this signal. If the option is not enabled then the acquisition will not start.

For the HiPAC request module on Generic UDP you also have to configure the UDP port where data needs to be sent to. You can do this in the property grid on the general tab.

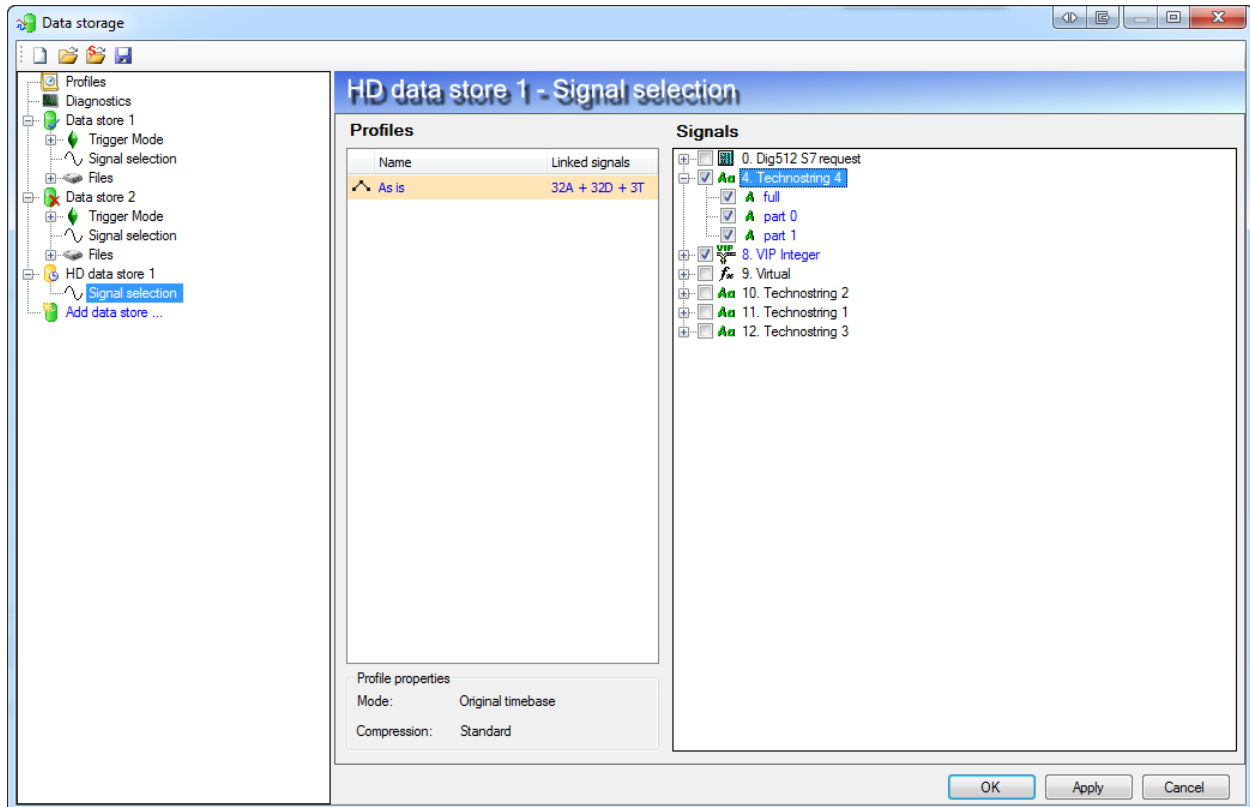


After you apply the configuration you can go to the Diagnostics tab and see the actual values of the requested symbols.

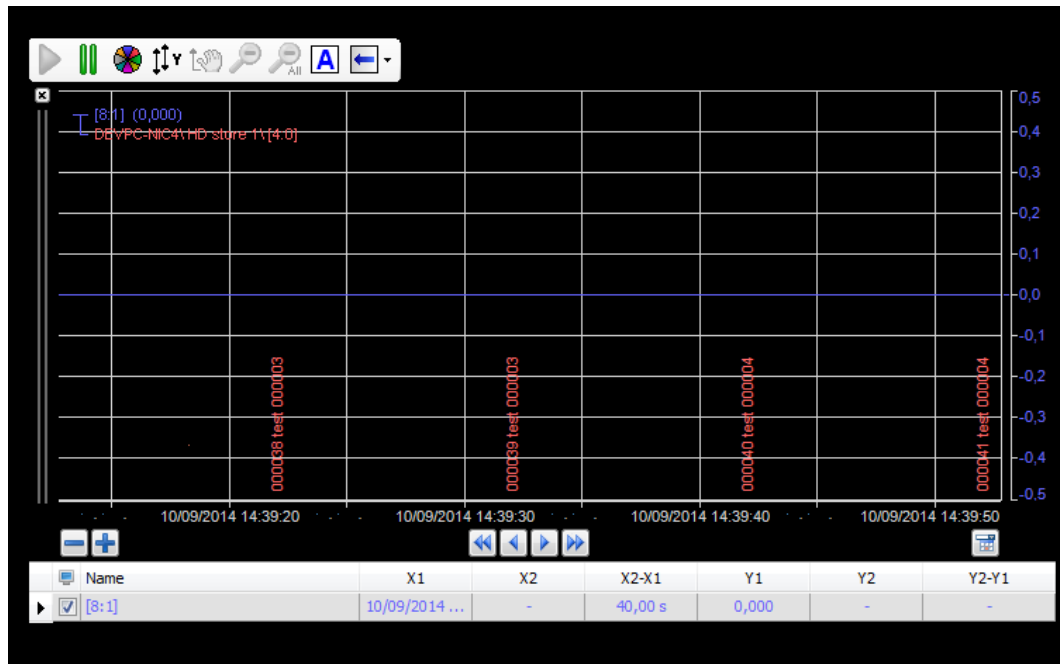
4 ibaHD server changes

4.1 Text channels

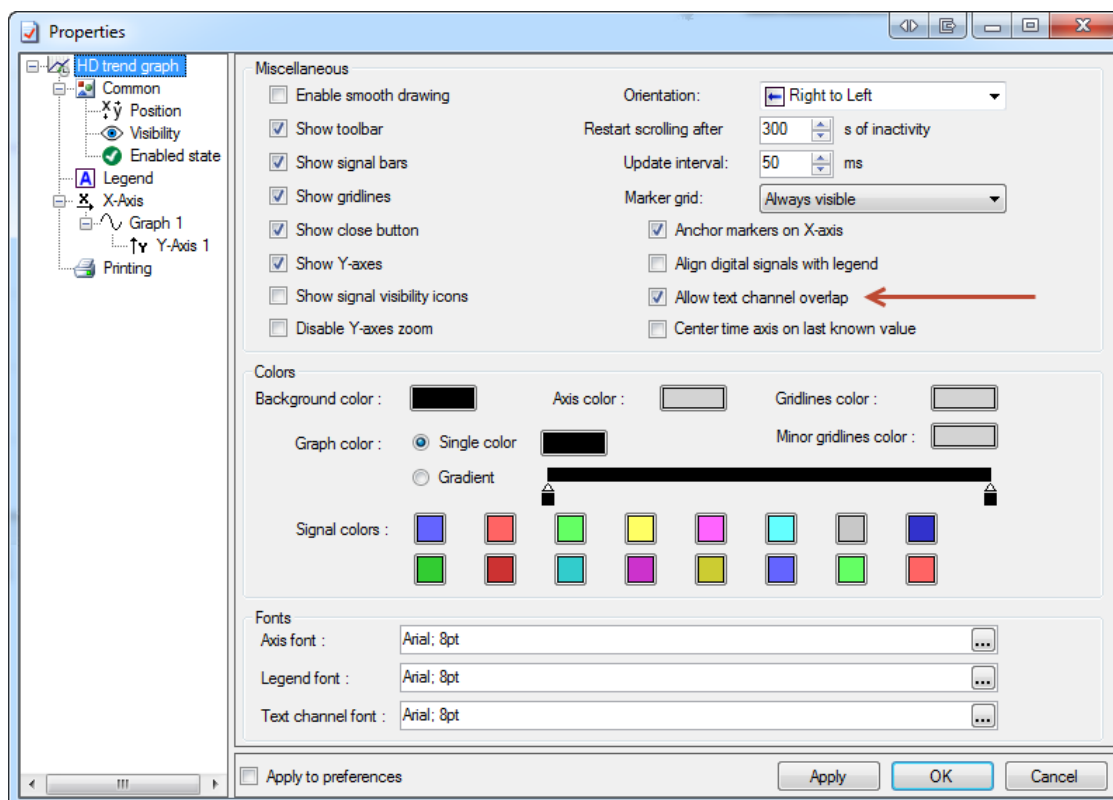
The HD server v1.4.1 can store text channels in timebased HD stores. In the data store configuration you can now assign the “As is” profile to text channels. You can only use the “As is” profile. This is the same as for normal dat file data stores.



The text channels are stored with a timestamp accuracy of 1ms. The HD server also uses levelling for text channels in the same way as for normal data channels. The text channels are stored with the following levels: raw level (1ms max), 5s, 3m20s, ~2h, ~88h. They can be shown in an HD trend graph. This only works when the HD trend graph is embedded in an ibaQPanel.



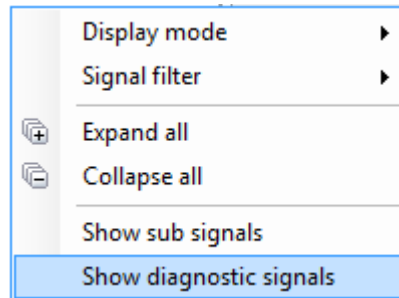
If you zoomout then the different texts will overlap. In order to still be able to read some of these texts you can disable the option “Allow text channel overlap” in the properties of the HD trend graph.



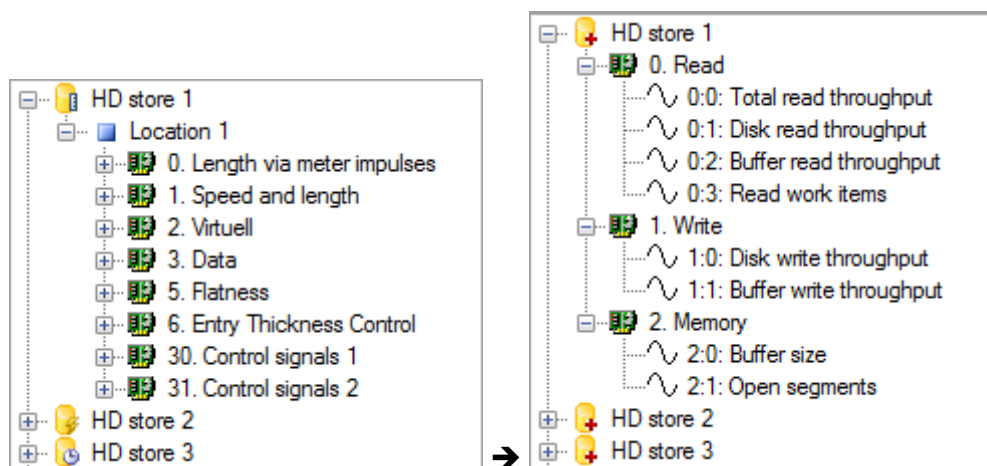
4.2 Diagnostic data

Since HD server v1.4.0 each HD store records diagnostic data when the store is active. When connected to HD server v1.4.1 it is now possible to display the diagnostic data in the ibaPDA client.

To do this the user must hold down shift while right clicking the HD signal tree. The context menu that appears will contain an extra item “Show diagnostic signals.”



When the item is checked the signal tree will replace each HD store with its diagnostic store.



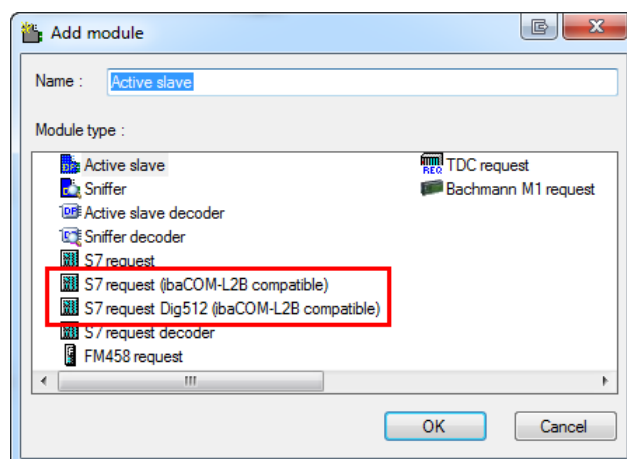
A diagnostic store always contains a fixed set of time based signals. In the future more signals can be added. The idea is that when the user experiences problems with his HD server, customer support can get an idea of the HD server activity at the time the problems occurred.

- Diagnostic signal data is sampled at 1 Hz
- The cleanup size of a diagnostic store is 512 MB
- The cleanup time span of a diagnostic store is 6 months

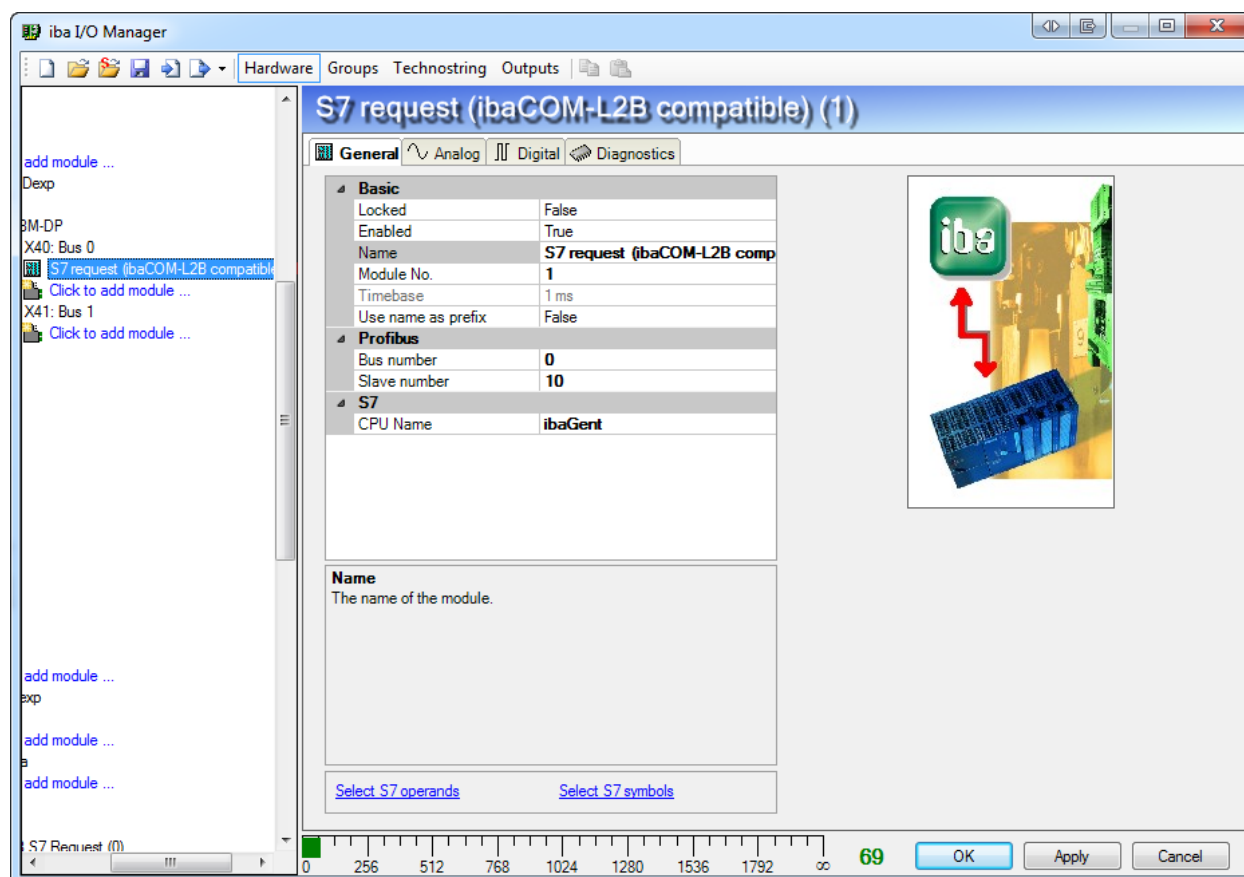
5 ibaBM-DP

5.1 S7 request compatible with ibaCOM-L2B

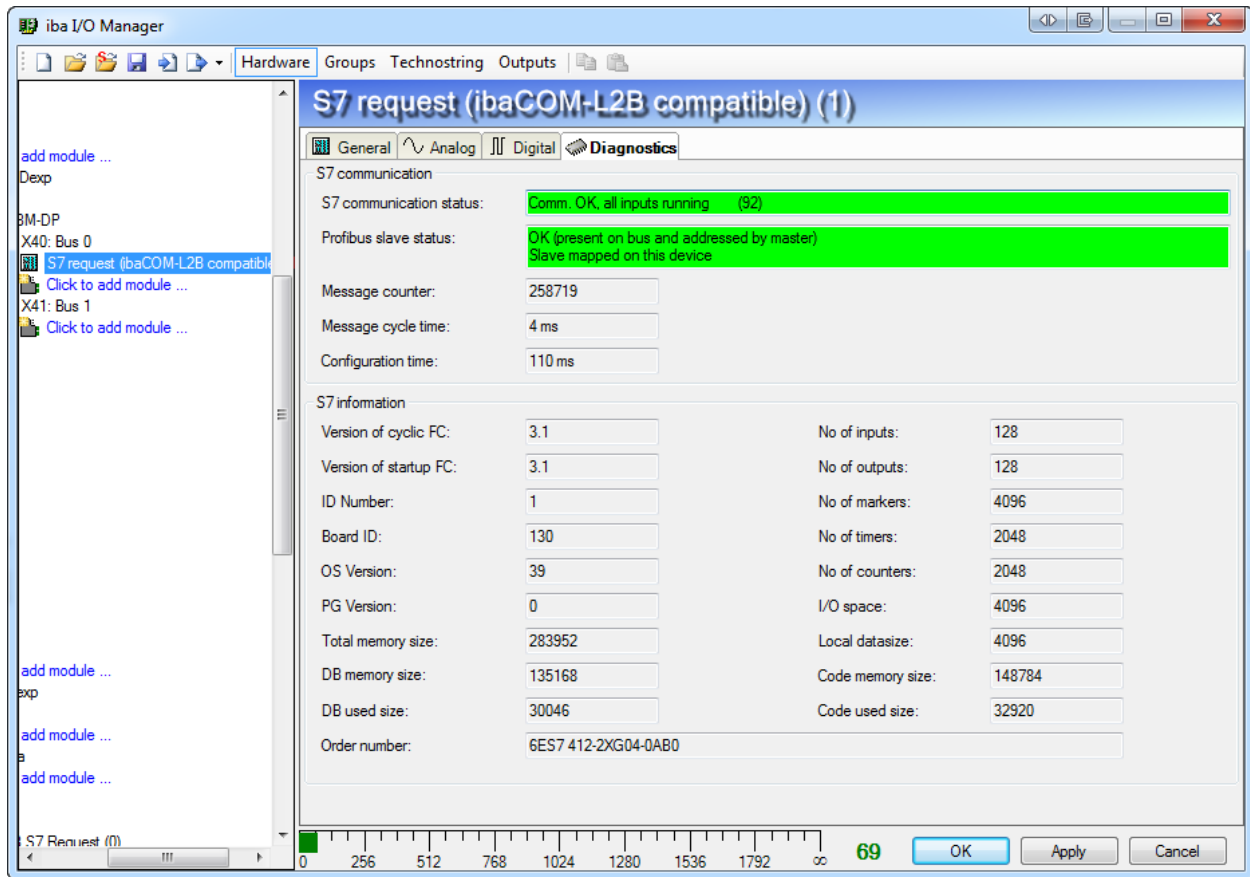
With the ibaCOM-L2B board you can do S7 request via the FC22, FC23 functions. The communication between ibaPDA and the S7 agent (FC22, FC23) is done over profibus via the ibaCOM-L2B board. Since the ibaCOM-L2B board is becoming obsolete the same S7 request communication has been implemented in the ibaBM-DP device with firmware v1.2 or later. The ibaBM-DP device can then be used as a 1 to 1 replacement for the ibaCOM-L2B board. For detailed information on how to setup this request system on the S7 side please read chapter 3 of the ibaPDA-Request-S7 manual.



On an ibaBM-DP device in ibaPDA you can now add the S7 request (ibaCOM-L2B compatible) and the S7 request Dig512 (ibaCOM-L2B compatible) modules. These behave in the same way as the L2B S7 request and L2B S7 request Dig512 modules on the ibaCOM-L2B board.



On the general tab you have to specify the profibus slave number that will be used to transfer the data.

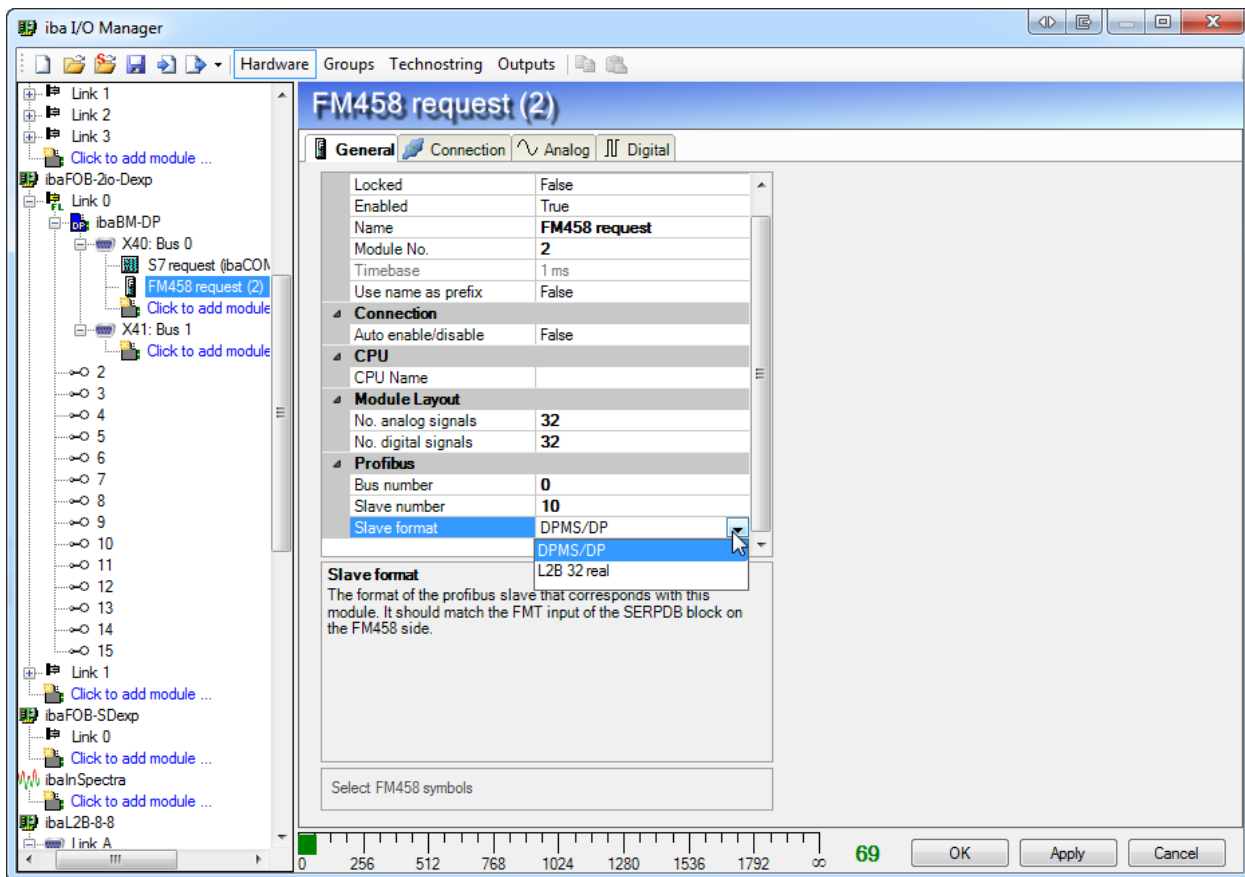


Once the acquisition is running you will see some diagnostic information on the Diagnostics tab. The displayed information is the same as shown on the Diagnostics tab of the L2B S7 request modules. The only difference are the following items:

- Message counter: Counts the number of data + control messages exchanged with the S7
- Message cycle time: The time it takes between 2 consecutive profibus messages.
- Configuration time: The total time it took to transfer and validate the requested list of operands

Feature	ibaCOM-L2B compatible S7 request via ibaBM-DP	S7 request via ibaBM-DP
Connections	FO connection from ibaPDA to ibaBM-DP Profibus connection from ibaBM-DP to S7	FO connection from ibaPDA to ibaBM-DP Profibus connection from ibaBM-DP to S7 Network or PC/CP connection from ibaPDA to S7
Data size	112 bytes	236 bytes
Functions	FC22, FC23, FC26	FC122

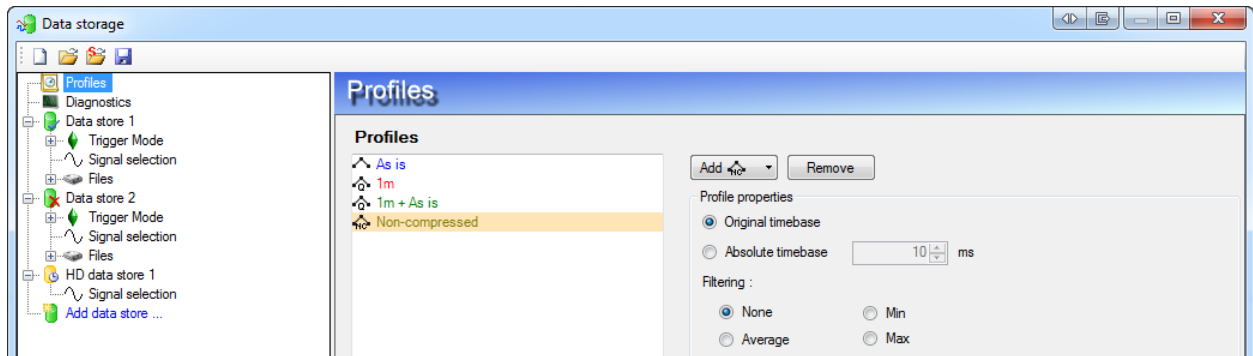
5.2 L2B real format supported for FM458 and TDC request



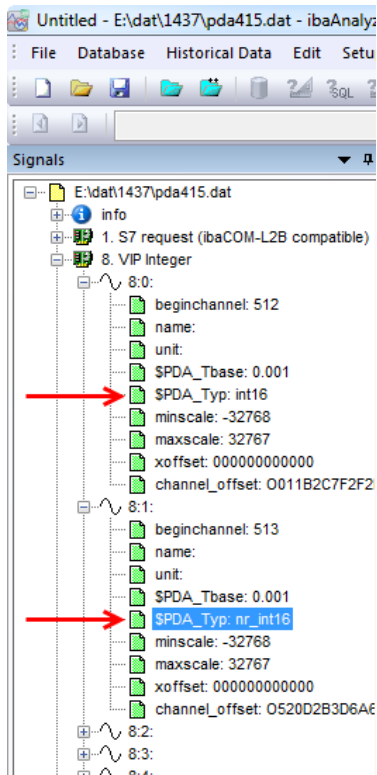
FM458 and TDC request can be done via ibaCOM-L2B and via ibaBM-DP/ibaBM-DPM-S. The device that is used is determined by the FMT input of the SERPDB function block. Basically the difference between the 2 formats is that via ibaBM-DP more bytes can be transferred than via ibaCOM-L2B. If a customer now replaces his ibaCOM-L2B with an ibaBM-DP then he needs to update his FM458/TDC program with another value for the FMT input. With v6.33.0 of ibaPDA this is no longer necessary since the ibaBM-DP/ibaBM-DPM-S request modules now also supports the ibaCOM-L2B format. The FM458 request module and TDC request module on ibaBM-DP and ibaBM-DPM-S now have a slave format property.

6 Non-compressed data

By default ibaPDA will compress the signal data inside a dat file. The compression works if there are multiple consecutive samples with the same value. If a signal has values that change with every sample than the compression actually generates more data than without compression. In the case of int16 data even 50% extra data is generated. In the case of float data 25% extra data is generated. This is typically the case for ICP signals and for fast analog signals like electric voltages and currents. For such signals ibaPDA now has new non-compressed profiles. These write the data uncompressed into the dat file. This saves space and it is also much faster to write and read these signals.



The non-compressed profiles have an icon with the letters NC in it. You can configure the timebase used. In case it is not the original timebase you can configure if the min, max or average value within the period of time is taken. The non-compressed profiles can only be used on analog signals, not on digital or text signals.



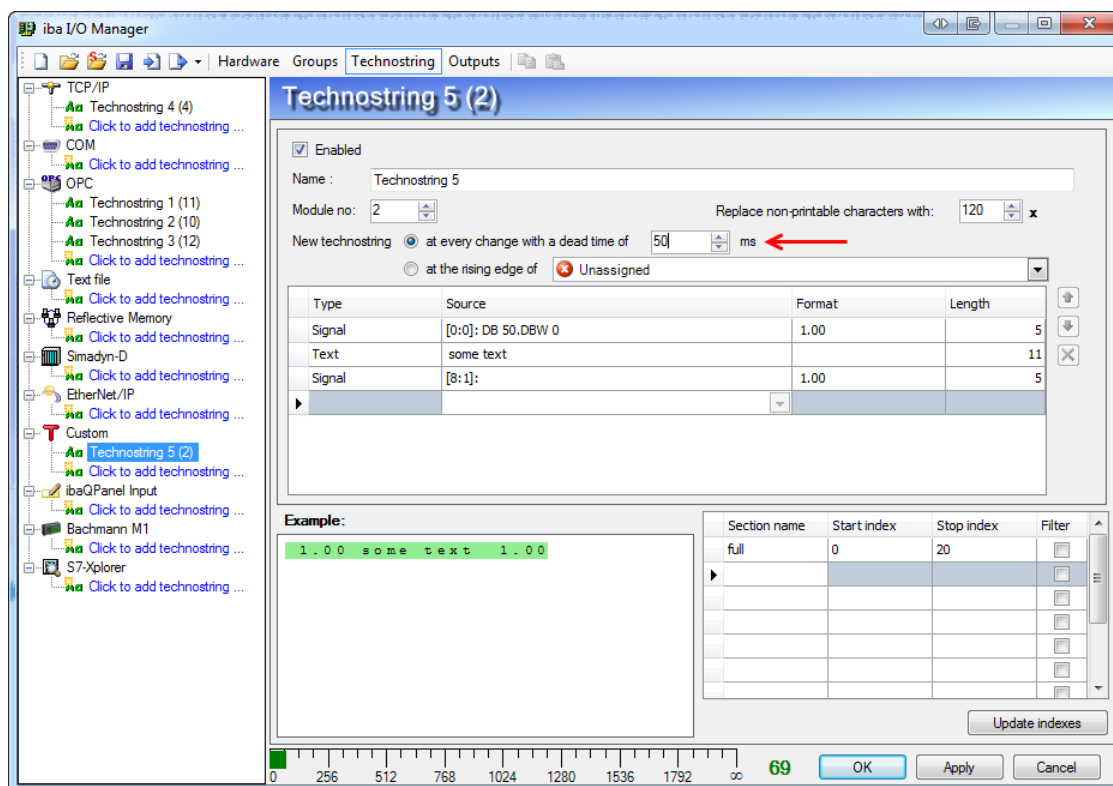
IbaAnalyzer v6.4.0 is required to read non-compressed data from a dat file. You can recognize if a signal is compressed or non-compressed by looking at the \$PDA_Type infofield of the signal. If it starts with nr_ then it is a non-compressed signal.

The non-compressed data can also be read by ibaFiles v6.3.0 and later.

7 Technostring changes

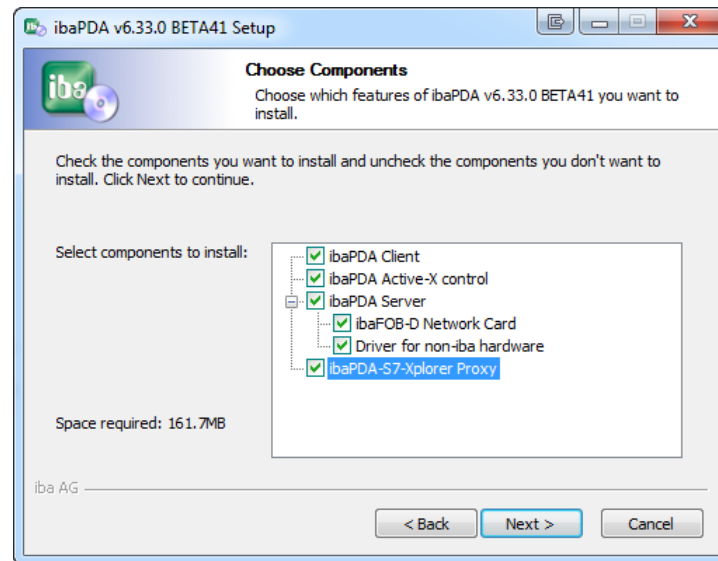
In ibaPDA data is processed in blocks. A block contains the samples of all signals for a time period that is equal to the least common multiple of all module timebases. So in case there are modules with timebase 1ms and modules with timebase 10ms then the blocks will be 10ms long and they will contain 10 samples for the 1ms signals and 1 sample for the 10ms signals. These blocks of data are generated by the driver. IbaPDA processes one or more of these blocks periodically. The period is at least 50ms and it is a multiple of the block time. In ibaPDA versions prior to v6.33.0 a technostring could only have a single value within one such processing period. This means that a technostring couldn't change faster than every 50ms. When a technostring was received then its timestamp was always set to the beginning of the next processing period. This wasn't very accurate. This principle led to all kinds of strange side-effects especially when using the technostring as a trigger or as a file name. This has now changed in ibaPDA v6.33.0. A technostring is accurately timestamped and it can have multiple values within one processing period. This makes the use of the technostring much more logical. These are the situations where technostrings are used and so which are impacted by this change:

- Dat file name
- Infofields in dat file
- Text channels in dat file
- Functions that work on text inputs (e.g. Technostring, TechnostringMsgCounter, AsciiValue, TextCompare, ...)
- E-mail outputs
- HD event store
- OPC technostring inputs



Because a technostring can now have lots of values we introduced a new dead time property to the custom technostring when it doesn't use a trigger. Any new change within dead time of the previous technostring doesn't generate a new technostring.

8 Installer changes



The ibaPDA installer has an extra component called ibaPDA-S7-Xplorer Proxy. By default it isn't selected. This can be selected independently from the other components. You can read more about this component in chapter 1.3. Use the command line option `/s7proxy` to enable the component.

The ibaPDA server component now has 2 subcomponents that are selected by default:

- ibaFOB-D network card
- Driver for non-iba hardware

The ibaFOB-D network card is used for the 32Mbit/s flex fiber optic protocol. In case you don't use the flex protocol you could unselect this component. When it isn't selected then the installer will uninstall the ibaFOB-D network card if it was already installed by a previous installation.

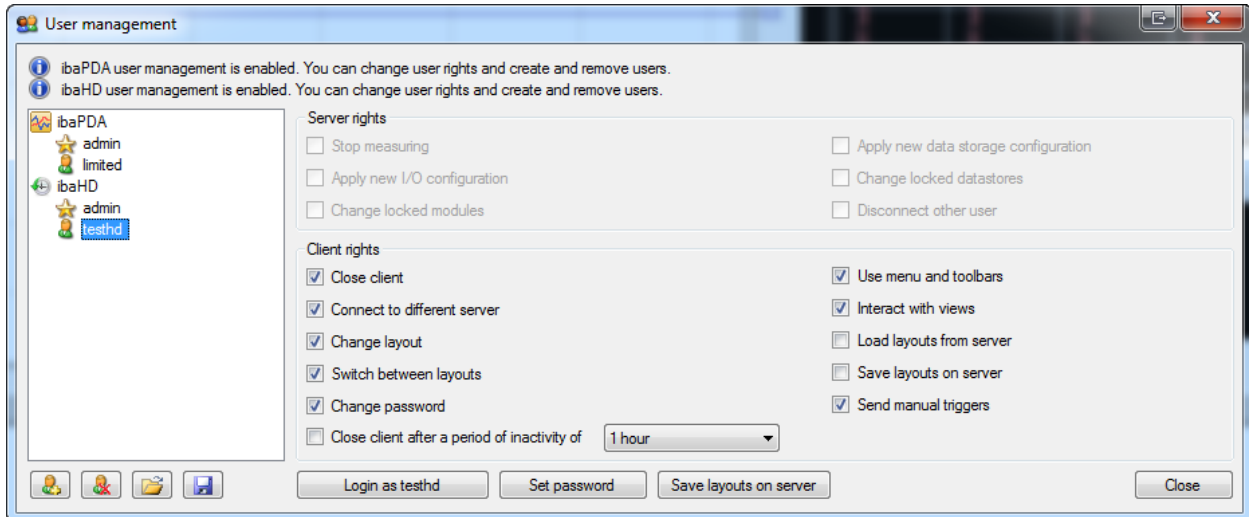
The driver for non-iba hardware component will install the iba driver for the following boards:

- Reflective memory
- CP1616
- DGM200P
- PCLink
- Scramnet+ SC150
- Toshiba ADMAP JAM11

The iba driver is required in case you want to measure data from these boards in ibaPDA. If you don't want to measure any data from these boards and you want to use the board in your own application then you can disable the installation of the ibaPDA driver.

9 User management changes

ibaPDA client user management has been extended to allow user creation on both ibaPDA server and ibaHD server. Creating users on an ibaHD server is only possible if ibaHD server v1.4.1 or later is installed.

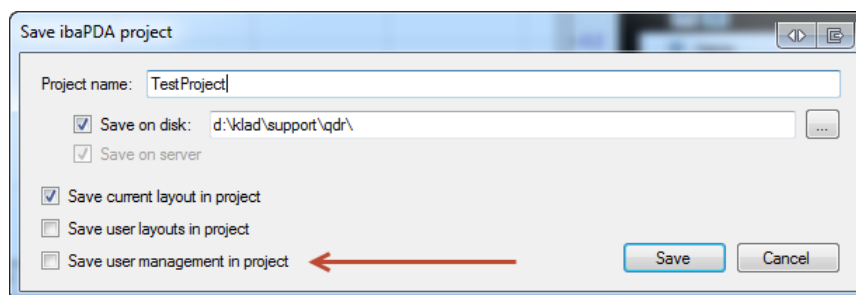


- For HD users, the server rights are always disabled
- If PDA server is not connected then the PDA user client rights are ignored
- If HD server is not connected then the HD user client rights are ignored
- If both PDA and HD server are connected then the client rights behave in different ways:
 - The following client rights need to be enabled on both PDA and HD server:
 - Switch between layouts
 - Change layout
 - Use menu and toolbar
 - Interact with views
 - Send manual triggers
 - The following client rights are evaluated separately for each server type:
 - Connect to different server
 - Save layout on server
 - Load layout from server
 - Change password
 - Other rights have custom behavior:
 - Close client
 - IF NOT ibaPDA right → ibaPDA admin password is asked
ELSE
IF NOT ibaHD right → ibaHD admin password is asked
 - Close client after a period of inactivity
 - MIN(ibaPDA timeout, ibaHD timeout) is used

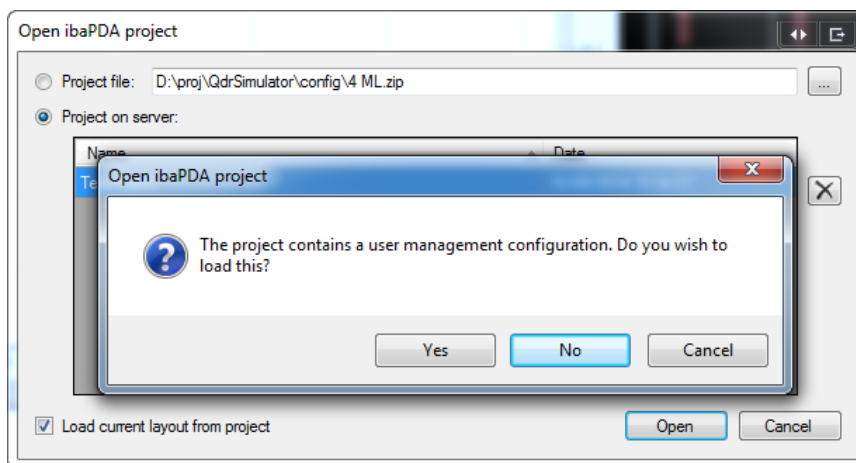
The ibaPDA client displays the user information for both ibaHD and ibaPDA server in the status bar:



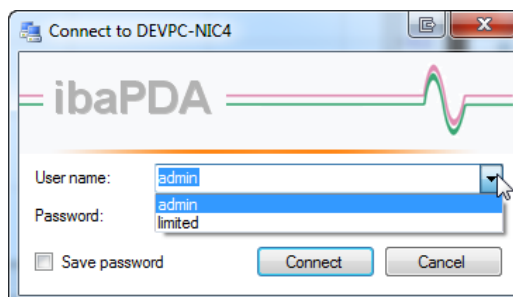
In the user management form 2 buttons have been added to load and save the user configuration. If the selected node is an ibaPDA node then the ibaPDA user configuration will be saved or loaded. If the selected node is an ibaHD node then the ibaHD user configuration will be saved or loaded. The configuration doesn't contain the admin password. This has to be set manually.



The user management configuration of ibaPDA can also be saved in the ibaPDA project by selecting the new "Save user management in project" option.



When you open a project which contains a user management configuration then ibaPDA will ask if you want to load this configuration or not.

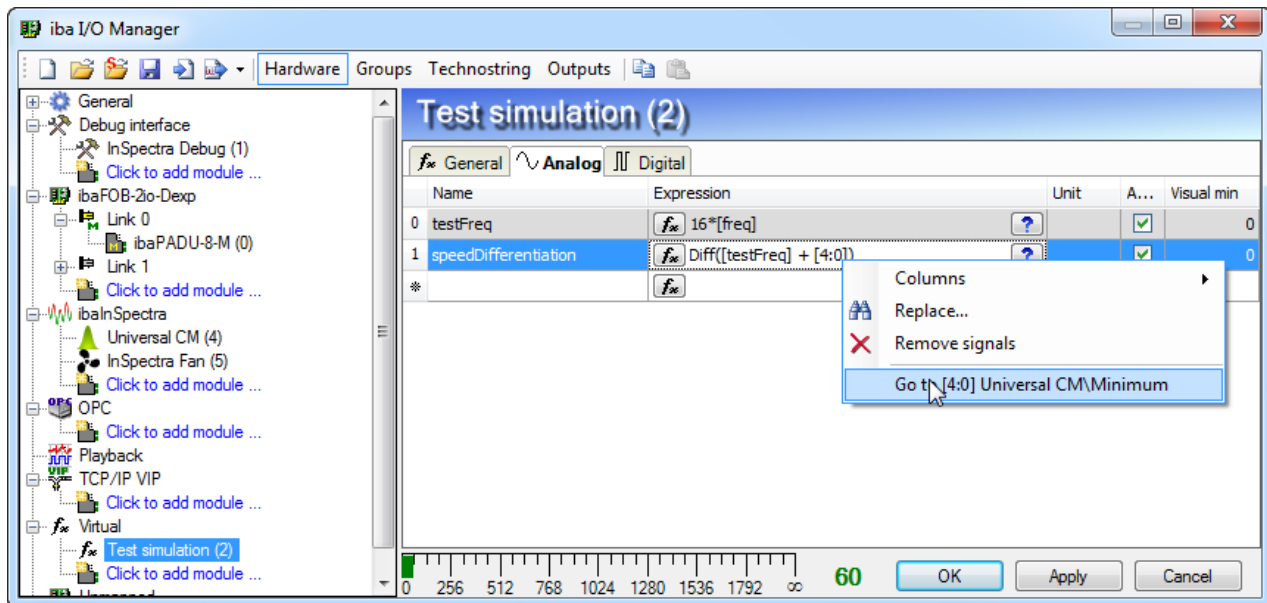


The ibaPDA login form now has a dropdown that shows all available users on the server.

10 Expressions: navigation to signals

Expressions often contain references to other signals. If a user wants to navigate to such signal in the I/O manager tree, he has to locate the module of the signal. In case there are many modules, it can be difficult to find the module. If the signal is referenced by name, it is even more difficult to find the corresponding module.

Therefore, a context menu was implemented in the analog and digital TAB of the virtual module. This works if you right click a signal in the cell of the grid, while the expression is not being edited.



If you click to go to the signal, the system will

- 1) automatically navigate to the correct module
- 2) go to the correct grid (analog or digital TAB)
- 3) select the correct signal

This functionality does not work in the expression editor, nor in the InSpectra profile dialog.

11 InSpectra

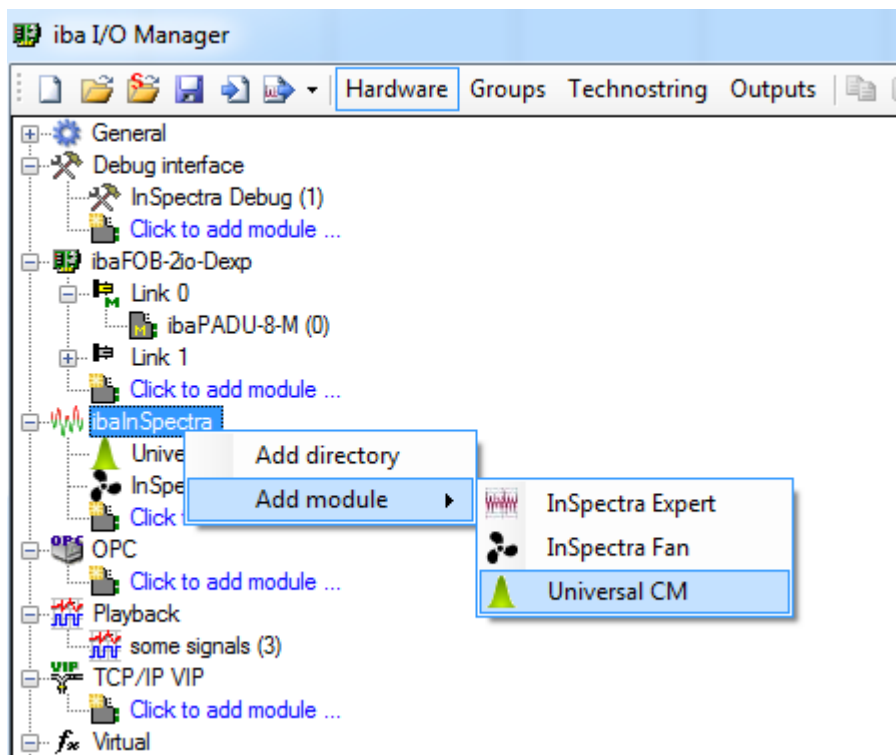
11.1 Universal CM module

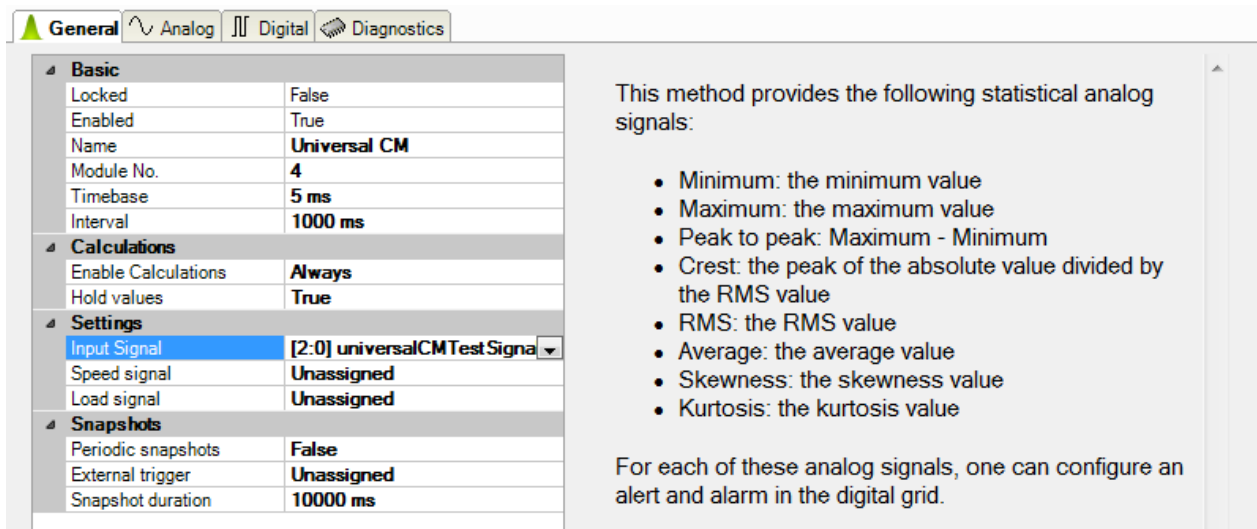
This new module allows studying some stochastic parameters of one input signal:

- Minimum
- Maximum
- Average
- Peak to peak
- RMS
- Crest
- Skewness
- Kurtosis

The module supports online monitoring and offline monitoring (snapshots).

11.1.1 Settings and signals





General | Analog | Digital | Diagnostics

Basic

Locked	False
Enabled	True
Name	Universal CM
Module No.	4
Timebase	5 ms
Interval	1000 ms

Calculations

Enable Calculations	Always
Hold values	True

Settings

Input Signal	[2:0] universalCMTest Signal
Speed signal	Unassigned
Load signal	Unassigned

Snapshots

Periodic snapshots	False
External trigger	Unassigned
Snapshot duration	10000 ms

This method provides the following statistical analog signals:

- Minimum: the minimum value
- Maximum: the maximum value
- Peak to peak: Maximum - Minimum
- Crest: the peak of the absolute value divided by the RMS value
- RMS: the RMS value
- Average: the average value
- Skewness: the skewness value
- Kurtosis: the kurtosis value

For each of these analog signals, one can configure an alert and alarm in the digital grid.

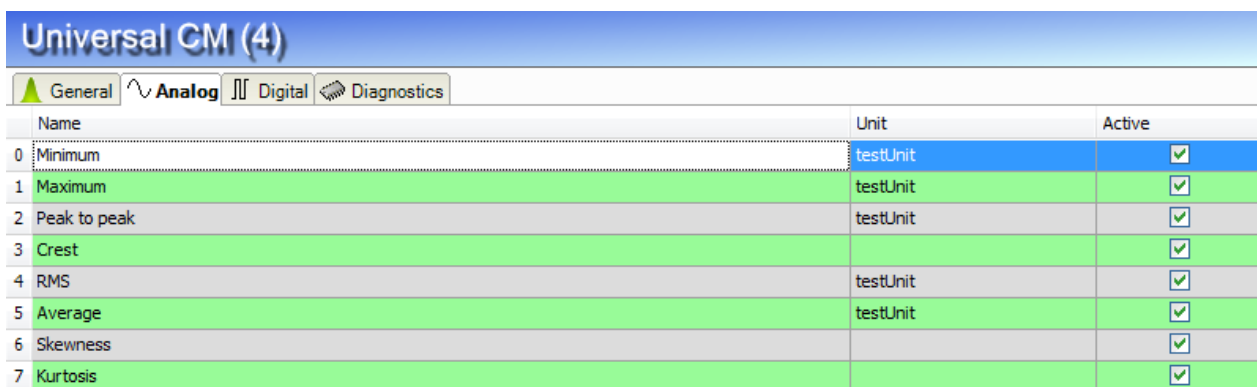
The module recalculates all these stochastic values every timebase with a moving window which has the length of the “interval” setting. The interval setting has to be a multiple of the timebase, but it cannot be higher than 10000 times the timebase.

The “Input Signal” setting corresponds to the monitored signal.

The speed and load signal are optional. Their only purpose is to save them in snapshots.

As with the InSpectra expert module, the user can choose when these values have to be calculated via the “Enable Calculations” setting. The “Hold values” setting determines whether the module signals fall back to zero if the calculations get disabled.

The snapshot features are similar as in the other InSpectra modules. The snapshot duration allows choosing a snapshot interval length different from the online interval length.

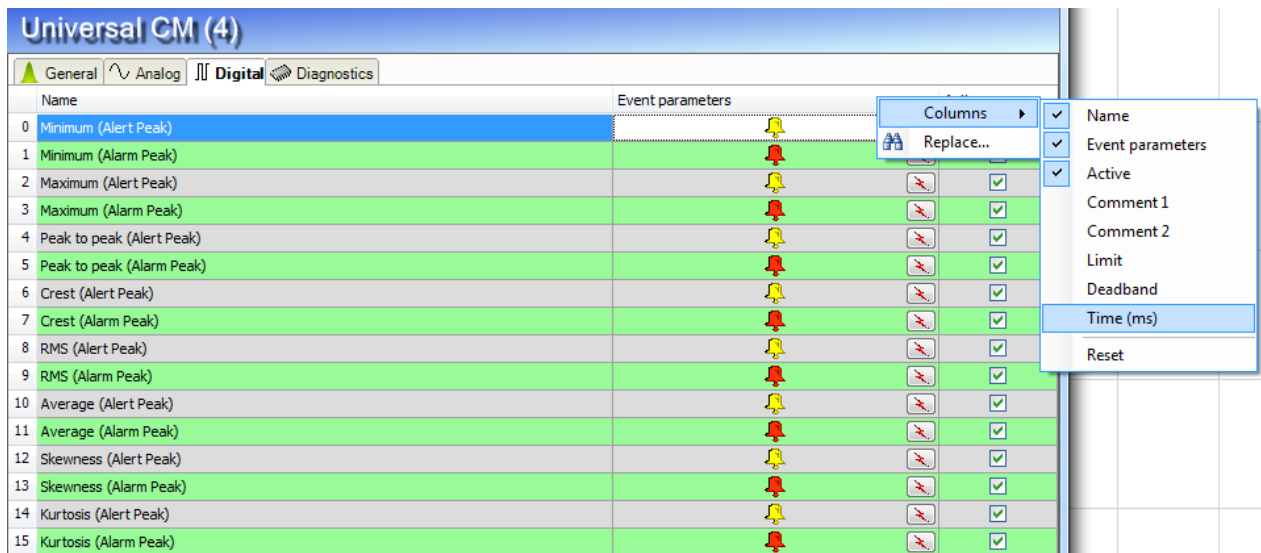


Universal CM (4)

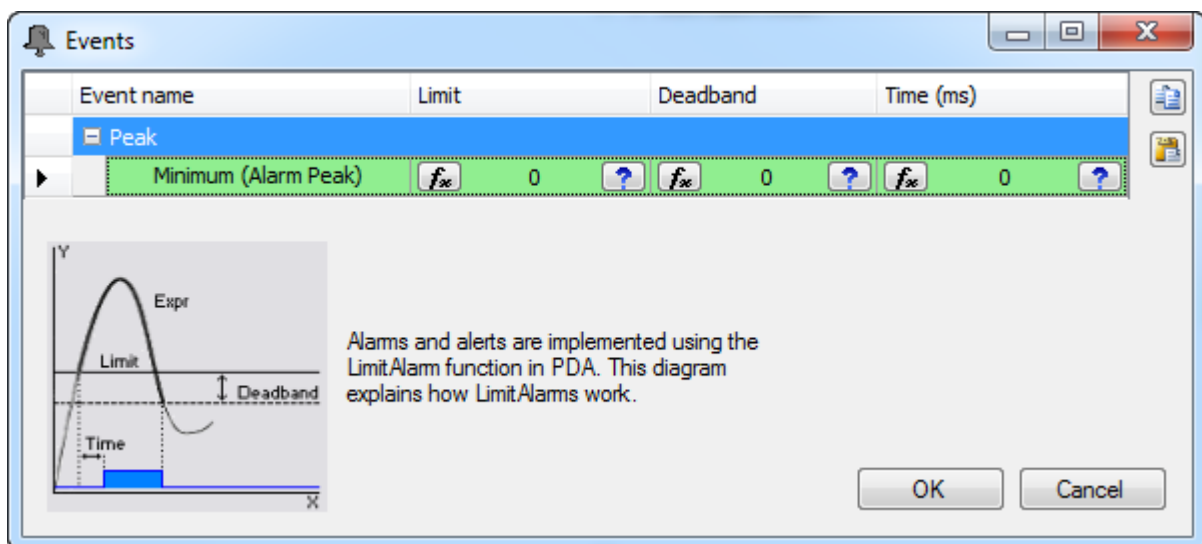
General | **Analog** | Digital | Diagnostics

Name	Unit	Active
0 Minimum	testUnit	<input checked="" type="checkbox"/>
1 Maximum	testUnit	<input checked="" type="checkbox"/>
2 Peak to peak	testUnit	<input checked="" type="checkbox"/>
3 Crest		<input checked="" type="checkbox"/>
4 RMS	testUnit	<input checked="" type="checkbox"/>
5 Average	testUnit	<input checked="" type="checkbox"/>
6 Skewness		<input checked="" type="checkbox"/>
7 Kurtosis		<input checked="" type="checkbox"/>

The units and the signal names are determined automatically. The unit is the unit of the input signal.



In the digital grid you can set alerts or alarms on the analog signals. You can define events by clicking on the button in the “Event parameters” column.



Alternatively, via the context menu, you can display 3 additional columns named “Limit”, “Deadband” and “Time (ms)” to enter the corresponding expressions immediately in the digital grid.

In the diagnostics TAB, you can see the current values if the acquisition is running.

11.1.2 Snapshots

A Universal CM snapshot contains all analog signals during the snapshot duration. There is an offset between the start of these signals and the start of the dat file. This offset is equal to the interval length.

Additionally, it contains the data of the speed and load signal during the snapshot duration. The average of these signals is saved as an infofield.

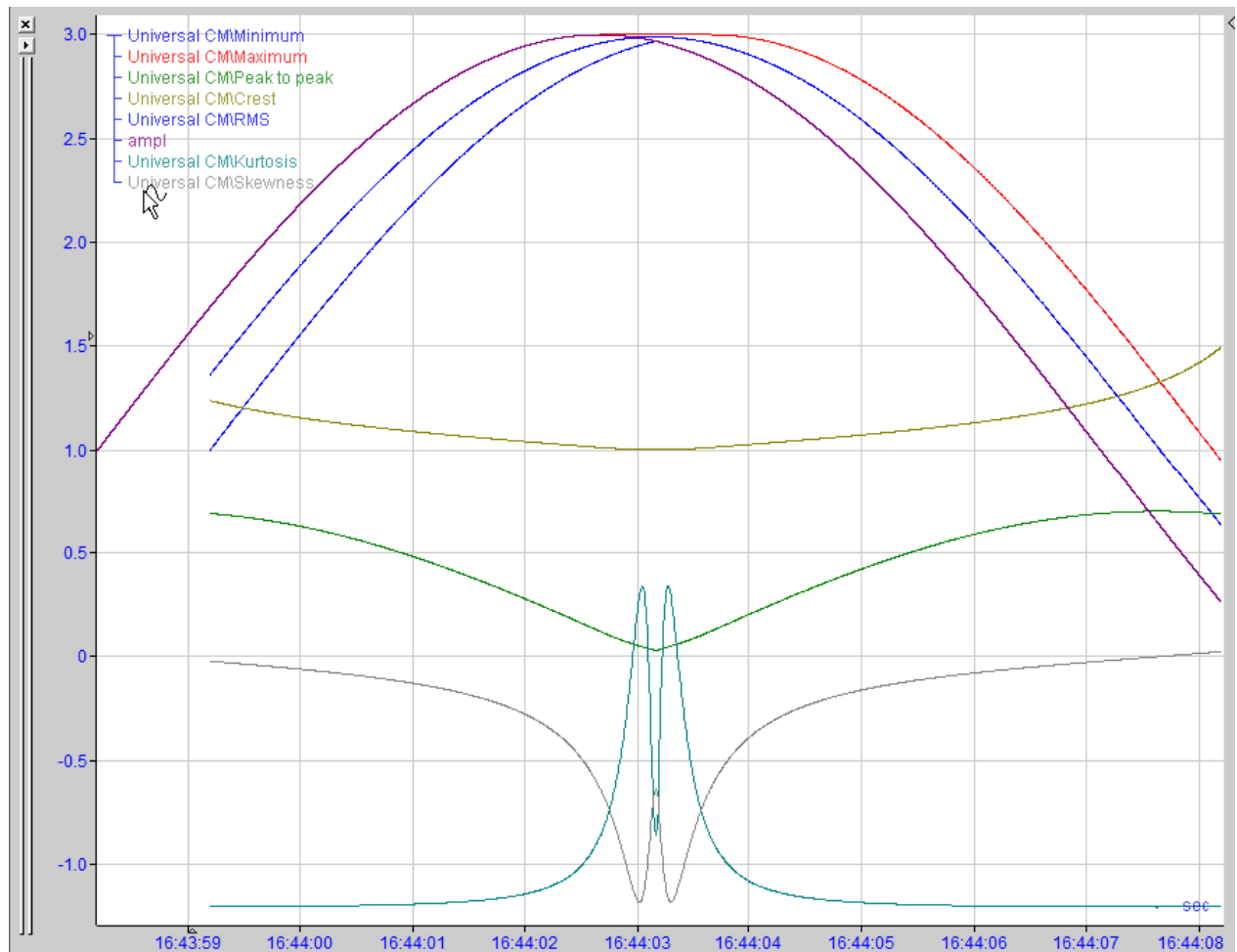
To be able to distinguish between the speed and the load signal, two infofields are available, specifying the id of the speed and the load signal.

Signals

D:\datanaps\DEVPC-IGOR_InSpectra_4_Universal CM_Time_2014-09-30_16.43.58.dat

- info
 - clk: 0.000125
 - typ: real
 - frames: 0000080001
 - starttime: 30.09.2014 16:43:58.197334
 - SPDA_RefTimestamp: 00063547692237190000
 - \$INSPECTRA_Module_name: Universal CM**
 - version: ibaPDA 6.33.0 BETA45
 - beginmodule: 00000000004
 - \$INSPECTRA_InputSignal: 2:2**
 - \$INSPECTRA_SpeedSignal: 2:1**
 - endmodule.
- 2. Test simulation
 - 2:1: freq
 - beginchannel: 129
 - name: freq
 - unit:
 - SPDA_Tbase: 0.000125
 - \$INSPECTRA_Avg: 100.303958768089**
 - xoffset: 000000000000
 - channel_offset: 0182A67CABEBE1884
 - 2:2: ampl
 - beginchannel: 130
 - name: ampl
 - unit:
 - SPDA_Tbase: 0.000125
 - xoffset: 000000000000
 - channel_offset: 00003486129290023
- 4. Universal CM
 - 4:0: Universal CMMinimum
 - 4:1: Universal CMMaximum
 - 4:2: Universal CMPeak to peak
 - 4:3: Universal CM\Crest
 - 4:4: Universal CM\RMS
 - 4:5: Universal CMAverage
 - 4:6: Universal CM\Skewness
 - 4:7: Universal CM\Kurtosis

Signals Search Report info Analysis files

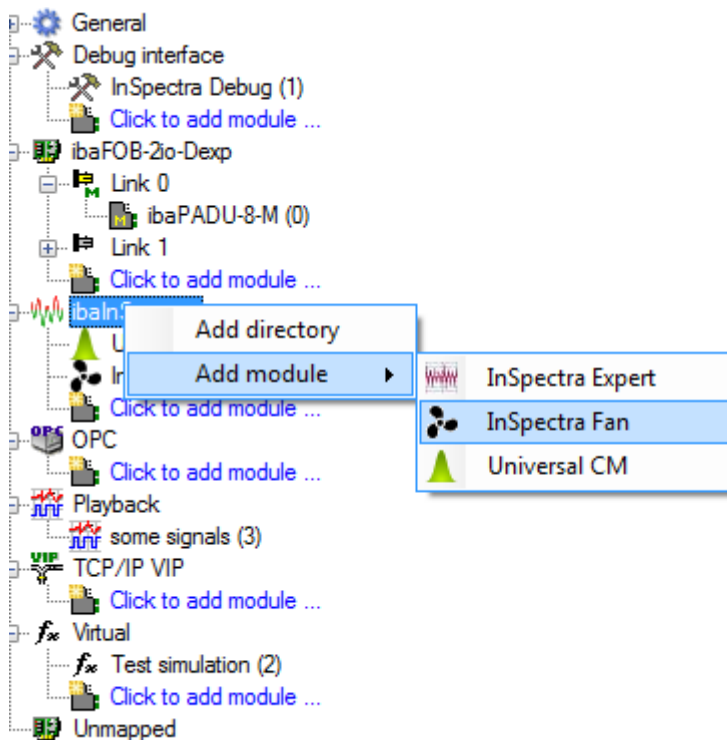


11.2 Fan module

The fan module allows studying 4 features: imbalance, blade, flow turbulence and blade rubbing.

The module supports online monitoring and offline monitoring (snapshots).

11.2.1 Settings and signals



The fan module allows studying 4 features:

- Imbalance: indicator for imbalance
- Blade: indicator for structural blade problem
- Flow turbulence: indicator for blade rubbing
- Blade rubbing: indicator for flow turbulence

General
Analog
Digital
Diagnostics

Basic

Locked	False
Enabled	True
Name	InSpectra Fan
Module No.	5
Timebase	1 ms
Interval	500 ms

Calculations

Enable Calculations	Always
Hold values	False

Fan

Number of blades	3
Maximum speed (RPM)	1000

Settings

Input Signal	[2:3] inspectraFan Test Signal
Speed (RPM)	[2:4] testFreq
Load signal	Unassigned

Snapshots

Periodic snapshots	False
External trigger	Unassigned

Name

The name of the module.

This method provides the following analog signals:

- Imbalance: indicator for imbalance
- Blade: indicator for structural blade problem
- Blade rubbing: indicator for blade rubbing
- Flow turbulence: indicator for flow turbulence

For each of these analog signals, one can configure an alert and an alarm in the digital grid.

The goal is to monitor the trend of these signals. A rising long term trend indicates there is a problem. The module also allows online monitoring by setting alerts or alarms on the short term.

The four analog signals should be considered as features not having any dimension. Therefore, we do not assign a unit to them.

InSpectra Fan (5)		
<div> General Analog Digital Diagnostics </div>		
Name		Active
0 Imbalance		<input checked="" type="checkbox"/>
1 Blade		<input checked="" type="checkbox"/>
2 Flow turbulence		<input checked="" type="checkbox"/>
3 Blade rubbing		<input type="checkbox"/>

In the digital grid you can set alerts or alarms on the analog signals. You can define events by clicking on the button in the “Event parameters” column.

InSpectra Fan (5)		
<div> General Analog Digital Diagnostics </div>		
Name	Event parameters	Active
0 Imbalance (Alert)		<input type="checkbox"/>
1 Imbalance (Alarm)		<input type="checkbox"/>
2 Blade (Alert)		<input type="checkbox"/>
3 Blade (Alarm)		<input type="checkbox"/>
4 Flow turbulence (Alert)		<input type="checkbox"/>
5 Flow turbulence (Alarm)		<input type="checkbox"/>
6 Blade rubbing (Alert)		<input type="checkbox"/>
7 Blade rubbing (Alarm)		<input type="checkbox"/>

11.2.2 Snapshots

The fan module always creates snapshot files in pairs: a time snapshot file and a spectrum snapshot file.

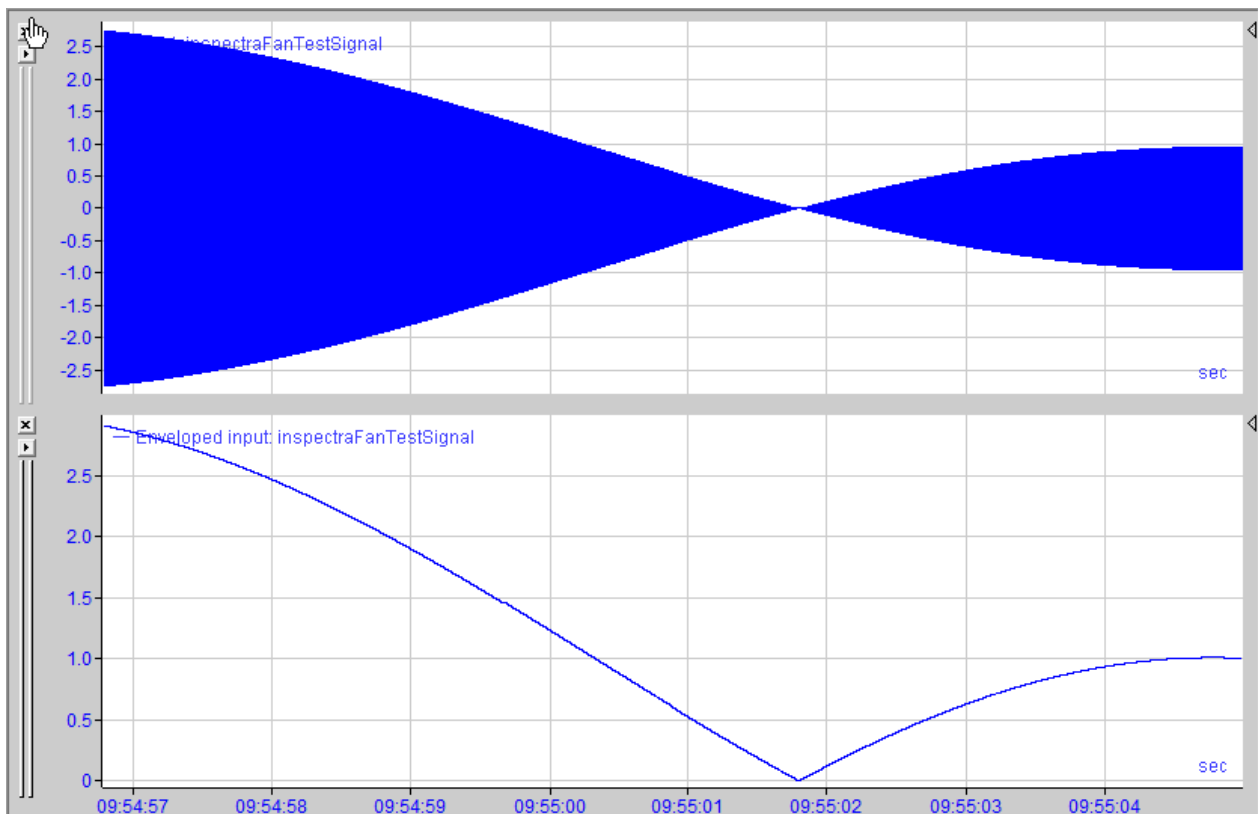
The time snapshot file contains the following signals / infofields:

- Input (time)
- Speed / load signal
- Envelope (time)
- Averaged features (imbalance, blade, ...)

The screenshot displays the ibaPDA software interface with the project tree on the left and a bottom bar with tabs: Signals, Search, Report info, and Analysis files.

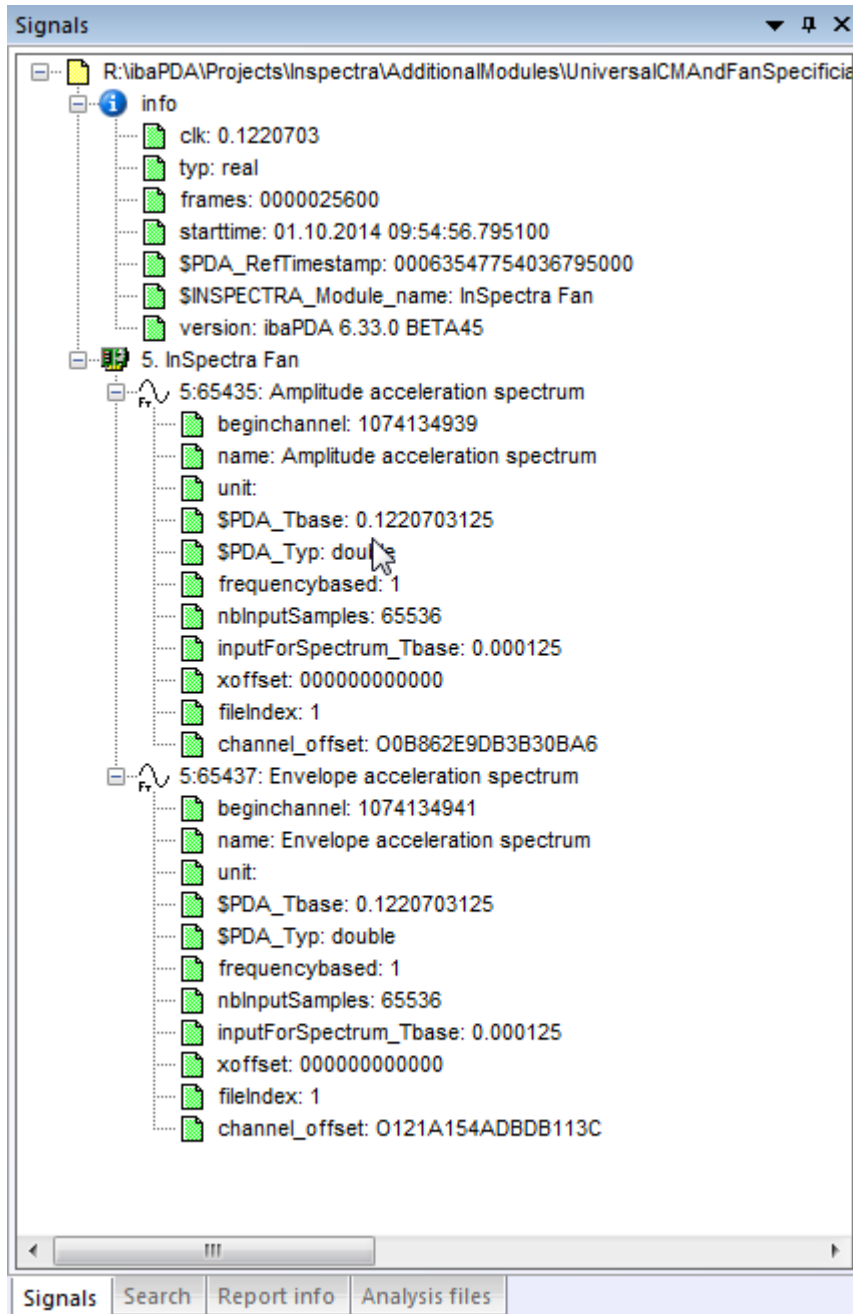
Project Tree:

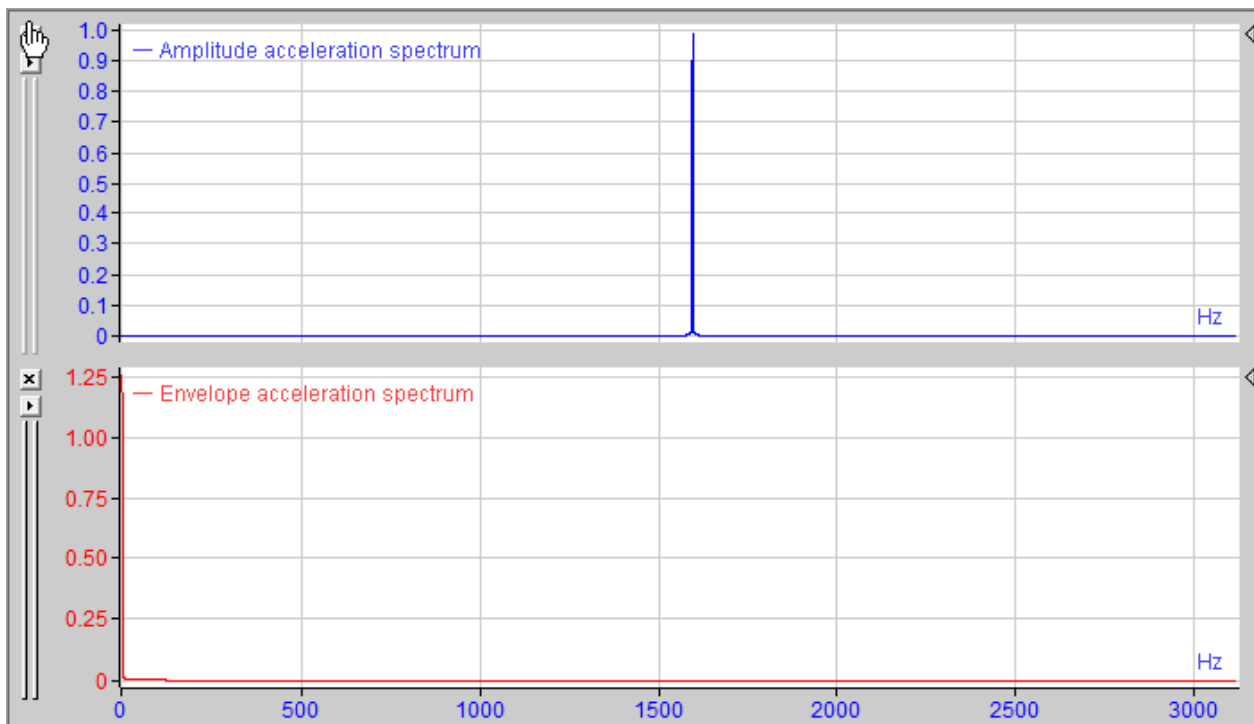
- R:\ibaPDA\Projects\Inspectra\AdditionalModules\UniversalICMAndFanSpecification
 - info
 - clk: 0.000125
 - typ: real
 - frames: 0000065536
 - starttime: 01.10.2014 09:54:56.795100
 - SPDA_RefTimestamp: 00063547754036795000
 - \$INSPECTRA_Module_name: InSpectra Fan
 - version: ibaPDA 6.33.0 BETA45
 - beginmodule: 0000000005
 - \$INSPECTRA_SpeedSignal: 2:4**
 - \$INSPECTRA_Imbalance: 3.18338722899124E-07**
 - \$INSPECTRA_Blade: 5.29271539537259E-05**
 - \$INSPECTRA_FlowTurbulence: 5.27187540207974E-05**
 - \$INSPECTRA_BladeRubbing: 0.00639139241922879**
 - endmodule:
 - 2. Test simulation
 - 2:4: testFreq
 - beginchannel: 132
 - name: testFreq
 - unit:
 - SPDA_Tbase: 0.000125
 - \$INSPECTRA_Avg: 1600**
 - xoffset: 000000000000
 - channel_offset: 016D16D24494916F1
 - 5. InSpectra Fan
 - 5:65434: Input: inspectraFanTestSignal
 - 5:65436: Enveloped input: inspectraFanTestSignal



The spectrum snapshot file contains the following signals

- Spectrum of input
- Spectrum of envelope

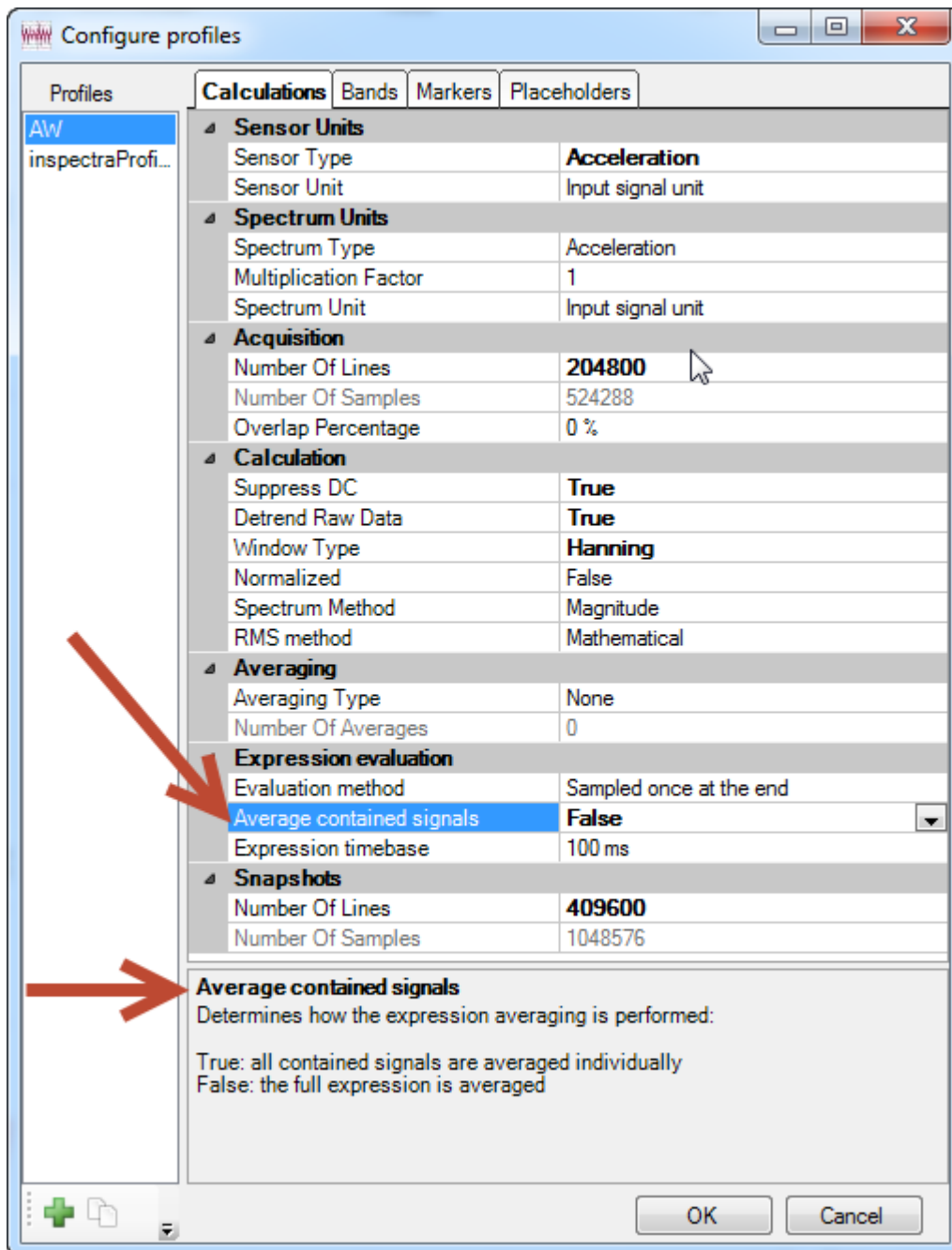




11.3 InSpectra Expert

11.3.1 Averaging

In the InSpectra profiles, one can choose to average the expressions. An option was added to determine how the averaging is done: "Average contained signals". If this setting is off (as in previous versions), the complete expression is averaged. If this setting is on (default from now on), all signals in the expression are averaged before being used in the expression.

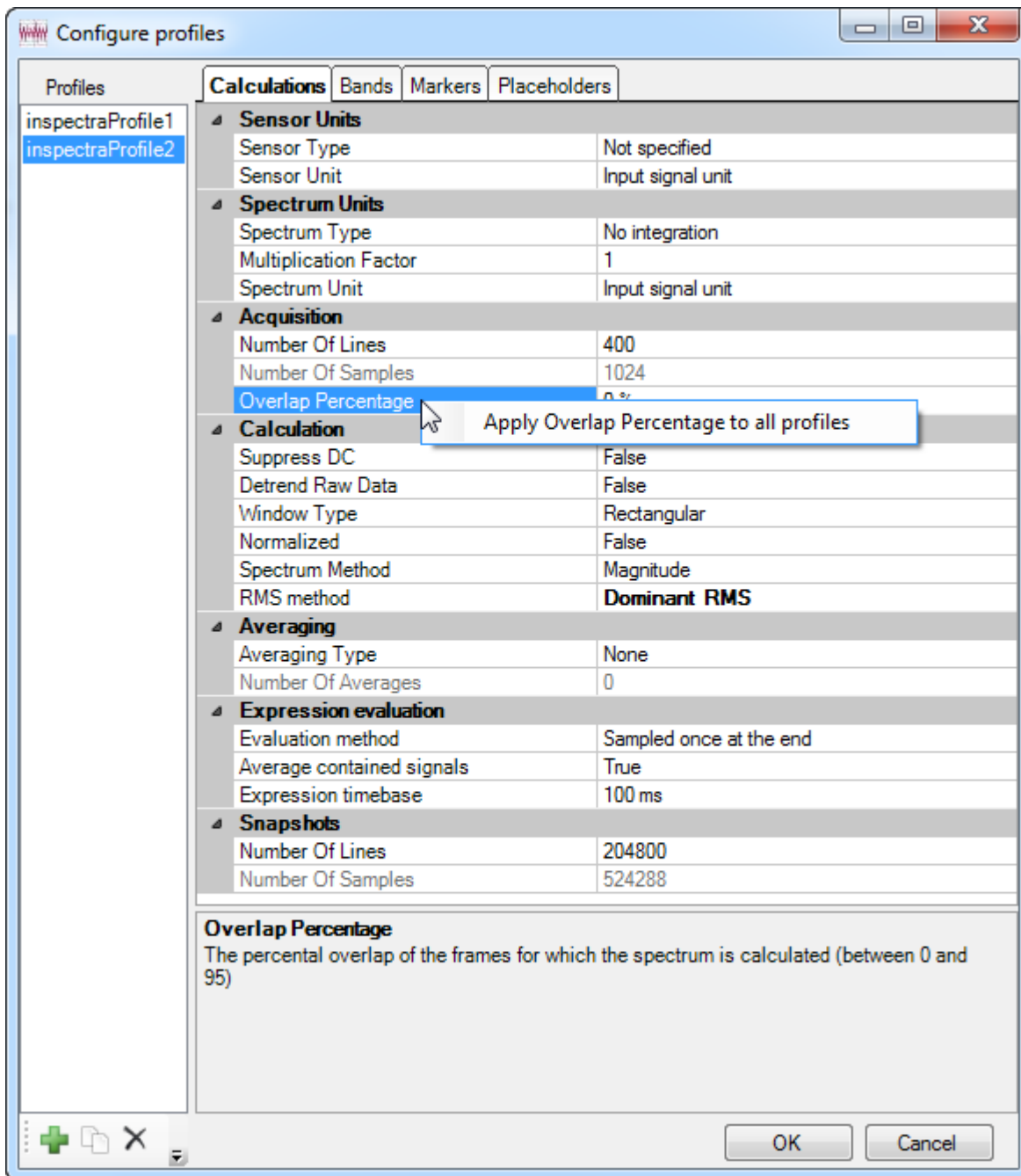


11.3.2 Copy / paste calculation settings

It is possible to copy one specific calculation setting (e.g. *Overlap Percentage*) or a complete group of calculations settings (e.g. *Averaging*) to all other InSpectra profiles. The settings are not copied into memory; they are applied immediately to the other profiles.

This functionality is only available if there is more than one profile.

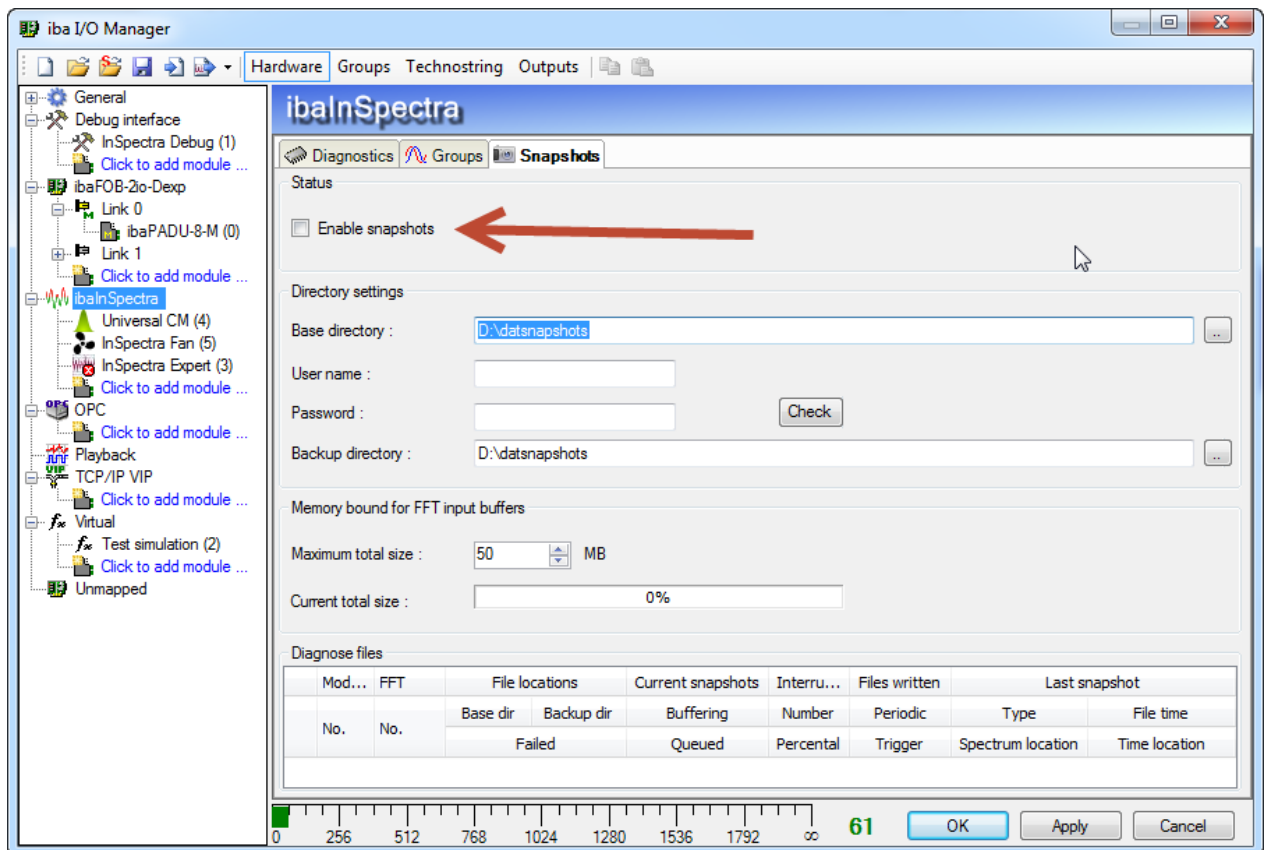
One can do this via the context menu that appears after right clicking the setting or the category:



This does not work for all settings and categories. It is not possible to copy a separate setting with complex dependencies such as *Spectrum type*. The sensor and spectrum unit settings can only be copied together.

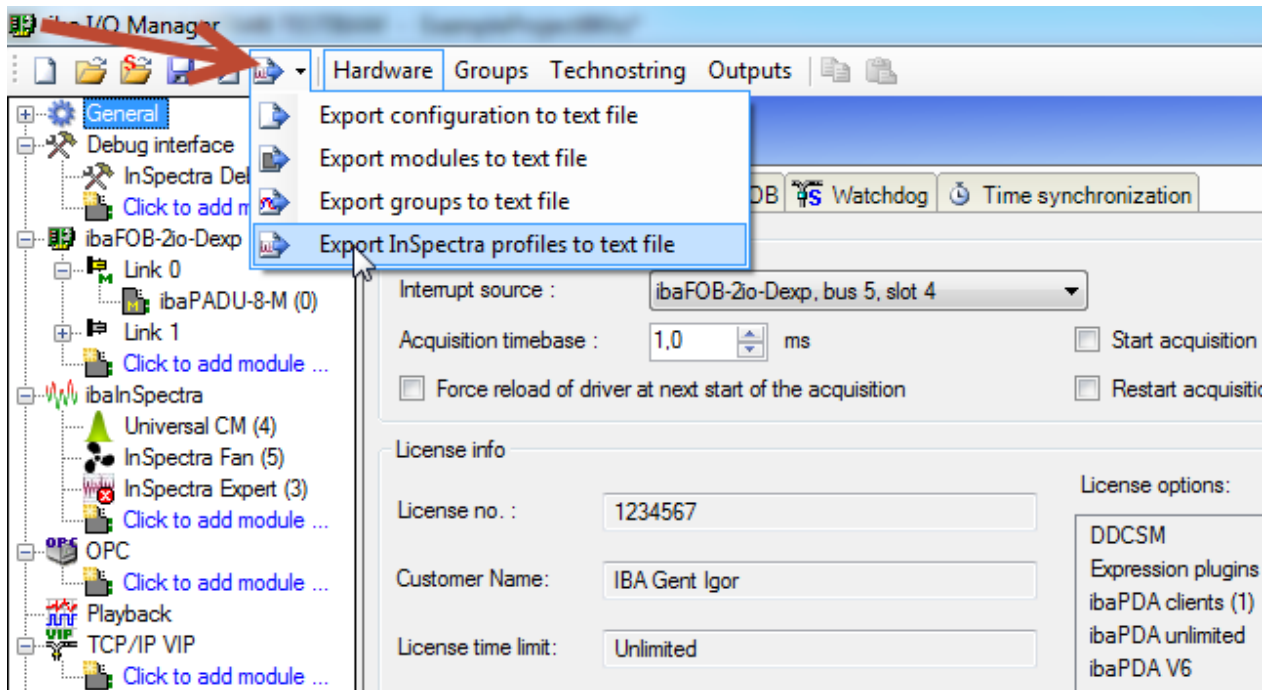
11.4 InSpectra interface: toggle snapshots

One can enable or disable all snapshots with a checkbox in the I/O manager on the InSpectra interface:



12 Improved import / export (text files)

From now on, it is possible to export/import groups and InSpectra profiles to/from text files. The export button in the toolbar of the I/O manager has four options:



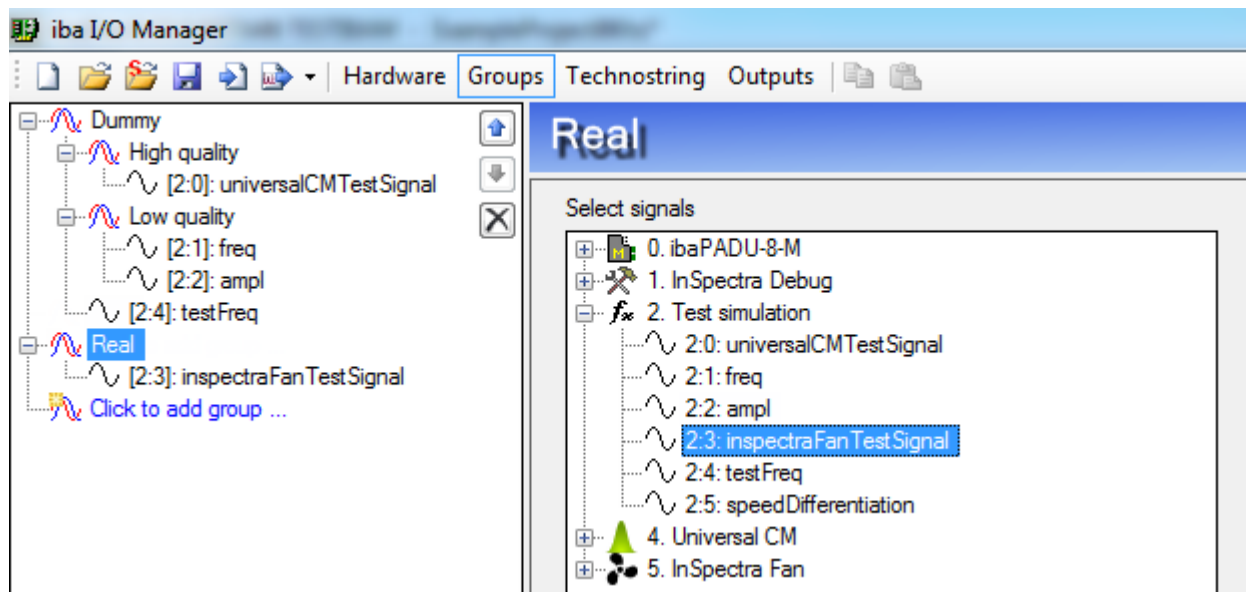
The first option, “Export configuration to text file”, exports all modules, groups and profiles in one text file, in this exact order.

By clicking the import button, which is left of the export button, one can import all modules, groups and profiles in the text file.

12.1 Groups

If exporting all groups to text file, all groups, except the ones that were generated automatically, are saved in the text file. The group information always starts with the keyword “GROUPS”.

Example: below you can find a text export of the following group hierarchy:



Text file:

```
GROUPS
Group SignalId      SignalName
Dummy      [2:4]    testFreq
Dummy\High quality [2:0]    universalCMTTestSignal
Dummy\Low quality  [2:1]    freq
Dummy\Low quality  [2:2]    ampl
Real  [2:3]    inspectraFanTestSignal
```

Each Inie corresponds to one signal. Empty groups (groups not containing any signal directly or indirectly) are not saved.

12.2 InSpectra profiles

12.2.1 Text format

Four keywords are used to save the InSpectra profiles in a text file:

- INSPECTRA\PROFILESETTINGS
 - Calculation settings
 - References to bands, markers and placeholders
 - Each profile setting corresponds to one InSpectra profile
- INSPECTRA\BANDS
- INSPECTRA\MARKERS
- INSPECTRA\PLACEHOLDERS

Each keyword is followed by a header, specifying the columns. The following columns are possible. The columns in **bold** are obligatory and the underlined columns are special, they can be added manually in the text file if required; they are not created by ibaPda when exporting:

- INSPECTRA\PROFILESETTINGS:

Id;Name;CopyFromProfile;Bands;UsedBands;Markers;UsedMarkers;Placeholders;SensorType;SensorUnit;SpectrumType;MultiplicationFactor;NumberOfLines;OverlapPercentage;SuppressDc;Detrend;WindowType;Normalized;SpectrumMethod;RMSMethod;AveragingType;EvaluationMethod;AverageSignals;ExpressionTimebase;NumberOfLinesSnapshots

- **Id**: this id has to be an integer, e.g. 0 or 1. It has to be unique because it is the key of the profile.
- **CopyFromProfile**: the id or name of another profile in the text file. Allows including the calculation settings of other text file profiles that are not specified in this profile. Recursive references are allowed, but circular references are not. One can use the comma (",") to specify multiple profiles to copy from.
- **Bands/Markers /Placeholders**: the ids or names of the bands, markers and placeholders that are used by this profile. To indicate several, use the following syntax:
 - % (percent): placeholder for any character. Example: if this field is "0_%", then all bands having an id or name starting with "0_" will be considered as part of the profile.
 - , (comma): separator character: Example: if this field is "0_0,0_2", then all bands having an id or name equal to "0_0" or "0_2" will be considered as part of the profile.
 - - (hyphen): Example: if this field is "0-10" then all bands having an integer id between 0 and 10 will be considered as part of the profile. You cannot use this character in combination with the % placeholder.
- **UsedBands/UsedMarkers**: the same as Bands/Markers, but specifying what band and markers are used (this corresponds to the "Used" checkbox in the band table of the InSpectra profiles dialog). If empty, all bands/markers are supposed to be used. If UsedBands is specified, the Bands field can be empty; same for markers.

- INSPECTRA\BANDS

Id;Name;Center;Delta;AlertPeakName;AlertPeakLimit;AlertPeakDeadband;AlertPeakTime;AlertPeakUsed;AlarmPeakName;AlarmPeakLimit;AlarmPeakDeadband;AlarmPeakTime;AlarmPeakUsed;AlertRMSName;AlertRMSLimit;AlertRMSDeadband;AlertRMSTime;AlertRMSUsed;AlarmRMSName;AlarmRMSLimit;AlarmRMSDeadband;AlarmRMSTime;AlarmRMSUsed;BandNumber

- **Id**: a unique string or integer, it is the key of the band
- **BandNumber**: the band number that is preferred by the band. If the band is used in a profile, it is not guaranteed that this band number is used if the profile includes another band with the same band number. In such case, or if this field is empty, the system will choose a band number each time the band is included in a profile.

- INSPECTRA\MARKERS

Id;Name;Center;MarkerNumber

- **Id**: a unique string or integer, it is the key of the marker
- **MarkerNumber**: similar to BandNumber

- INSPECTRA\PLACEHOLDERS

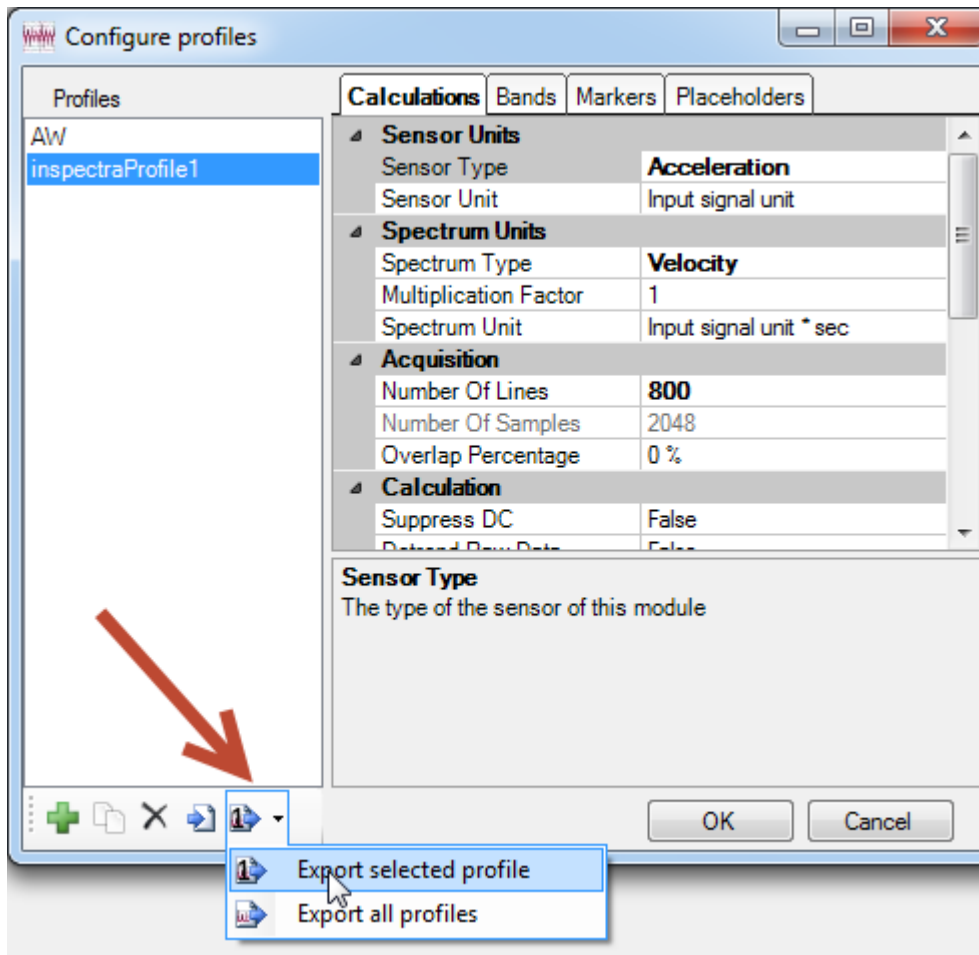
Id;Name;DefaultValue;Comments

- **Id:** a unique string or integer, it is the key of the placeholder

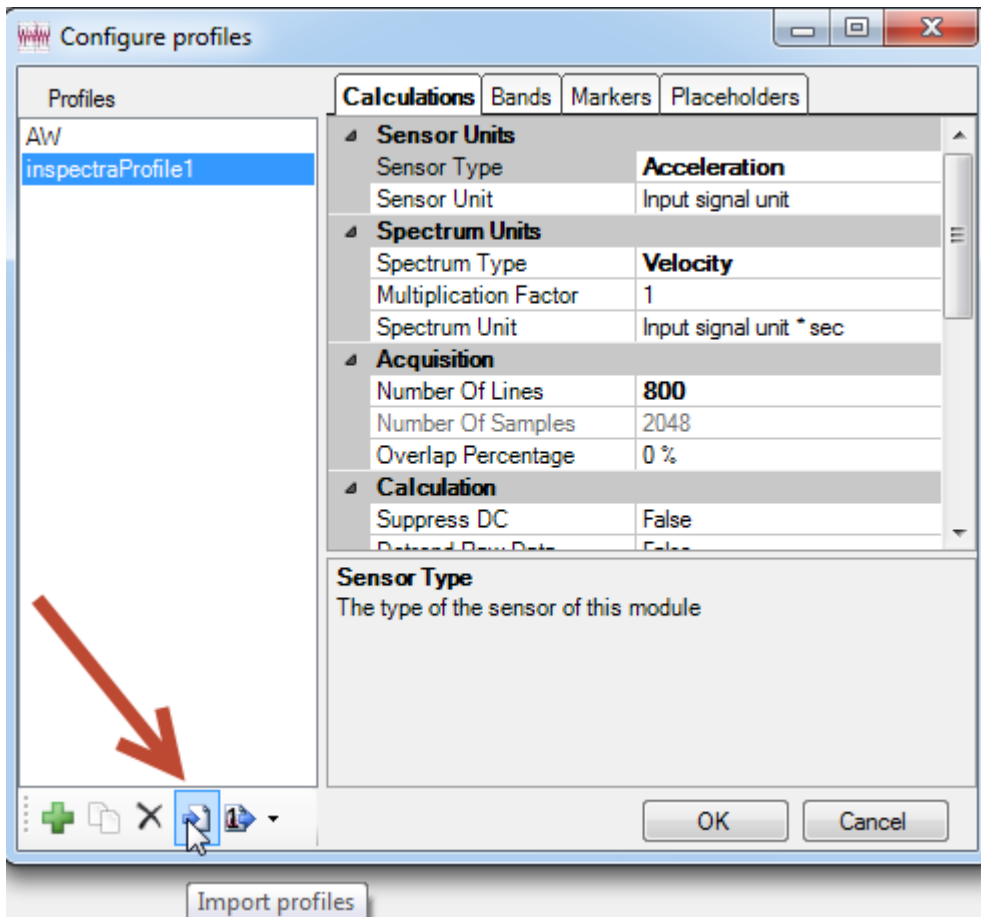
12.2.2 Import / export via InSpectra profiles dialog

The InSpectra profiles dialog allows more customization when exporting or importing InSpectra profiles.

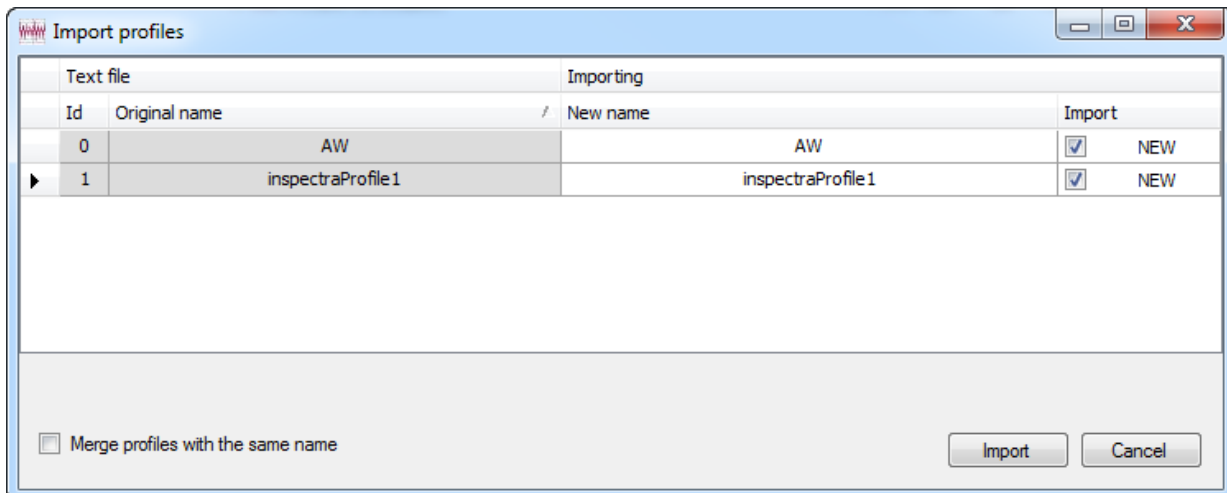
One can export one profile at a time via the toolbar:



Via the import button, one can choose the exact profiles one wishes to import. Suppose you have a file containing 100 profiles, then you can choose the profiles you need. First one has to click the import button:



The following dialog will appear:



In this example, 2 profiles were found in the text file. The "Id" and "Original name" column are copied from the text file. By changing the "New name" column, the user can change the profile name. The user can uncheck the "Import" checkbox, if the profile is not needed.

In case there is already a profile having a name equal to "New name", the system will add a number to the name, it will not override the existing profile. If the checkbox "Merge profiles with the same name" is checked, the system will keep the existing profile and override all settings available in the text file profile.

13 FFT view

13.1 Slave organization

In the trend graph, a slider appears when many subgraphs are displayed. In previous versions, the FFT view had no such slider. In this version, we implemented such slider for the slave windows. This allows adding multiple slaves without decreasing the size of the existing slave windows. As we have added two new slave types, the slider makes it easier to display several slave types at the same time.

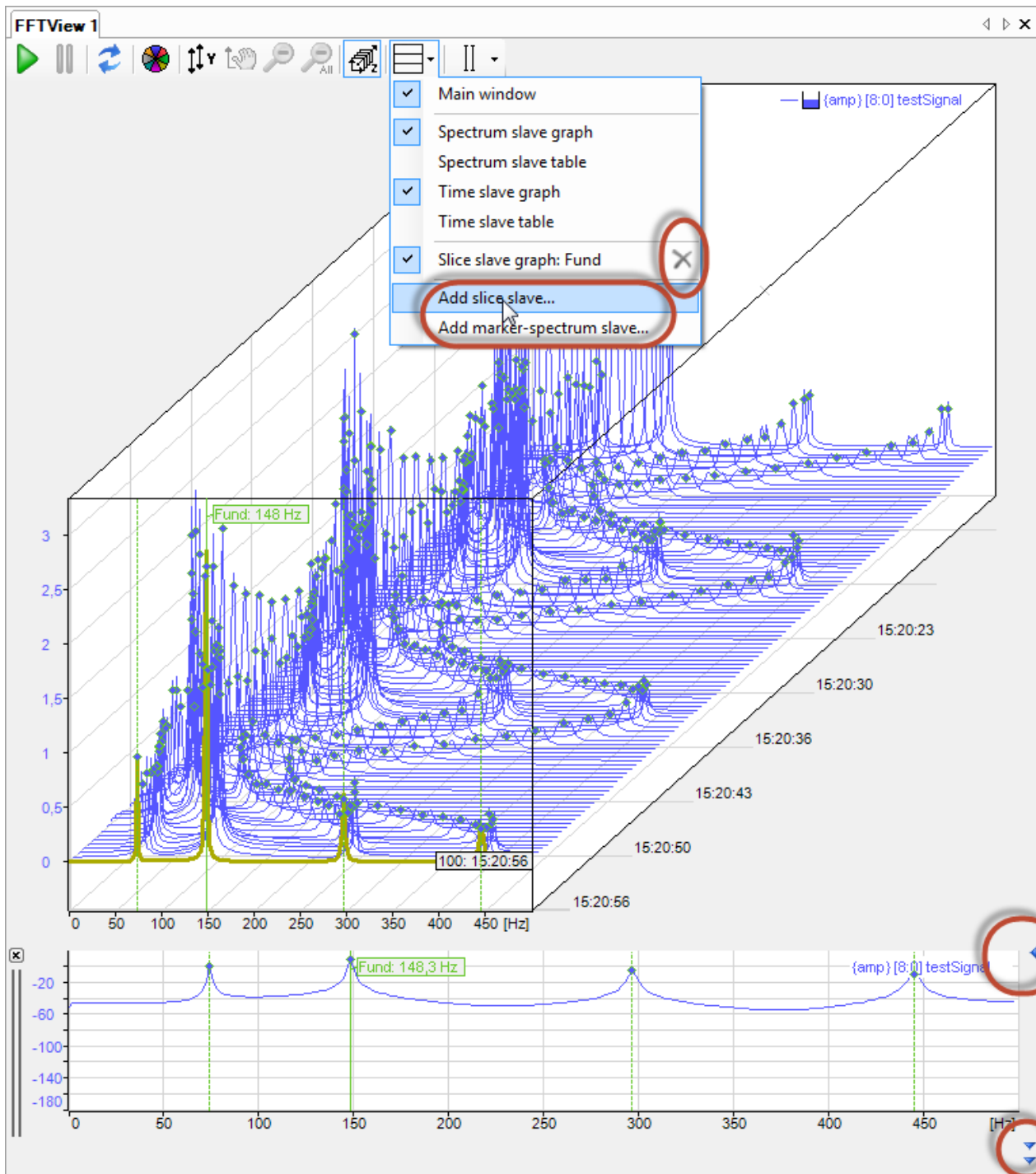
We also introduced the concept of “collapsed” slaves. A slave is indicated with a small triangle on the right side; the content of a collapsed slave is not visible. It works in the same way as in ibaAnalyzer: by clicking the triangle you can hide or show a slave very quickly.

A slave can have three states:

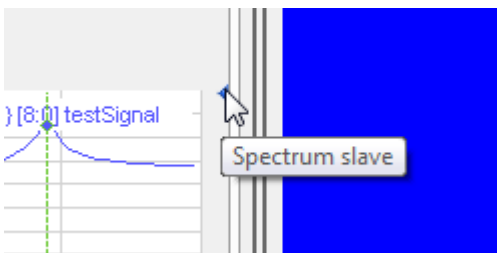
- a) The slave exists: the time and spectrum slave always exist
- b) The slave is shown but collapsed (you only see a triangle)
- c) The slave is shown and not collapsed (you see the slave content)

The spectrum and time slave cannot be deleted; they always exist in the background. It is not possible to have two spectrum or two time slaves, each is present only once. The slave types that are new to this version, the slice slave and the marker-spectrum slave, can be present multiple times in one FFT view. One can add or delete them via the *Slaves and mainwindow* button in the toolbar.

The following screenshot makes the things more clear:



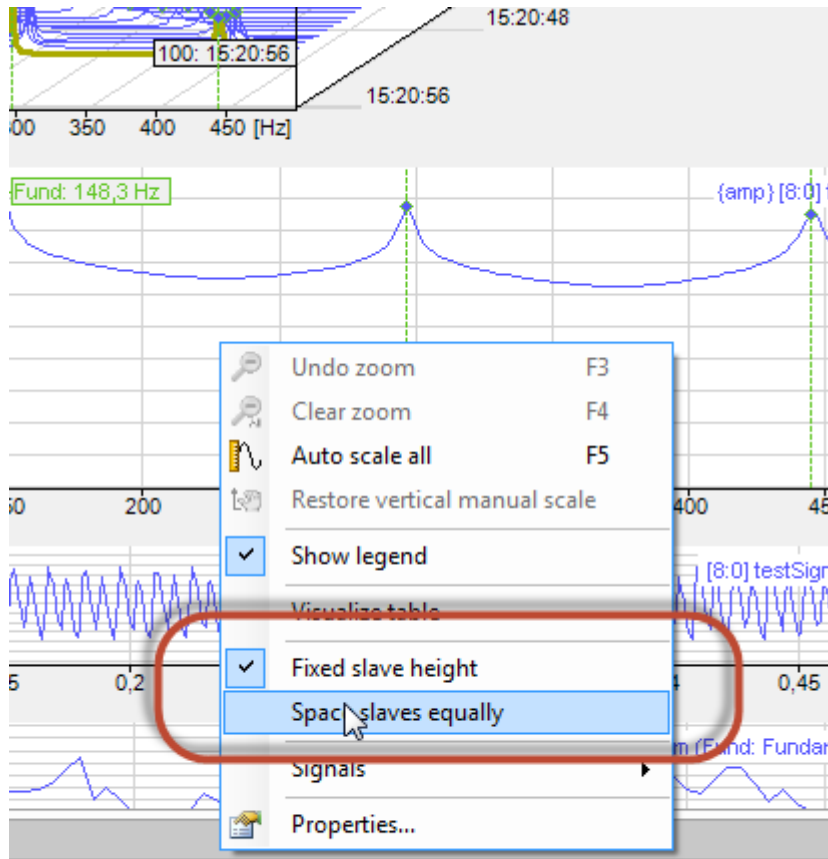
If you hover the blue triangle of a slave, a tooltip appears displaying the slave type:



By default, the FFT view will apportion the available height among the slaves. The FFT view will respect a minimum slave height, and will add a slider to the right of the slave windows if necessary. One can overrule the height of a slave by a drag operation. One can do this by

positioning the mouse in between two slaves: the mouse cursor will change into a splitter. After changing the height, the slave window will have a fixed slave height; it will remember the height in pixels.

Via the context menu one can see whether the slave has a fixed slave height: the *Fixed slave height* setting is checked. To forget about the configured height, one can click the *Space slaves equally* item in the context menu of the slave:



09:50)

Clicking *Space slaves equally* has no influence on the fixed slave height of any other slave.

13.2 New Slaves

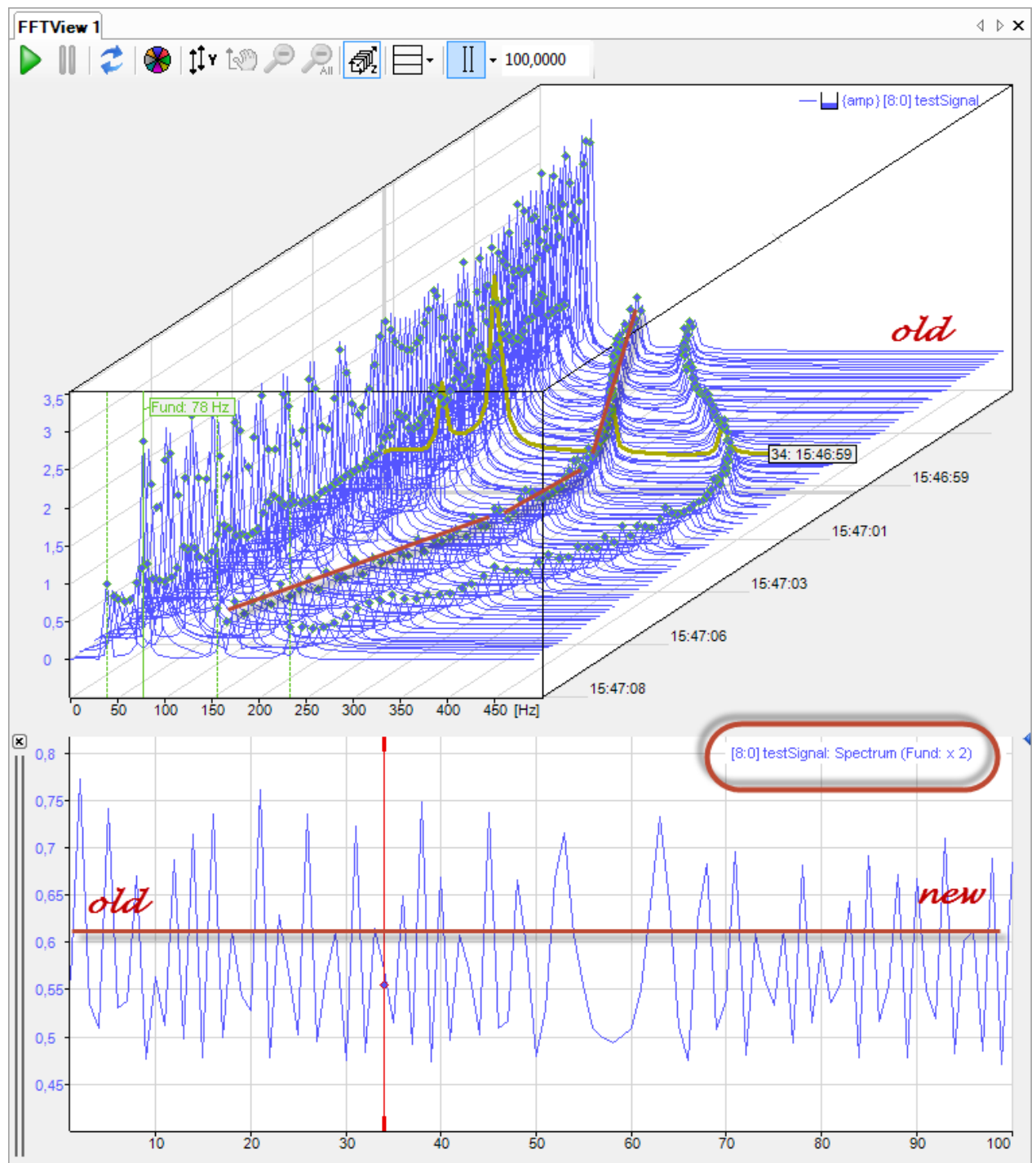
The slice and marker-spectrum slaves are two new slave types. As with the spectrum and time slave, these new slaves support multiple signals at the same time.

13.2.1 Slice slave

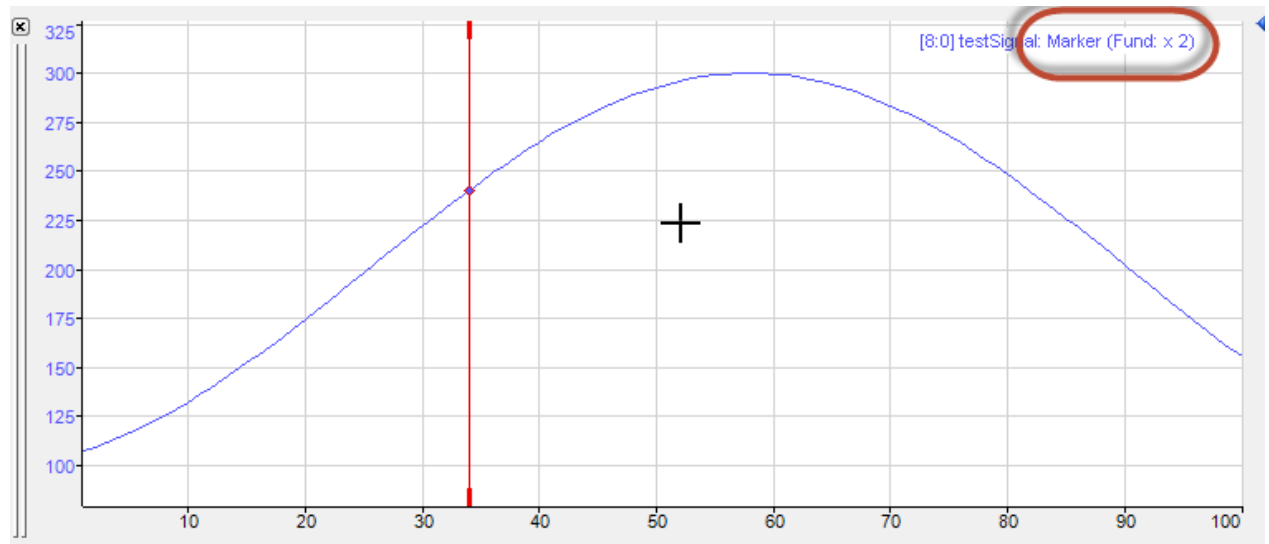
The new slice slave has two modes:

- The “spectrum” mode allows monitoring the spectrum value at a frequency which is moving in time:
 - The time dimension corresponds to the plane number in the waterfall. The plane with the highest plane number corresponds to the most recent data.
 - The frequency dimension is defined by a dynamic marker which is linked to a signal, e.g. a speed signal.

This is demonstrated in the following screenshot:

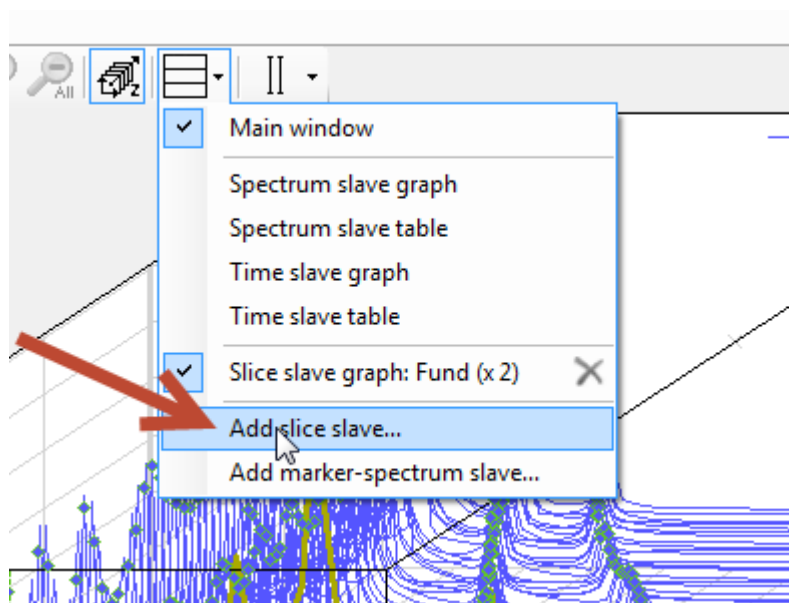


- The Marker mode allows monitoring a frequency which is moving in time. Again, the time dimension corresponds to the plane number in the waterfall. For instance, one can study a speed marker to display the trend of the speed.



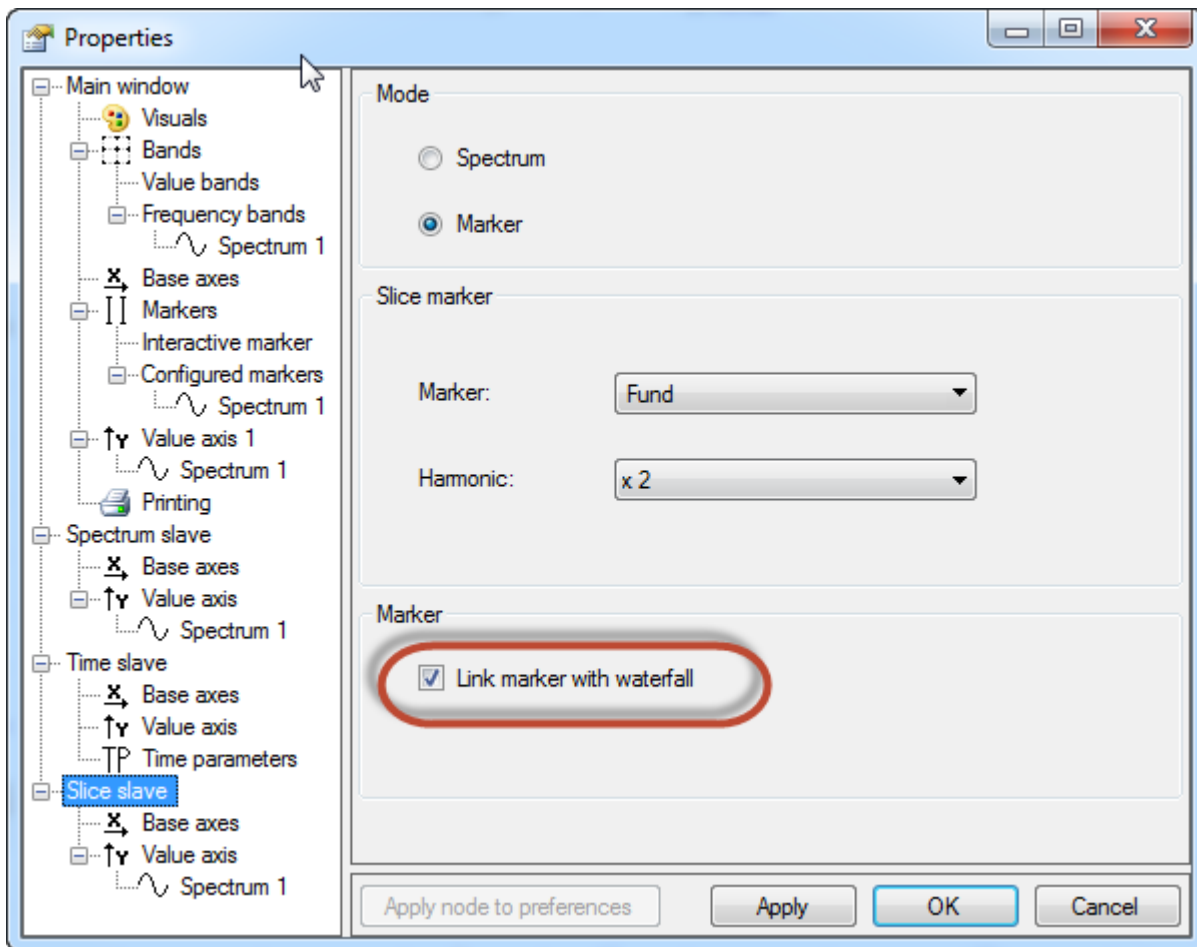
Note that the legend indicates the mode of the slave as well as the marker to which this slave is linked.

A slice slave can be added or deleted via the toolbar:

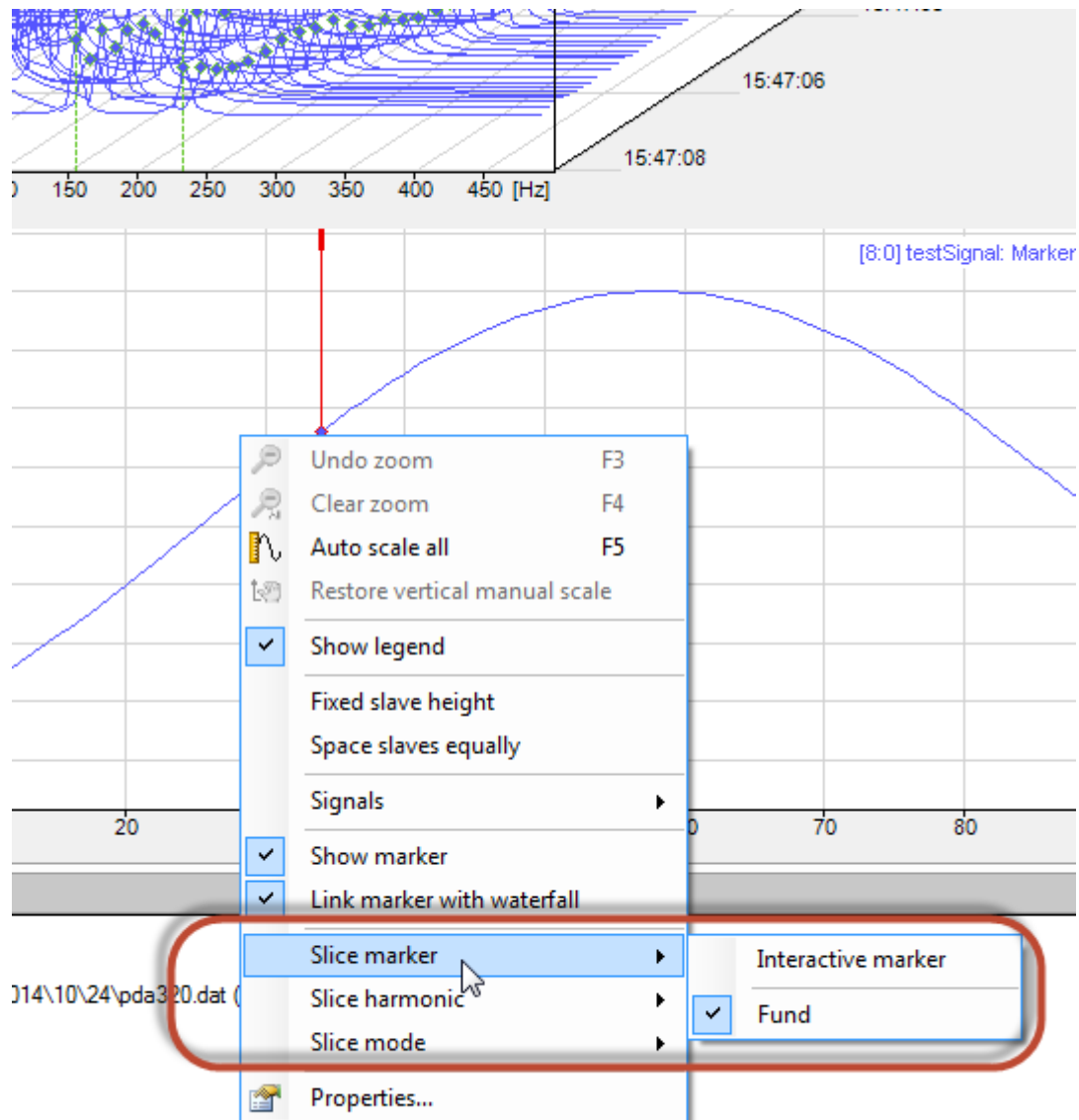


In the property dialog of the slice slave, one can choose the marker that is used by the slice slave. Any marker can be selected, including the available harmonic markers.

The slice slave has its own interactive marker. Via the setting *Link marker with waterfall* in the property dialog of the slice slave, one can link the slave's interactive marker to the plane number that is selected in the waterfall. Remember that the horizontal axis of the slice slave displays the plane numbers.



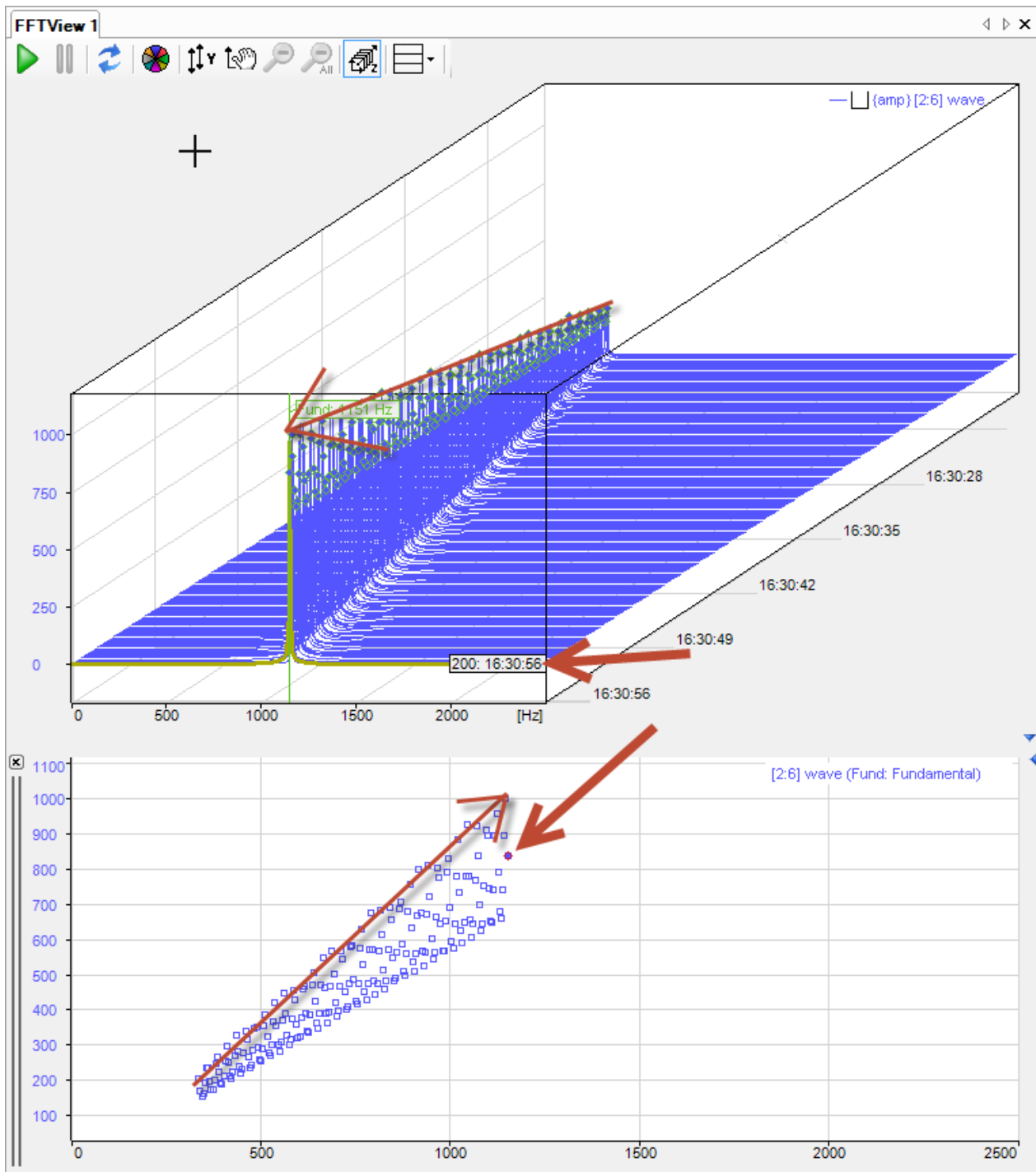
The marker to which the slave is linked can be changed very quickly via the context menu of the slave:



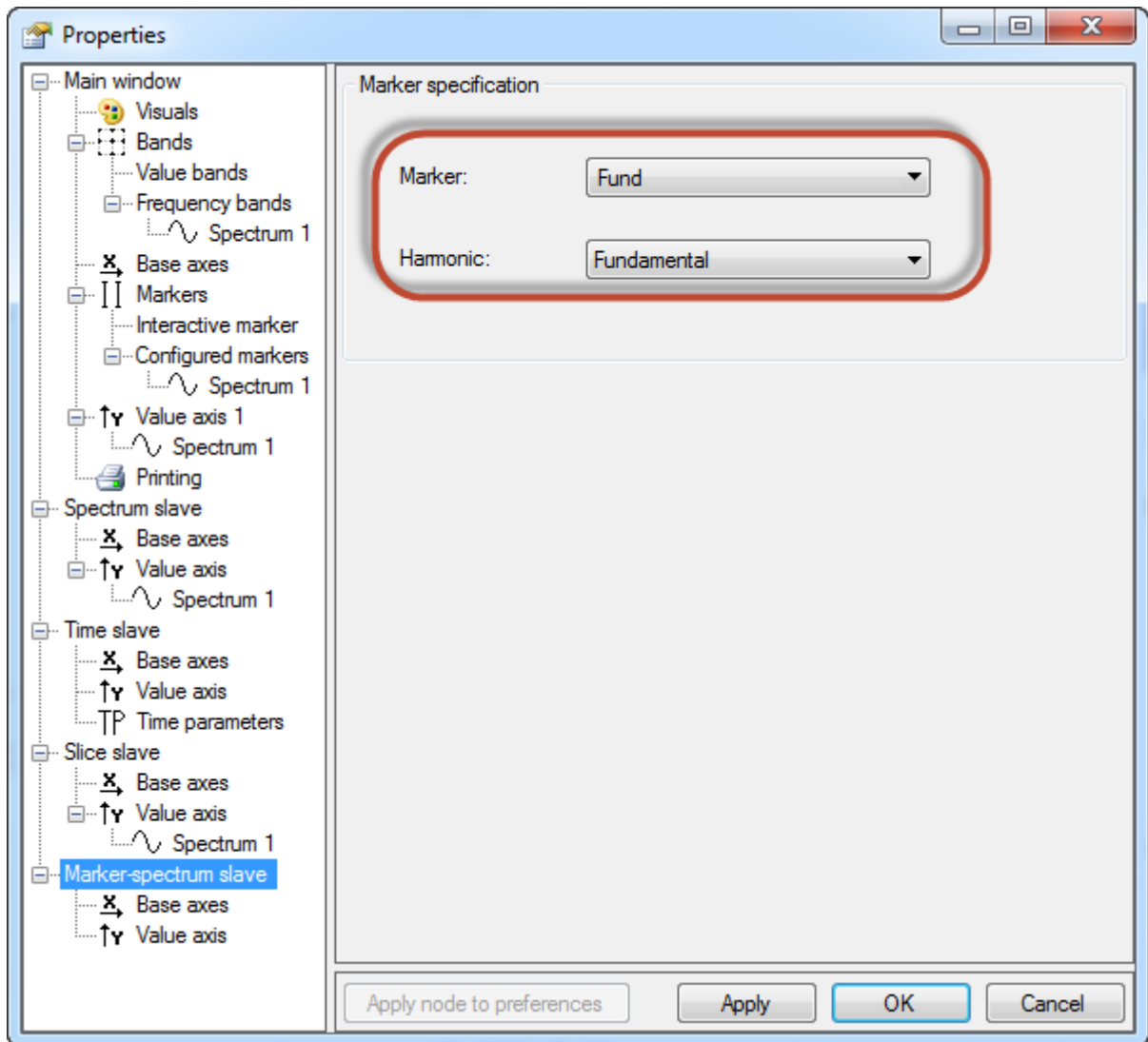
13.2.2 Marker-spectrum slave

The new marker-spectrum allows studying the relation between a dynamic marker (horizontal scale in Hz) and the corresponding spectrum value (vertical scale).

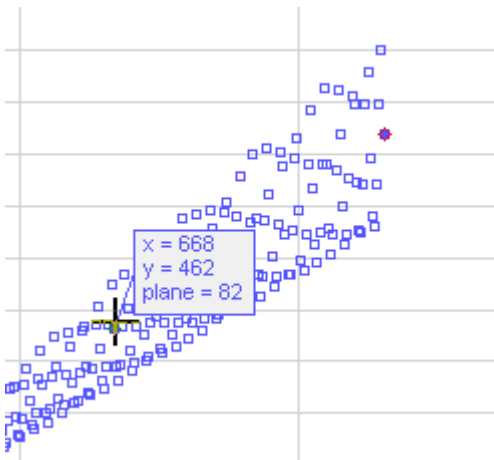
For each plane in the waterfall, this slave displays one dot:



This slave has no interactive marker. The dot corresponding to the selected plane is highlighted. In the property dialog of the slice slave, one can choose the marker that is used by the slave. Any marker can be selected, including all available harmonic markers



If one presses the M key and hovers near a dot, a popup appears with the corresponding marker value, spectrum value and plane number:



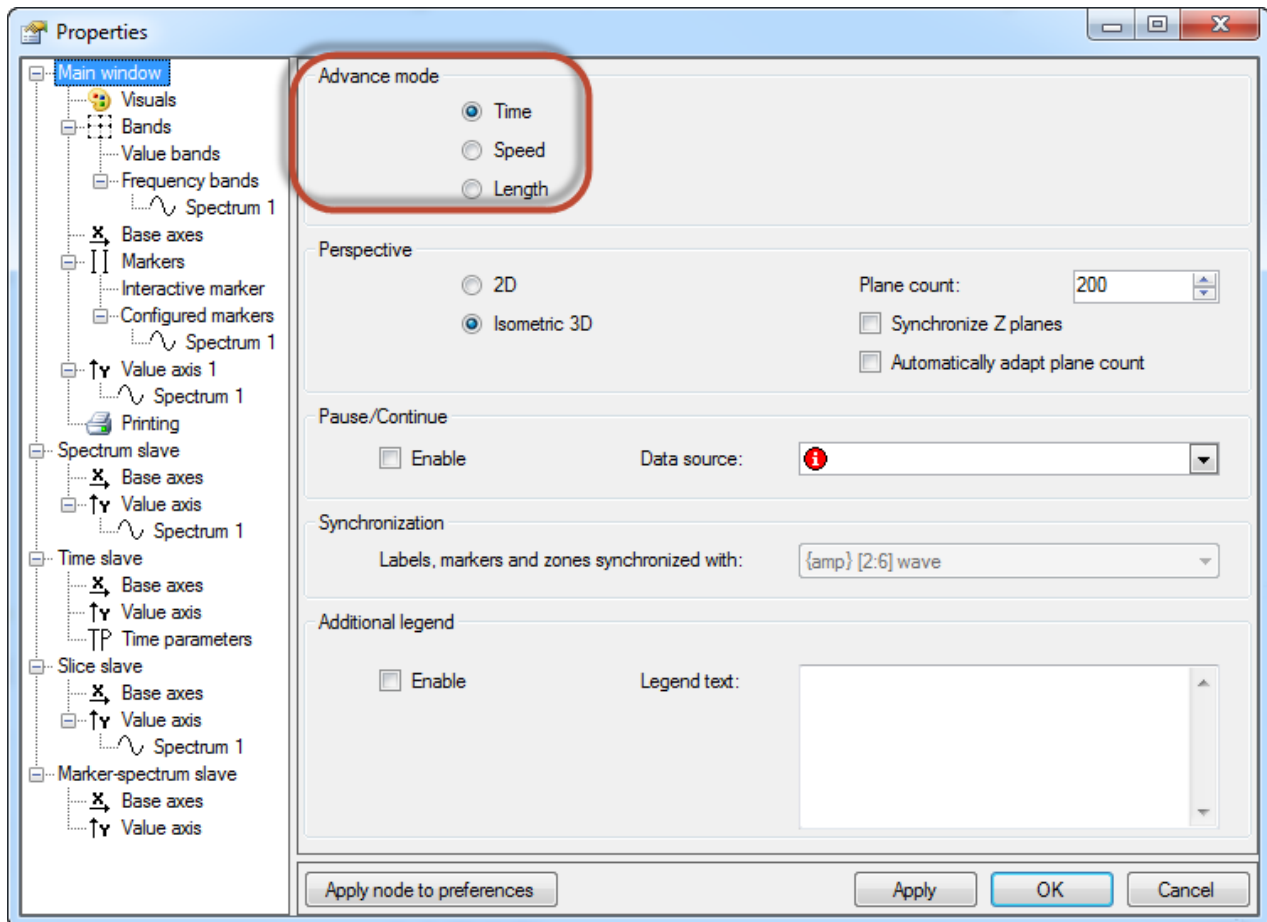
Also here, the marker to which the slave is linked can be changed very quickly via the context meny of the slave.

13.3 FFT Calculations

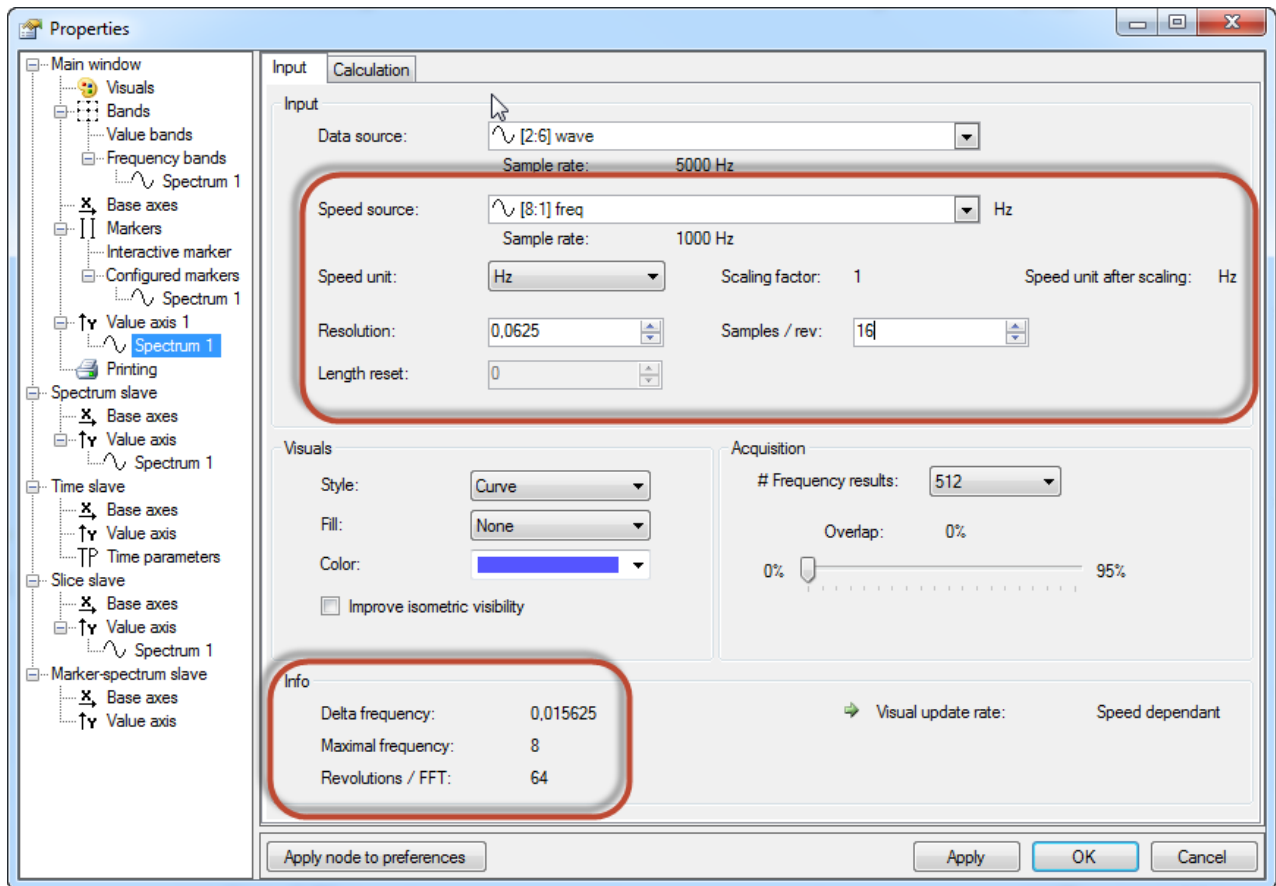
13.3.1 Order calculations

The order calculation features have been extended and the corresponding settings in the property dialog have been rearranged. An order spectrum is like a normal FFT spectrum, but it is normalized against one specific parameter of the process, e.g. the speed at which an object is moving through a production line or the rotational speed of a motor.

The advance mode of the fftview (time, speed or length) must be set on the top node:

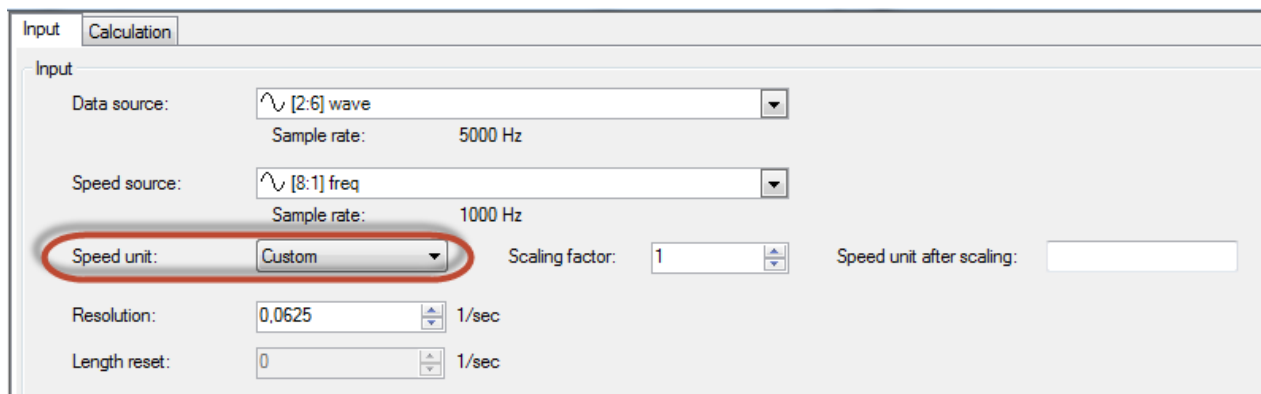


The resolution and length reset settings have been moved to the level of the spectrum node. This makes it easier to configure the order calculation:



It is possible to specify the unit of the speed/length datasource. In case the speed unit is Hz or RPM, the resolution can be set via the *Samples per revolution* setting. The latter value is the inverse of the resolution.

If the speed/length unit is set to *Custom*, one can define a custom scaling factor and a custom unit:



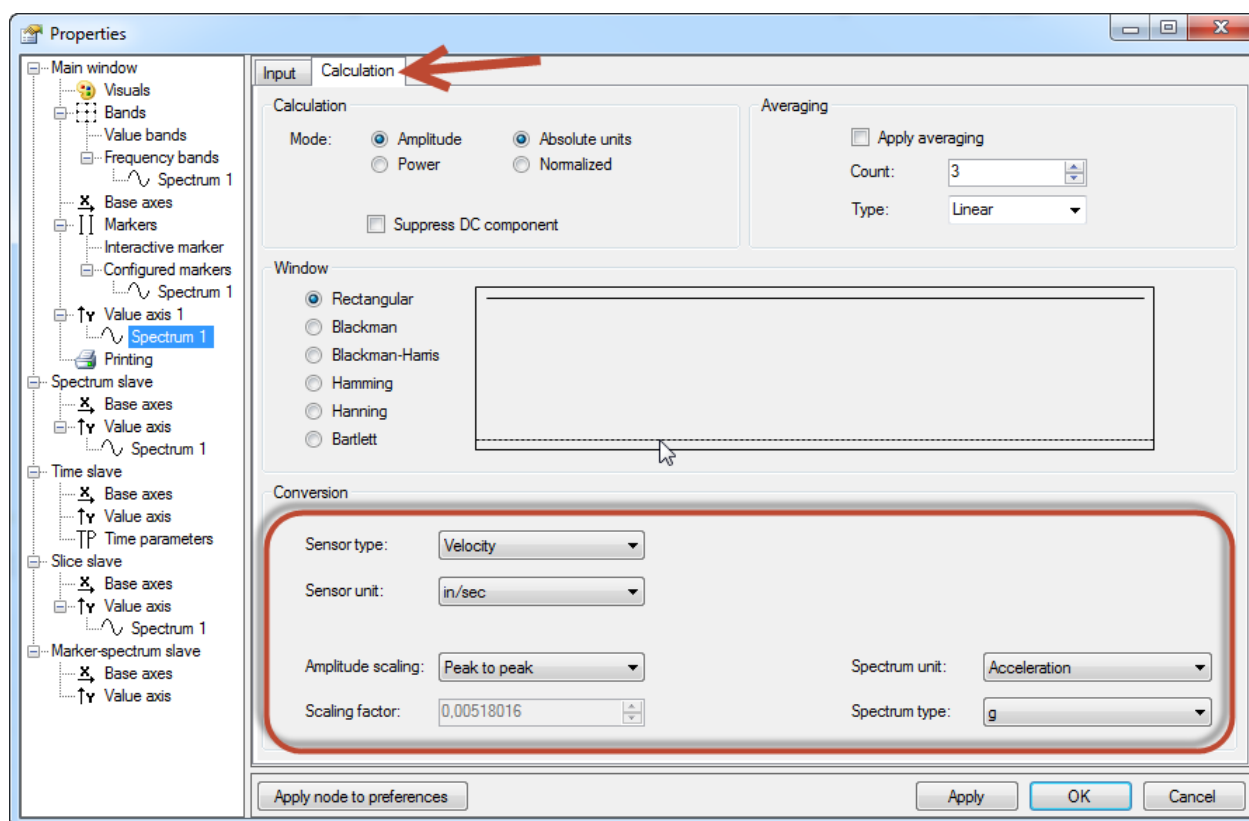
The *Length reset* setting is only applicable in case of length mode. If the *Length reset* value is higher than 0, the calculations will stall until the length signal has reached the *Length reset* value. Each time the length signal drops, the calculations will stall until the length signal has again increased with the *Length reset* value.

Note that order calculations are a very complex matter; please contact iba support for more information.

13.3.2 Integration / differentiation, units and scaling

Support for integration and differentiation was added, as well as support for units and additional scaling.

In the dialog of the properties, you can specify the required settings:



The following options are available:

- The sensor type and the sensor unit:
 - Sensor type: *not specified* - *displacement* - *velocity* - *acceleration*
 - Sensor unit, available choices depend on the sensor type:
 - *input signal unit* - *mm* - *inch* - *mils*
 - *mm/sec* - *inch/sec* - *mils/sec*
 - *mm/sec²* - *inch/sec²* - *mils/sec²*
- The spectrum type and the spectrum unit
 - Spectrum type, available choices depend on the sensor type:
 - *no integration* - *single integration* - *double integration* - *single differentiation* - *double differentiation*
 - *displacement* - *velocity* - *acceleration*
 - Spectrum unit, available options depend on the sensor type, sensor unit and spectrum type:
 - all variants of *mm*, *inch*, *mils* and *input signal unit*
 - in case the calculation mode is *Power* - *Absolute units*, all units are squared

- Additional scaling factor
 - Only editable if the sensor unit is set to *input signal unit*
 - If not editable, the scaling factor is determined by the combination of the sensor and spectrum unit. For instance, if the sensor unit is *in/sec* and the spectrum unit *g*, then the scaling factor is 0.00259008.
- Amplitude scaling
 - Apply a hardcoded scaling factor, three choices:
 - *None* (no factor): default
 - *Peak to peak* (factor 2): the spectrum peak of a perfect sine will be two times the amplitude of this sine
 - *RMS* ($1/\sqrt{2}$): the spectrum peak of a perfect sine will be equal to the rms value of this sine
 - This setting only applies if:
 - the sensor unit is not set to *input signal unit*
 - the calculation mode is *Amplitude - Absolute units*

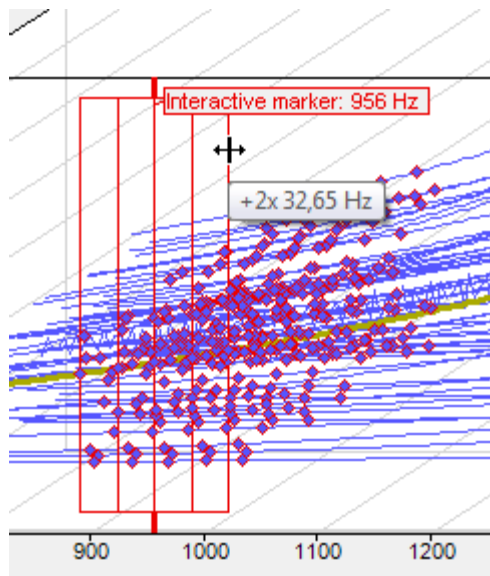
The differentiation and integration is done in the frequency domain.

In case the sensor unit is *Input signal unit*, the system will use the unit string of the input signal. The corresponding spectrum unit is displayed in the legend.

13.4 Markers

13.4.1 Sideband tooltip

When dragging the sideband of a marker, a tooltip appears with the actual sideband offset:

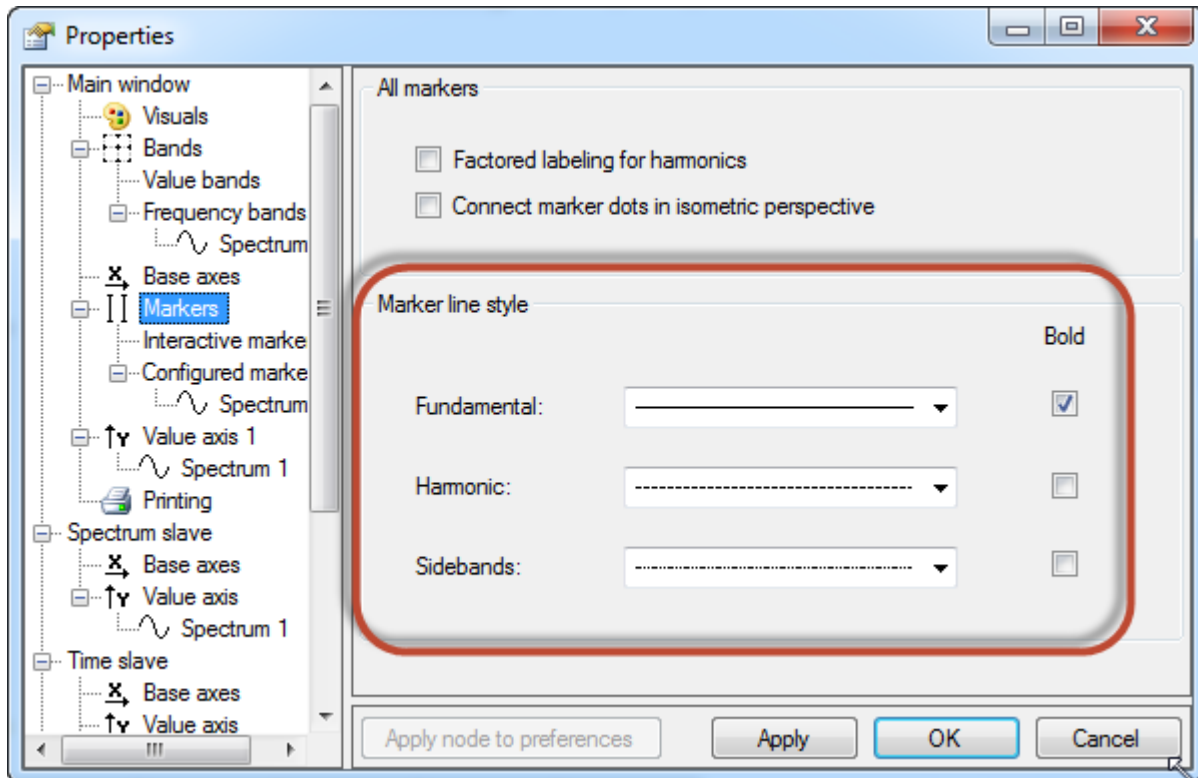


This only works if the sideband offset is a constant.

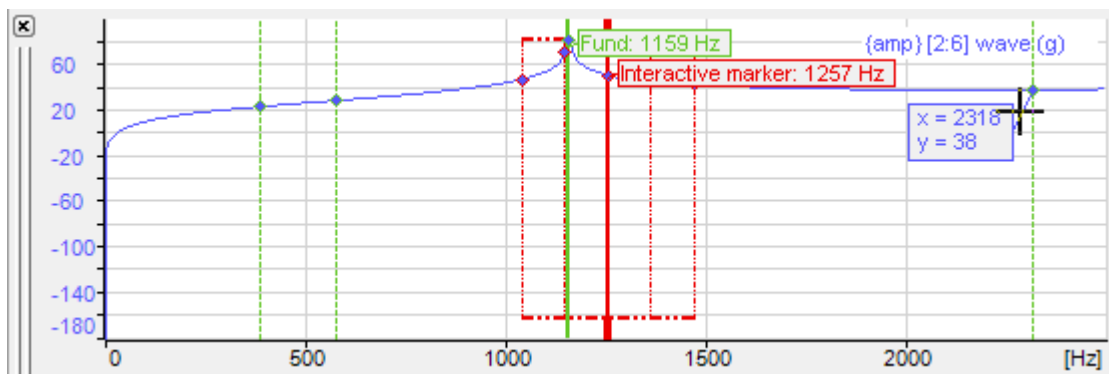
13.4.2 Marker styles

From this version on, it is possible to customize the style of the markers. One can configure the dashstyle and the thickness independently for the three types of markers:

- Fundamental
- Harmonics
- Sidebands



Below you find a screenshot corresponding to the previous settings:



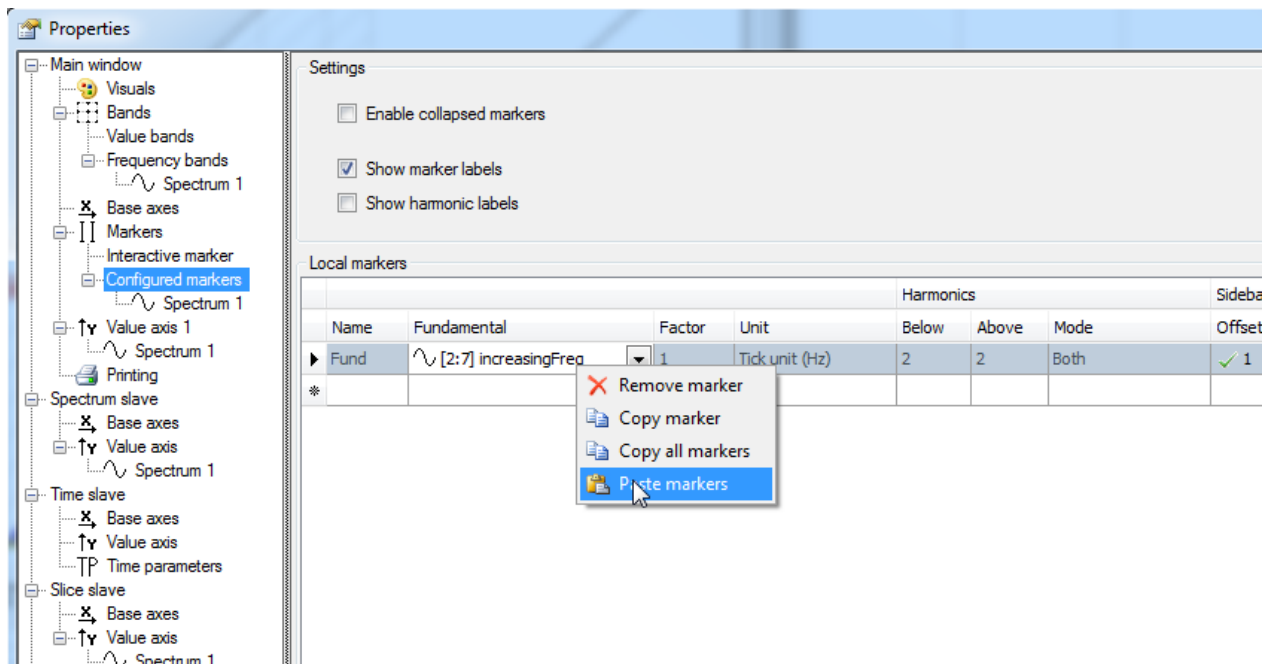
13.4.3 Harmonic marker dragging

One can drag a marker by dragging one of its harmonics. This only works if the frequency of the marker is a constant; it does not work in case the marker is linked to a signal.

13.4.4 Copy / paste of markers

One can copy the configured markers to the clipboard and paste them back in the markers grid. This allows reusing marker definitions in other FFT views. The copied markers are saved as tab-separated values on the clipboard.

With the shortcut **CTRL + C** one can copy the selected marker. With the shortcut **CTRL + V** one can paste the markers present on the clipboard onto the markers grid.

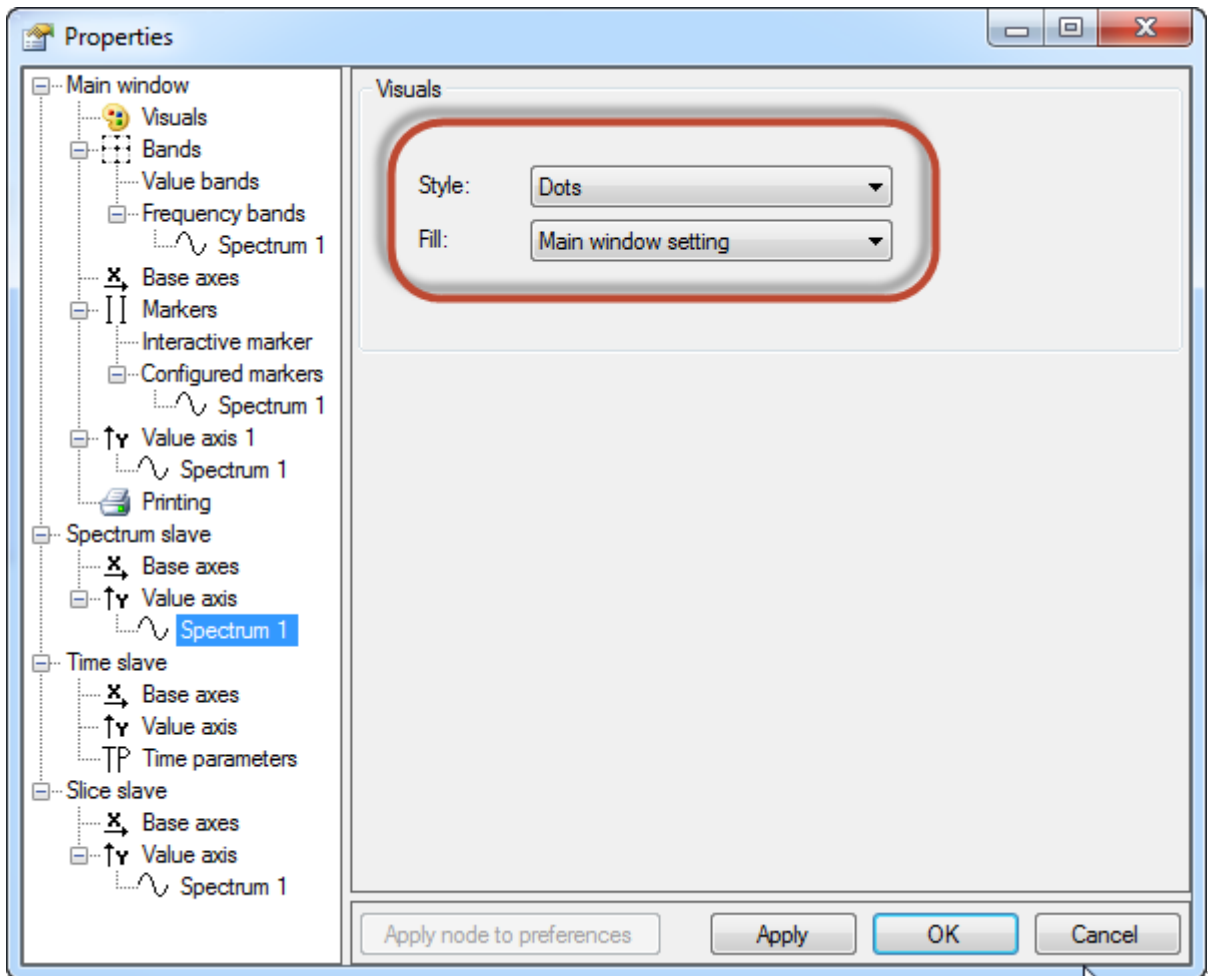


13.5 Printing

The printing function for the FFT view was improved. The tables in the spectrum and time slave are also printable now.

13.6 Slave visualization

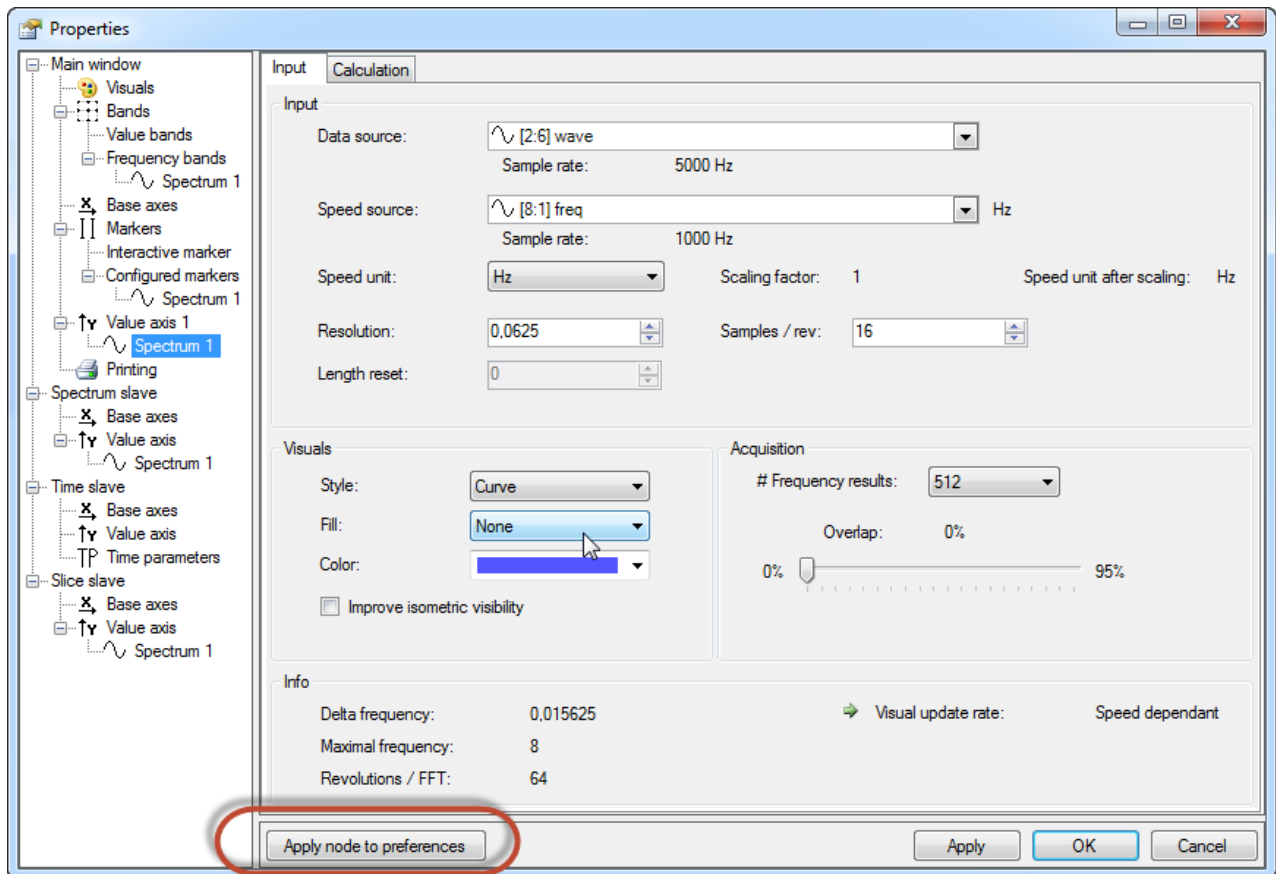
For each spectrum and slice slave it is possible to choose the curve and fill style individually:



For both settings, one of the choices is *Main window setting*. If this choice is selected, the curve style and fill style are the same as in the waterfall (main window). *Main window setting* is default for both settings.

13.7 ibaPda preferences

In the property dialog of the FFT view, a button was added to apply the visible settings to the ibaPda preferences.



The ibaPda preferences are kept in the registry. Everytime you create a new FFT view in the ibaPda client, this new view will be initialized with the settings from the registry. You can access the ibaPda preferences via the menu: *Configure - Preferences*.

It is important to know that not all settings of the FFT view are saved into the registry. For instance, it does not make sense to save the data or speed source in the registry. That's why signal-specific settings are never saved into the registry.