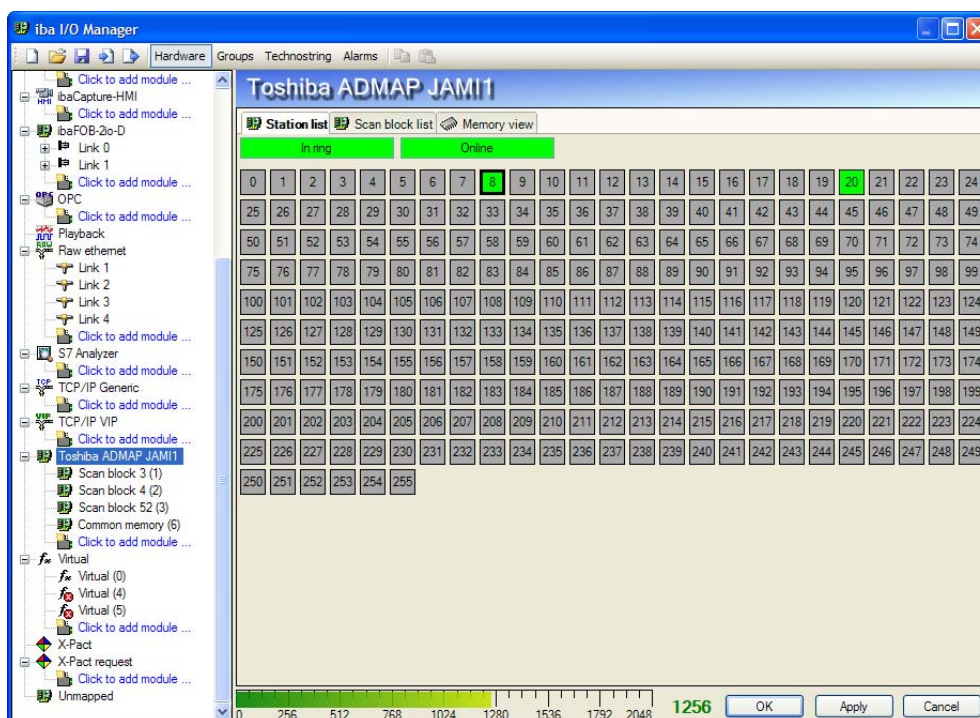


New features in ibaPDA v6.27.0

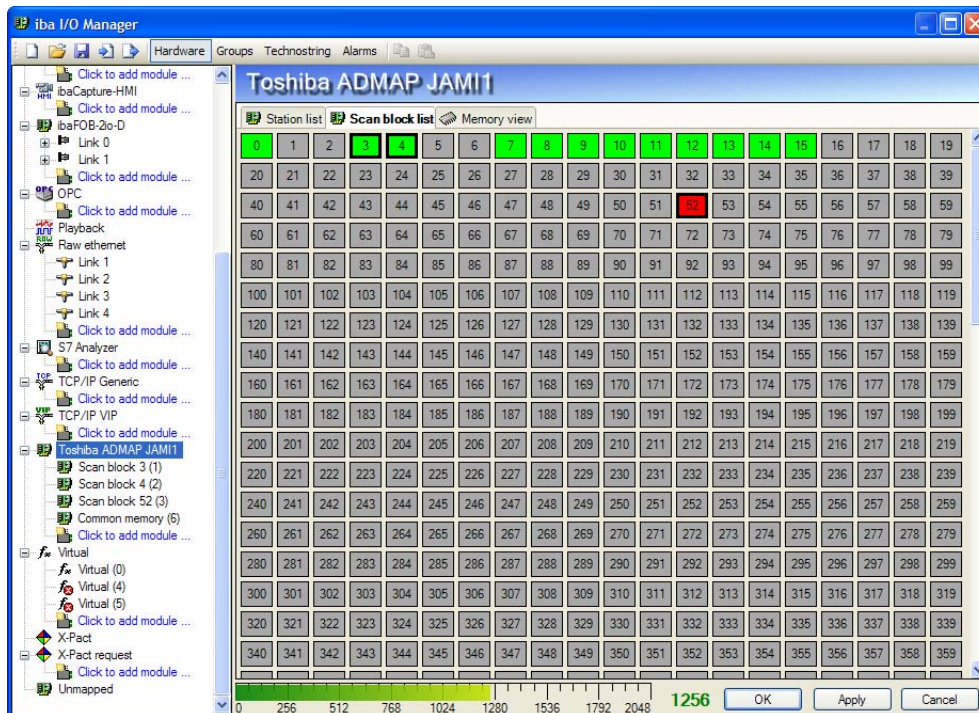
1 Toshiba ADMAP JAMI1 interface

The JAMI1 board is a PCI board from Toshiba that provides a PC access to the legacy ADMAP network. The ADMAP network is a redundant network. All stations on the network see the same 128 kB of data. This data is organized in 1024 blocks of 128 bytes. Each block can have 1 writer and multiple readers. The station that is writing to a block is called a talker. There are 3 time classes defined for the blocks: fast, medium and slow. The talker determines the update rate of the block it writes to.

When the board is installed in the PC the I/O manager will show the Toshiba ADMAP JAMI1 interface. If you have a license for the board then the title area will be a blue gradient otherwise it will be a red gradient with an icon of a dongle with a red cross. There are 3 tabs. The first tab shows the station list.

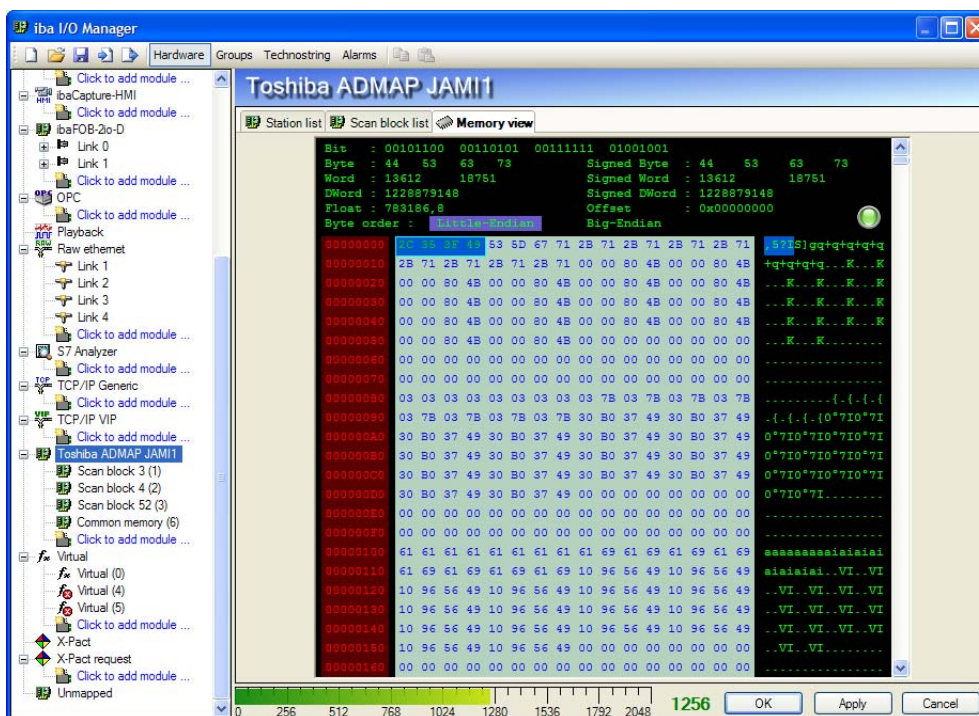


The first two labels show if the board itself is connected to the ring and if it is online. When a board is online then all data that is written to the board will be transmitted to the other stations in the ring. When you boot the PC then the board will be automatically in online status. You also see 256 boxes. Each box represents a station on the ADMAP network. The station that corresponds with the JAMI1 board in the PC has a thick border. When a station is not connected then it is shown in dark gray. When a station is connected and it is in online mode then it is green. When a station is connected and it is in standby mode then it is red.



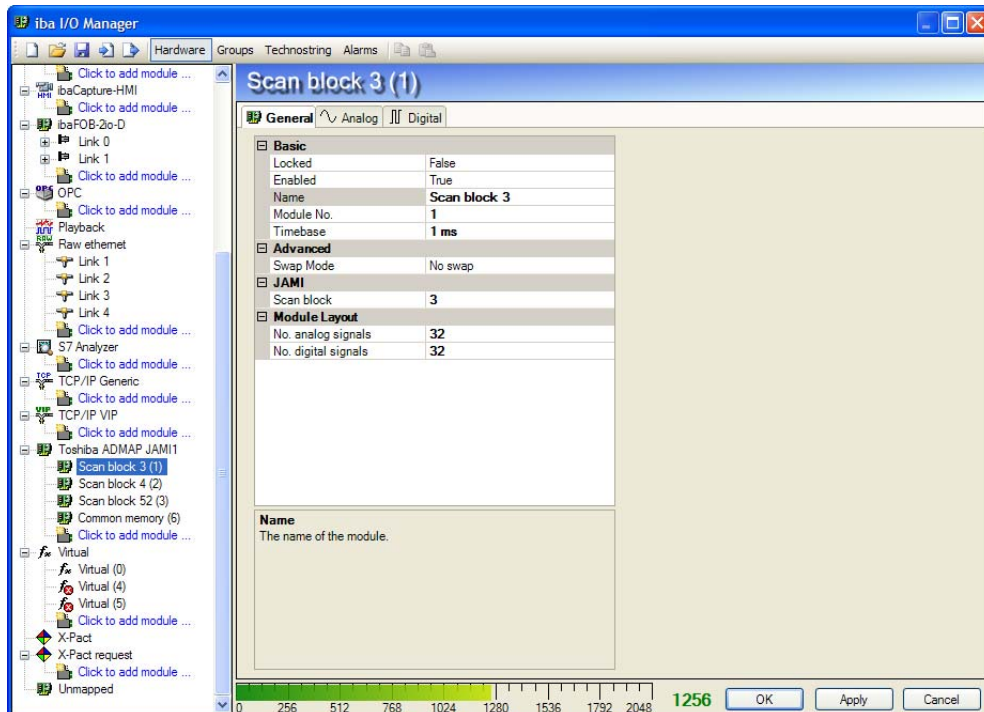
The second tab shows the scan blocks. There are 1024 boxes and each box corresponds to a scan block. A scan block is healthy when there is a talker for that block, so when there is a station writing to that block. The healthy blocks are shown in green. The unhealthy blocks are shown in red when pda is measuring from them and in gray otherwise. The blocks that pda is measuring have a thick border.

The third tab show a view of the complete memory on the board. The first 128kB (address 0x00000000 – 0x001FFFFF) correspond with the 1024 scan blocks. The second 128kB (address 0x00020000 – 0x0003FFFF) correspond with the common memory area. This area contains diagnostic information like the scan block healthy list. You can read the manual of the JAMI1 board to find out the exact meaning of each byte in the common memory area.

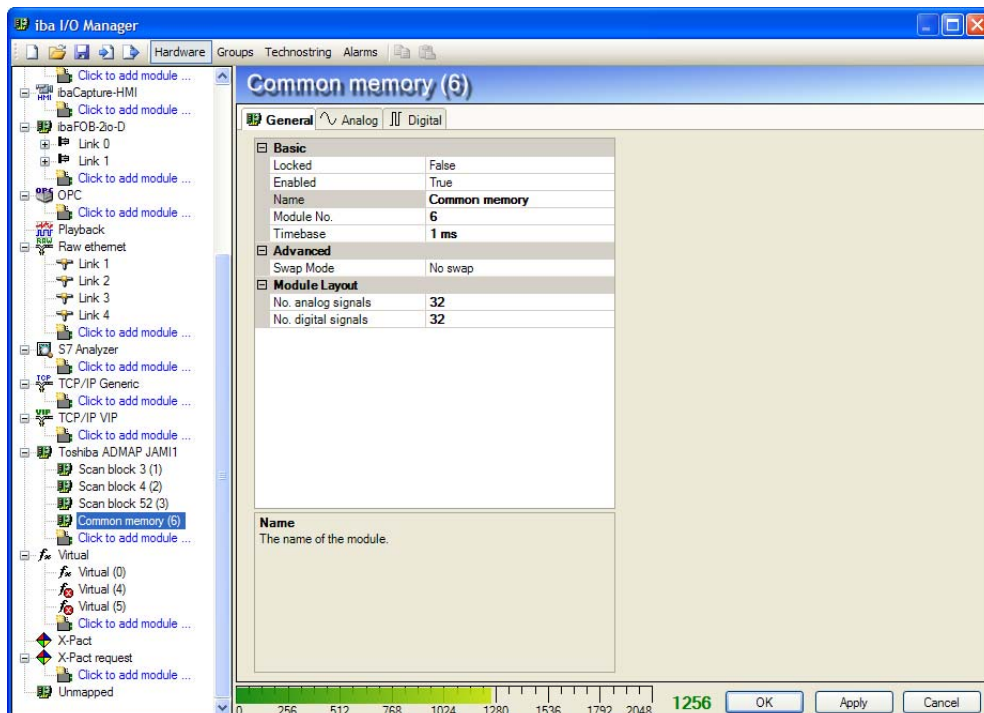


There are 2 types of modules you can add to the Toshiba ADMAP JAMI1 interface: scan block module and common memory module.

The scan block module is able to measure the 128 bytes of a scan block. On the general tab you have to specify the scan block number (0 – 1023). You can also specify the swap mode and the number of signals.



The common memory module is able to measure the 128kB of common memory. On the general tab you can specify the swap mode and the number of signals.



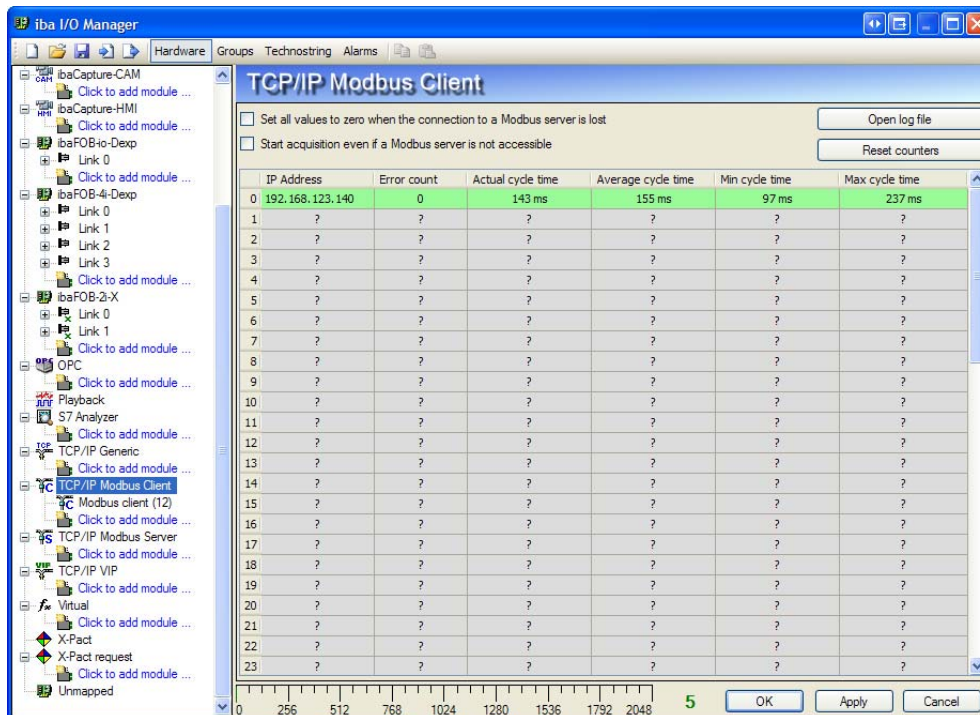
The analog and digital signals are configured in the same way for both module types. For an analog signal you have to specify the address (in hex or decimal) and the datatype. The actual column shows the current value of the signal. By clicking on the “actual column” header you can switch between raw and scaled view of the value. By clicking on the “address column” header you can let pda calculate the addresses based on the datatype of the signals.

Name	Unit	Gain	Offset	Address	DataType	Active	Actual
0		1	0	0x0	BYTE	<input checked="" type="checkbox"/>	145
1		1	0	0x1	BYTE	<input checked="" type="checkbox"/>	145
2		1	0	0x2	BYTE	<input checked="" type="checkbox"/>	145
3		1	0	0x3	BYTE	<input checked="" type="checkbox"/>	145
4		1	0	0x4	BYTE	<input checked="" type="checkbox"/>	145
5		1	0	0x5	BYTE	<input checked="" type="checkbox"/>	145
6		1	0	0x6	BYTE	<input checked="" type="checkbox"/>	145
7		1	0	0x7	BYTE	<input checked="" type="checkbox"/>	145
8		1	0	0x8	INT	<input checked="" type="checkbox"/>	27281
9		1	0	0xA	INT	<input checked="" type="checkbox"/>	27281
10		1	0	0xC	INT	<input checked="" type="checkbox"/>	27281
11		1	0	0xE	INT	<input checked="" type="checkbox"/>	27281
12		1	0	0x10	INT	<input checked="" type="checkbox"/>	27281
13		1	0	0x12	INT	<input checked="" type="checkbox"/>	27281
14		1	0	0x14	INT	<input checked="" type="checkbox"/>	27281
15		1	0	0x16	INT	<input checked="" type="checkbox"/>	27281
16		1	0	0x18	FLOAT	<input checked="" type="checkbox"/>	879249
17		1	0	0x1C	FLOAT	<input checked="" type="checkbox"/>	879249
18		1	0	0x20	FLOAT	<input checked="" type="checkbox"/>	879249
19		1	0	0x24	FLOAT	<input checked="" type="checkbox"/>	879249
20		1	0	0x28	FLOAT	<input checked="" type="checkbox"/>	879249
21		1	0	0x2C	FLOAT	<input checked="" type="checkbox"/>	879249
22		1	0	0x30	FLOAT	<input checked="" type="checkbox"/>	879249
23		1	0	0x34	FLOAT	<input checked="" type="checkbox"/>	879249

For a digital signal you have to specify the address and the bit number. The bit number goes from 0 to 31.

Name	Address	Bit no.	Active	Actual
0	0x0	0	<input checked="" type="checkbox"/>	1
1	0x0	1	<input checked="" type="checkbox"/>	0
2	0x0	2	<input checked="" type="checkbox"/>	0
3	0x0	3	<input checked="" type="checkbox"/>	0
4	0x0	4	<input checked="" type="checkbox"/>	1
5	0x0	5	<input checked="" type="checkbox"/>	0
6	0x0	6	<input checked="" type="checkbox"/>	0
7	0x0	7	<input checked="" type="checkbox"/>	1
8	0x0	8	<input checked="" type="checkbox"/>	1
9	0x0	9	<input checked="" type="checkbox"/>	0
10	0x0	10	<input checked="" type="checkbox"/>	0
11	0x0	11	<input checked="" type="checkbox"/>	0
12	0x0	12	<input checked="" type="checkbox"/>	1
13	0x0	13	<input checked="" type="checkbox"/>	0
14	0x0	14	<input checked="" type="checkbox"/>	0
15	0x0	15	<input checked="" type="checkbox"/>	1
16	0x0	16	<input checked="" type="checkbox"/>	1
17	0x0	17	<input checked="" type="checkbox"/>	0
18	0x0	18	<input checked="" type="checkbox"/>	0
19	0x0	19	<input checked="" type="checkbox"/>	0
20	0x0	20	<input checked="" type="checkbox"/>	1
21	0x0	21	<input checked="" type="checkbox"/>	0
22	0x0	22	<input checked="" type="checkbox"/>	0
23	0x0	23	<input checked="" type="checkbox"/>	1

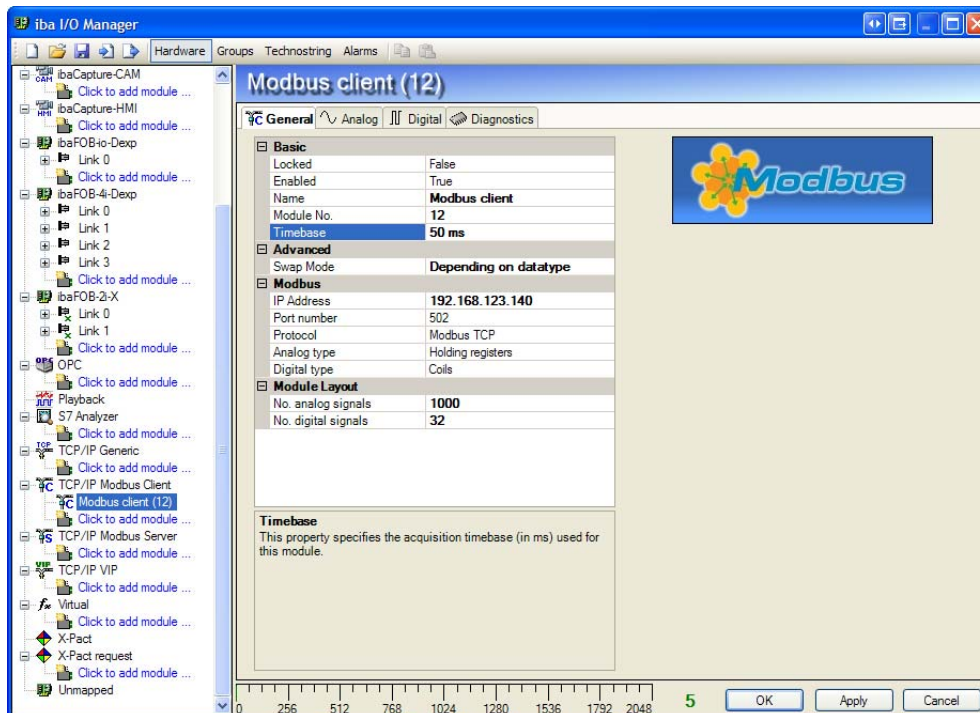
2 TCP/IP Modbus client interface



The TCP/IP Modbus client interface lets pda act as a Modbus client. Pda connects to a Modbus server and periodically reads data from it. In the I/O manager you see a list of connections. There are 64 possible connections. Each connection corresponds with a Modbus client module. Each connection has a separate thread so that all connections are independent of each other. A slow Modbus server on one connection will not stop a fast Modbus server on another connection. The cycle time mentioned in the connection grid is the time it takes to read all the data configured on that connection. The reset counters button can be used to reset the error counter and the average, min and max cycle times.

You can optionally set all values to zero when the connection to a Modbus server is lost by checking the checkbox. By default the acquisition cannot be started when a Modbus server is not accessible but there is a checkbox to force the start anyway.

In case of problems you can push the “Open log file” button to check out all communication via Modbus TCP/IP.



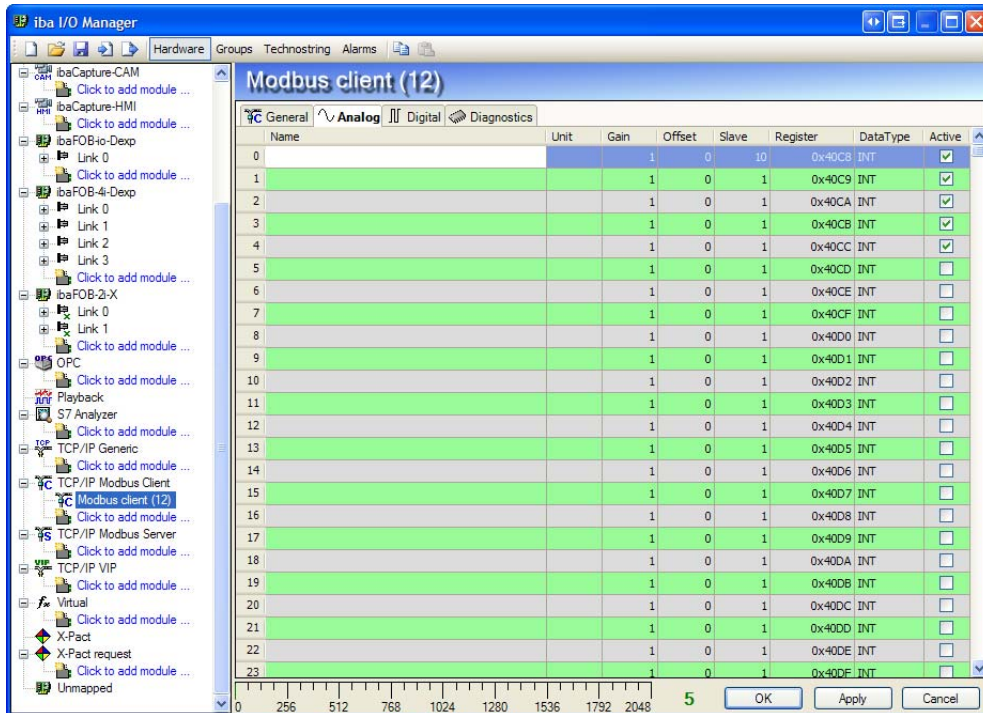
On the general tab of a Modbus client module you first have to enter the IP address of the Modbus server. You can set the port number. By default it is set to the standard Modbus port 502.

You then have to choose the protocol. Pda supports 2 protocols: Modbus TCP and Modbus serial. Modbus TCP is the normal protocol used for Modbus over TCP. Modbus serial is the serial RTU protocol that is sent as is over TCP. The serial protocol needs to be used with certain Ethernet to serial gateways, e.g. the IF2E001 from IME. This gateway just takes all bytes it receives from the network and puts them on the serial line.

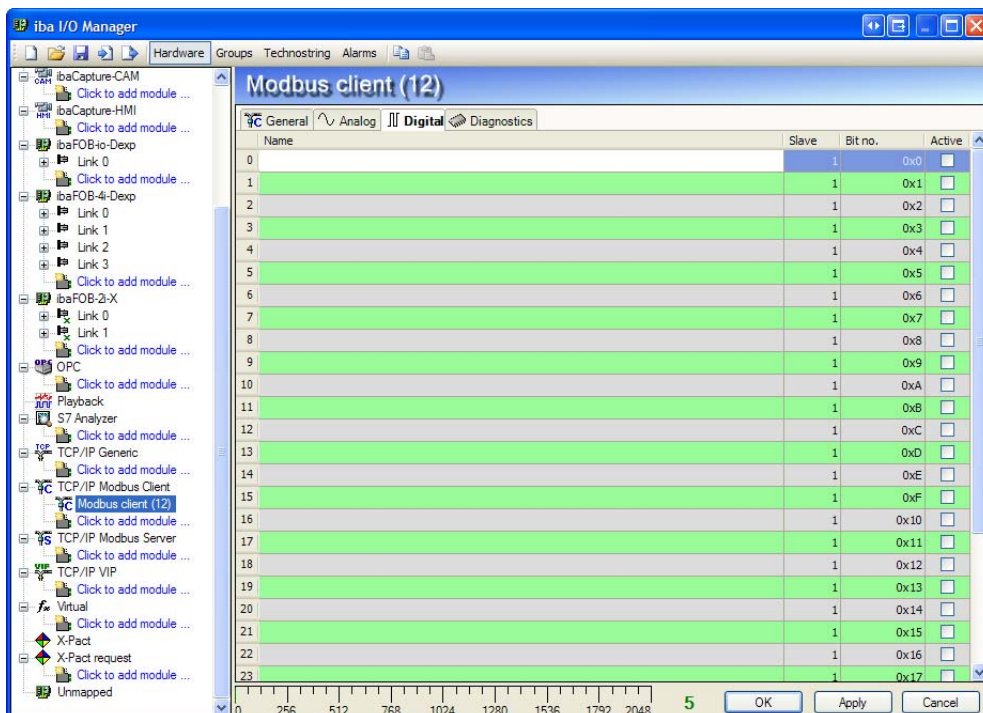
The next thing to setup is the type of data you want to measure. For the analog data you can choose between holding registers (function code 3) and input registers (function code 4). For the digital data you can choose between coils (function code 1) and discrete inputs (function code 2). Which type you have to choose will depend on the Modbus server.

You can setup the number of analog and digital signals you want to measure. The maximum number of signals is 1000.

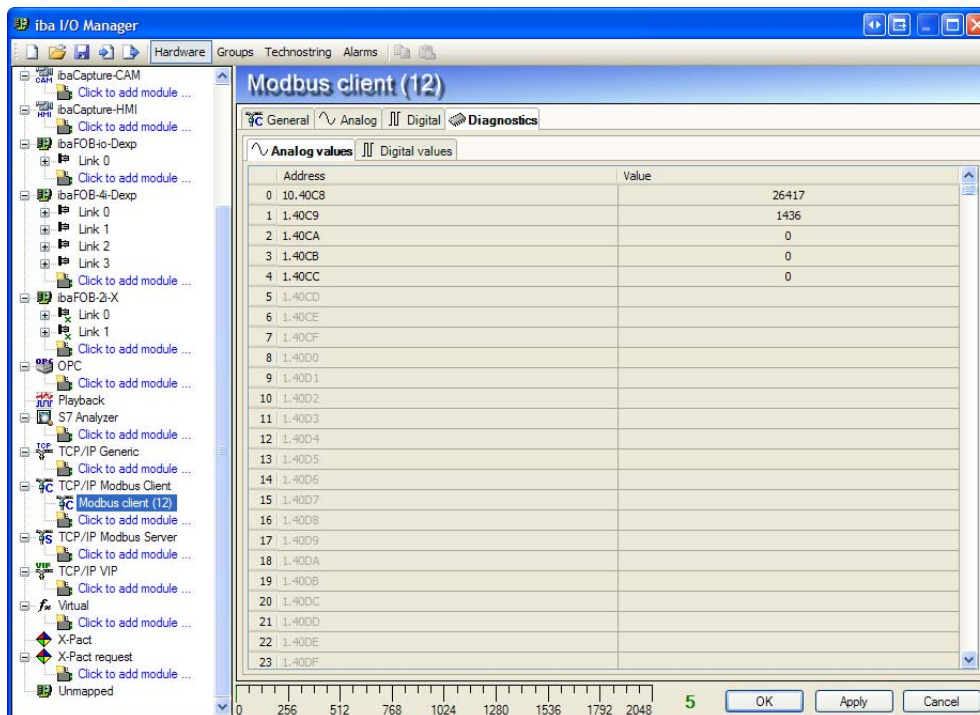
The timebase determines how fast pda will try to read the data from the Modbus server. If the Modbus server doesn't respond as fast as the timebase then pda will read the data slower. Make sure you select an appropriate timebase for your Modbus server. If the timebase is too fast then pda might overload the Modbus server.



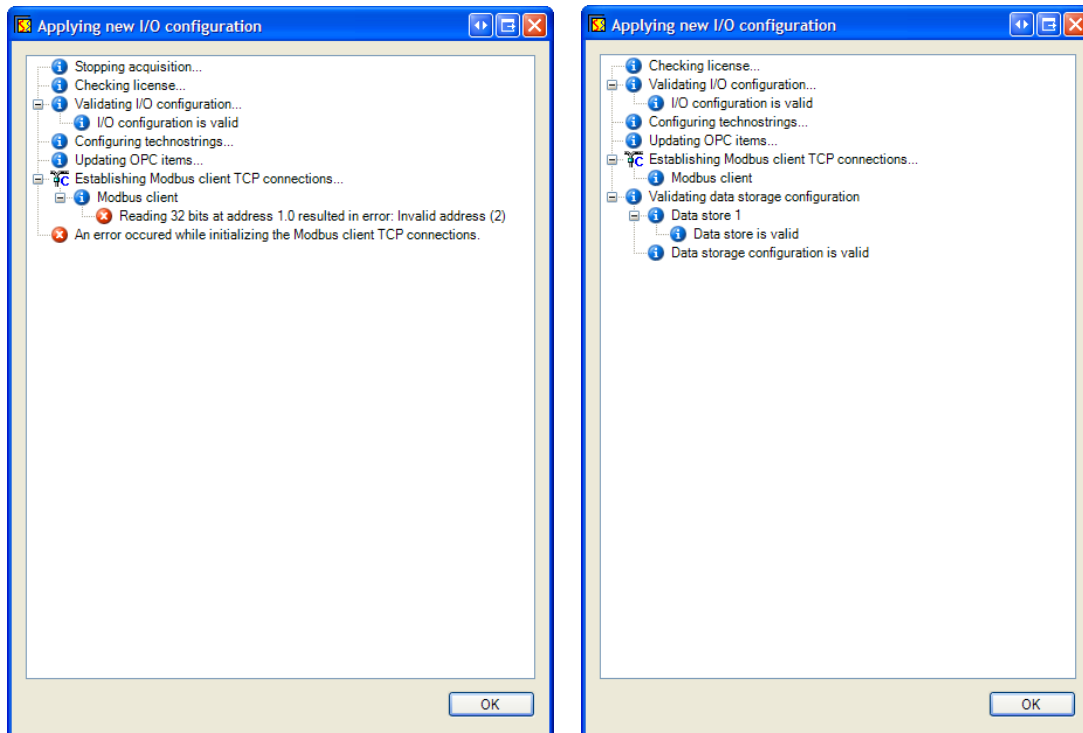
For each analog signal you have to specify the slave number, register number and data type. The slave number is normally only relevant when you are working with a gateway that has multiple Modbus slaves connected. In Modbus TCP mode the slave number is ignored by most Modbus servers. The register number goes from 0 to 65535. Each register is 16 bit wide. You can specify the register number in hexadecimal format or in decimal format. You switch between them via the context menu. The default data type is INT (= 16 bit signed integer) but if you want you can choose different types. On the general tab you can also change the swapping mode. By default it is set to “Depending on datatype” because the Modbus standard uses big-endian encoding and a PC uses little-endian encoding.



For each analog signal you have to specify the slave number and bit number. The bit number corresponds with the coil number or discrete input number. The valid range is from 0 to 65535. Like the analog register number you can switch between decimal and hexadecimal notation via the context menu.



The last tab is the diagnostics tab. This tab shows the current values of the measured signals.



At the start of the acquisition pda tries to connect to the Modbus servers and it tries to read the data once. If the server returns an error then it is shown in the validation form.

3 ibaBM-SiLink

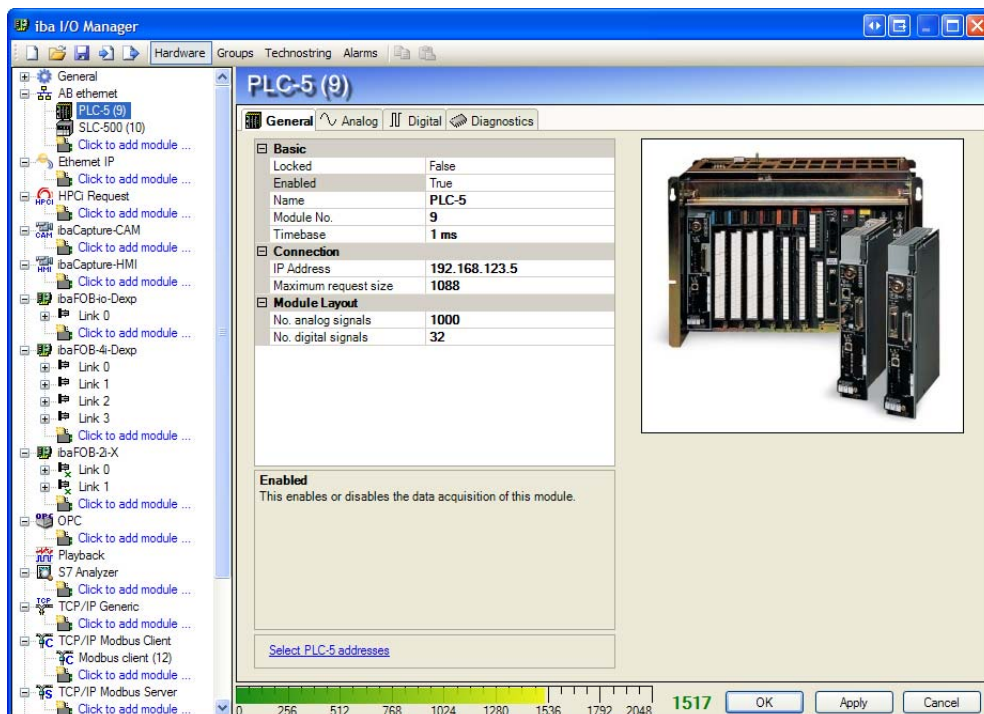
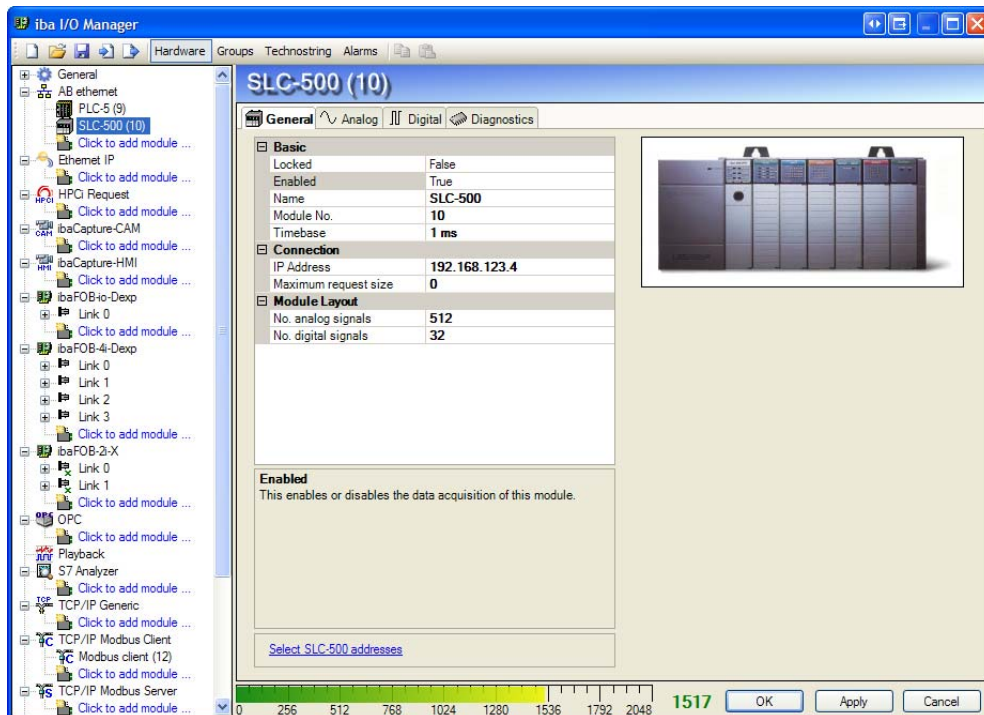
See separate manual ibaBM-SiLink_v1.0_en.doc

4 ibaBM-eCAT

The ibaBM-eCAT is actually internally an ibaPADU-S-CM connected to an ibaMS-eCAT. In pda it is shown as a single module. Check out chapter 3 of the new features document of ibaPDA 6.23.0 to get more information about the etherCAT sniffer module.

5 AB Ethernet interface

The AB Ethernet interface now also supports the SLC-500 series PLCs. The SLC5/05 has Ethernet on board. The SLC500, SLC5/01, SLC5/02, SLC5/03, SLC5/04 don't have Ethernet on board so they all need the 1761-NET-ENI to support AB Ethernet.



The SLC-500 module in pda is very similar to the PLC-5 module. The only difference is that the addresses of the SLC-500 have other limits than the addresses of the PLC-5. The file number is limited to 255 and the element number is also limited to 255.

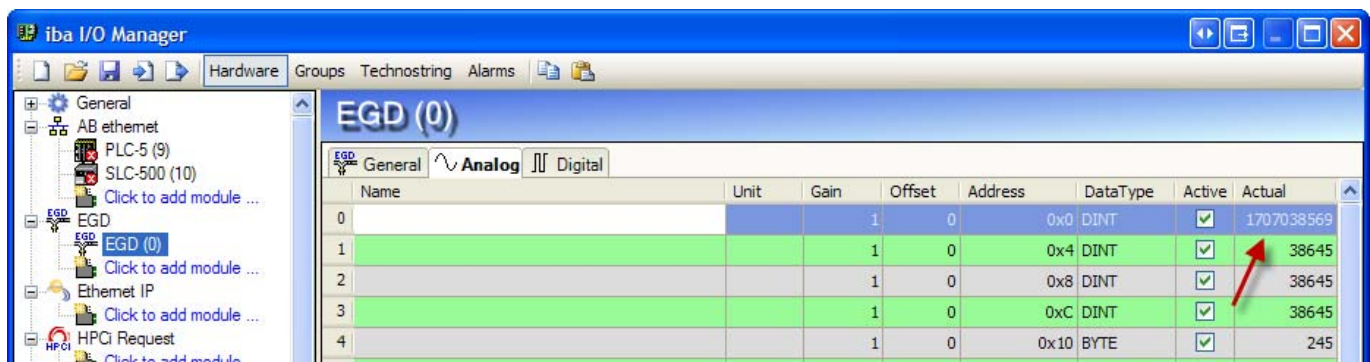
There is a new property for both SLC-500 and PLC-5 modules: maximum request size. With this property you can specify what the maximum number of words is that you can request within a single message sent to the PLC. In previous pda versions this was a fixed limit of 126 words. Now pda supports longer messages. The PLC-5/40 that I tested on supports 1088 words in a single message. On the SLC5/05 I haven't found a limit. The advantage of longer messages is that you can read more data in less time. It takes more time to read 1000 words with 2 messages of 500 words than 1 message of 1000 words. The only limitation is that 1 message reads always from 1 file so all words need to be in 1 file. If you want to read words from different files than pda will automatically send multiple messages.

6 Dat file changes

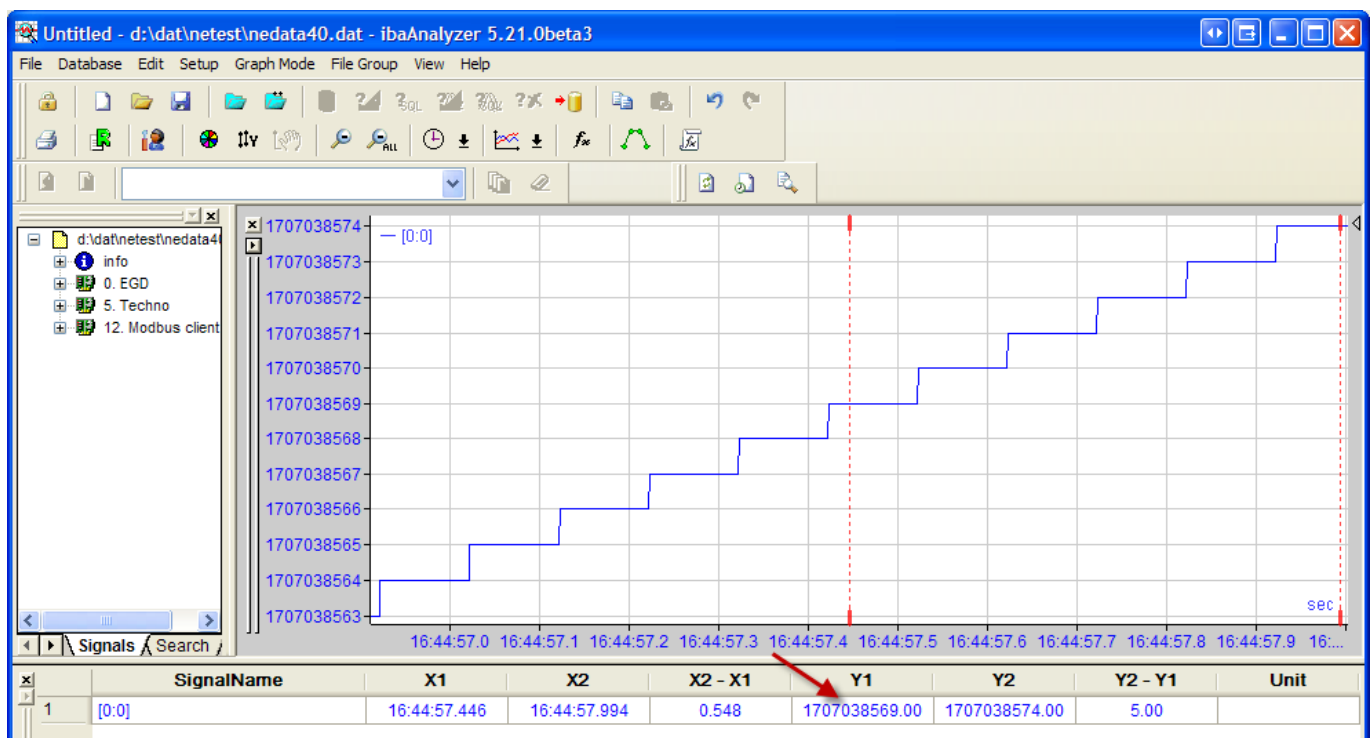
All these changes in the dat file format require ibaAnalyzer v5.21.0 or higher.

6.1 32 bit integer and 64 bit floating point data

In previous versions of pda the values of analog signals with 32 bit integer datatype (DWORD, DINT, DWORD_B, DINT_B) were converted to a 32 bit floating-point value. This resulted in a loss of resolution of 8 bits from 32 bits to 24 bits. Therefore you only had 7 significant digits in the value instead of the 10 from the original value. In pda 6.27.0 the value is no longer converted and it is written as is to the dat file. In the screenshots below you can see the value in pda and the value in ibaAnalyzer. It still has the full resolution. All calculations that ibaAnalyzer does with this 32 bit integer data will preserve the full resolution.



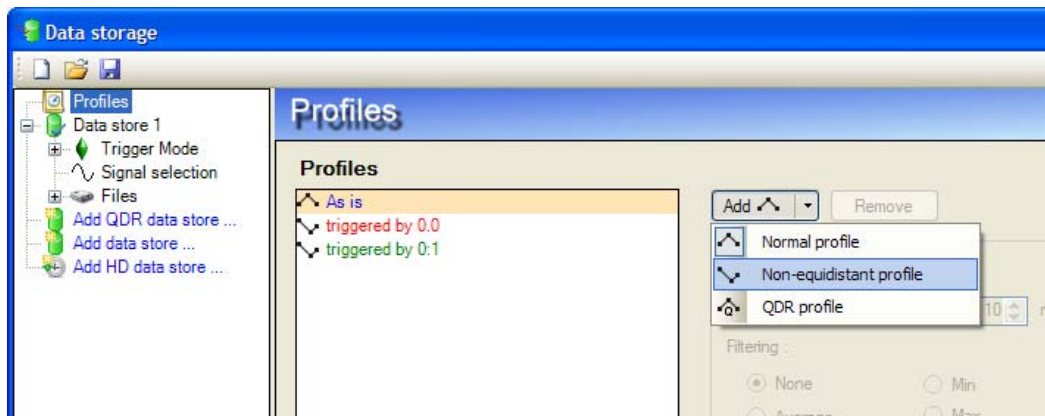
Name	Unit	Gain	Offset	Address	DataType	Active	Actual
0		1	0	0x0	DINT	<input checked="" type="checkbox"/>	1707038569
1		1	0	0x4	DINT	<input checked="" type="checkbox"/>	38645
2		1	0	0x8	DINT	<input checked="" type="checkbox"/>	38645
3		1	0	0xC	DINT	<input checked="" type="checkbox"/>	38645
4		1	0	0x10	BYTE	<input checked="" type="checkbox"/>	245



The same applies for analog signals with 64 bit floating point data type (DOUBLE). The double data type is preserved in the dat file and ibaAnalyzer will display and calculate the data correctly.

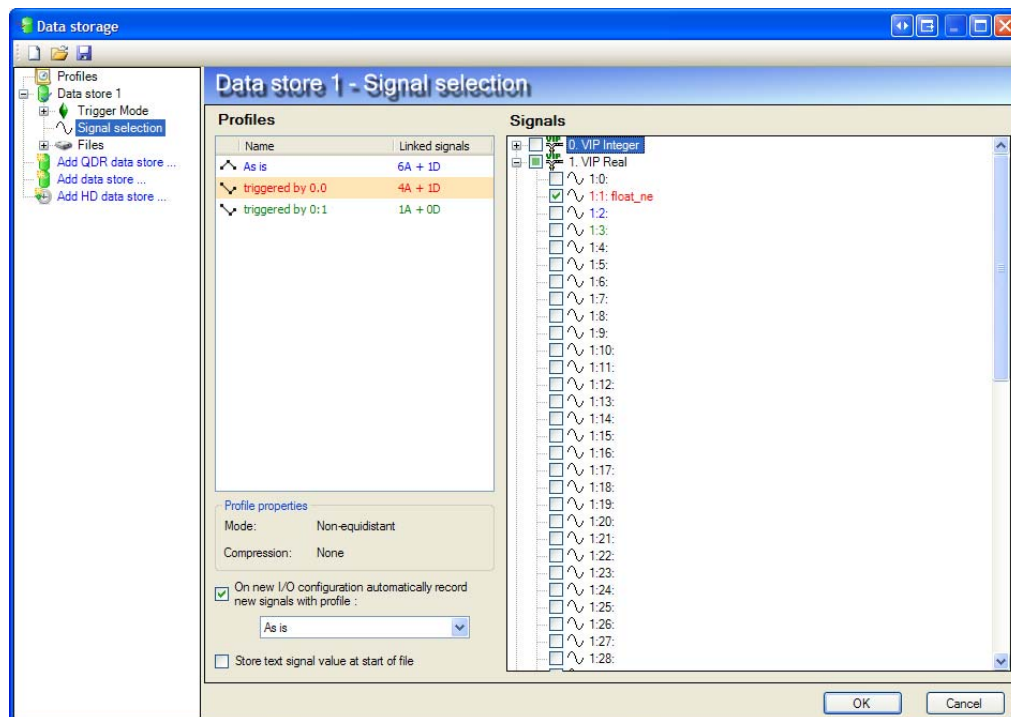
Pda still converts 32 bit integer and 64 bit floating point data to 32 bit floating point values when they are used in virtual signals and when they are transferred to the client for display.

6.2 Non-equidistant data

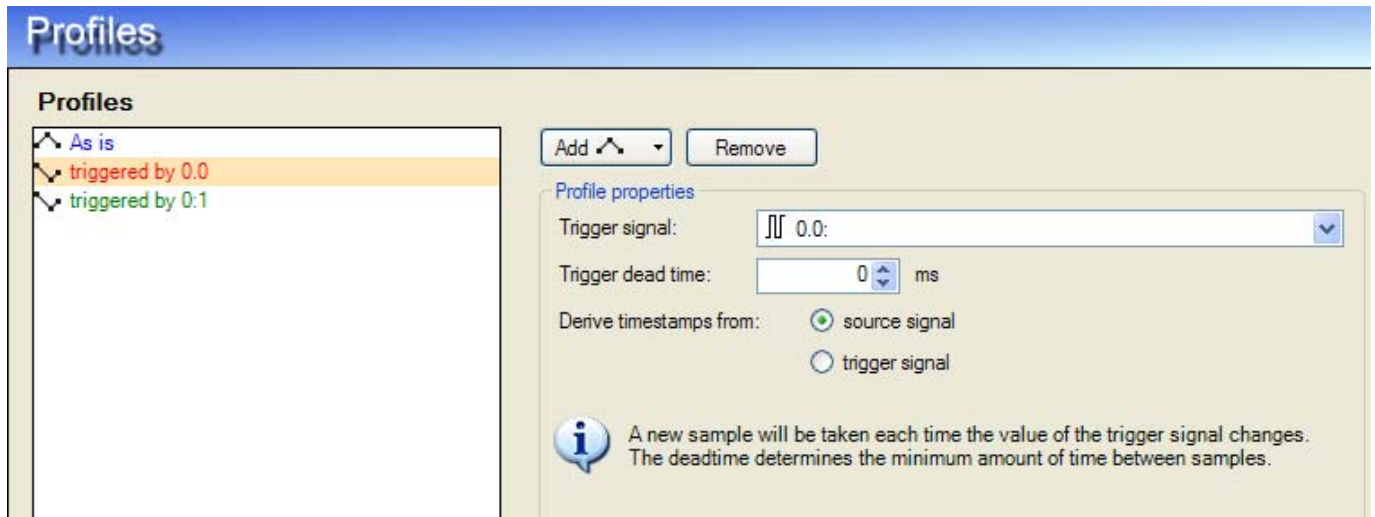


In pda you can now add different archive profile types. There are 3 types:

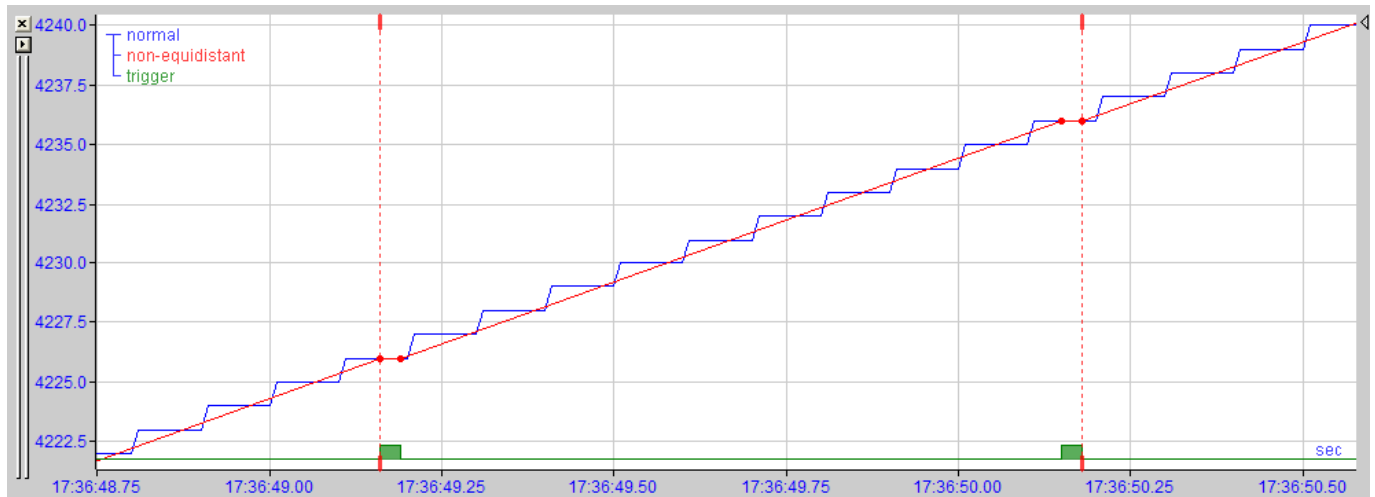
- Normal profile which is timebased
- QDR profile which is length and/or timebased
- Non-equidistant profile



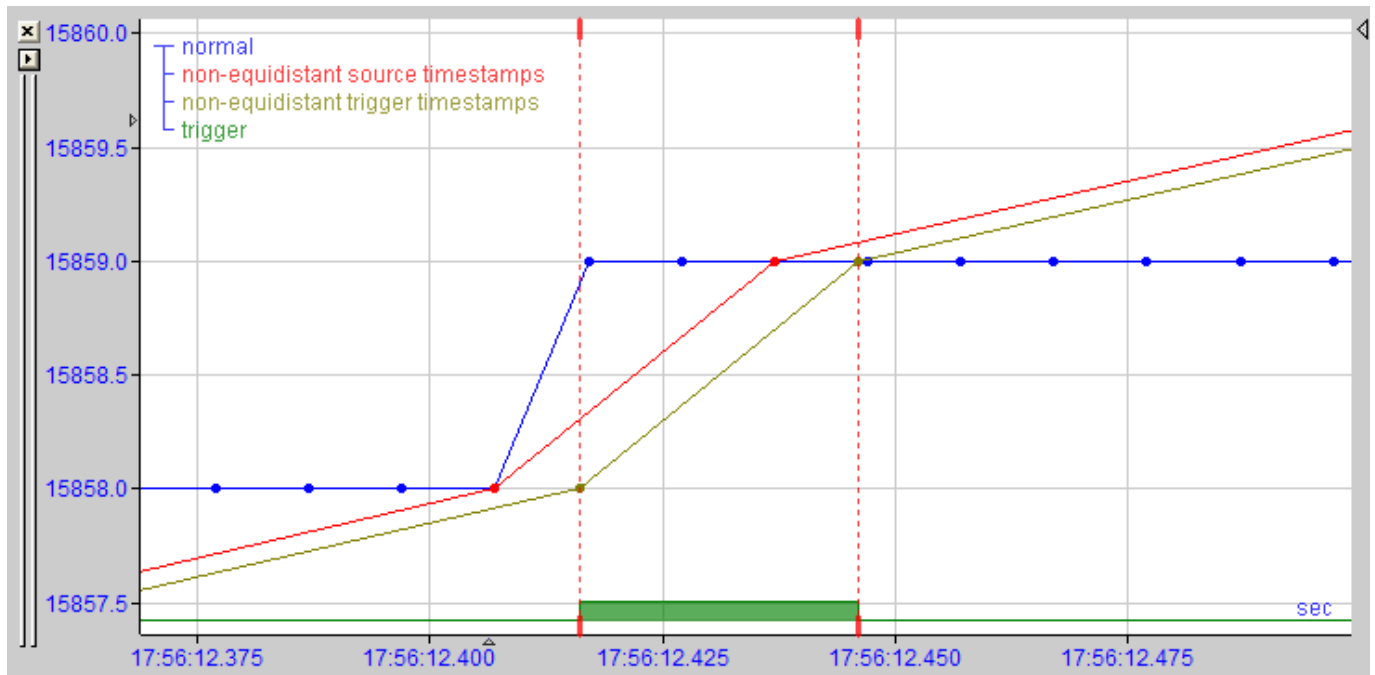
The different profile types have a different icon. This icon is shown again in the profiles grid at the signal selection.



The normal profile and the QDR profile already existed in pda versions prior to 6.27.0. The non-equidistant profile is new. This profile will take a sample each time the value of a trigger signal changes. The trigger dead time determines the minimum amount of time between samples.



The screenshot shows the result in ibiAnalyzer. The blue signal is the original signal measured with the As is profile. A copy of the original signal is measured with a non-equidistant profile. This is the red signal. The trigger signal was the green digital signal. You see that each time the trigger signal changes a sample for the red non-equidistant signal is generated.



When the trigger signal and the source signal have different timebases then you can choose which timestamps to use for the non-equidistant signal. The screenshots shows the two options. The blue signal is the source signal. The green digital signal is the trigger signal. The source signal is sampled at 10ms. The trigger signal is sampled at 1ms. You see that the trigger value changes at a time that doesn't correspond with a sample of the source signal.

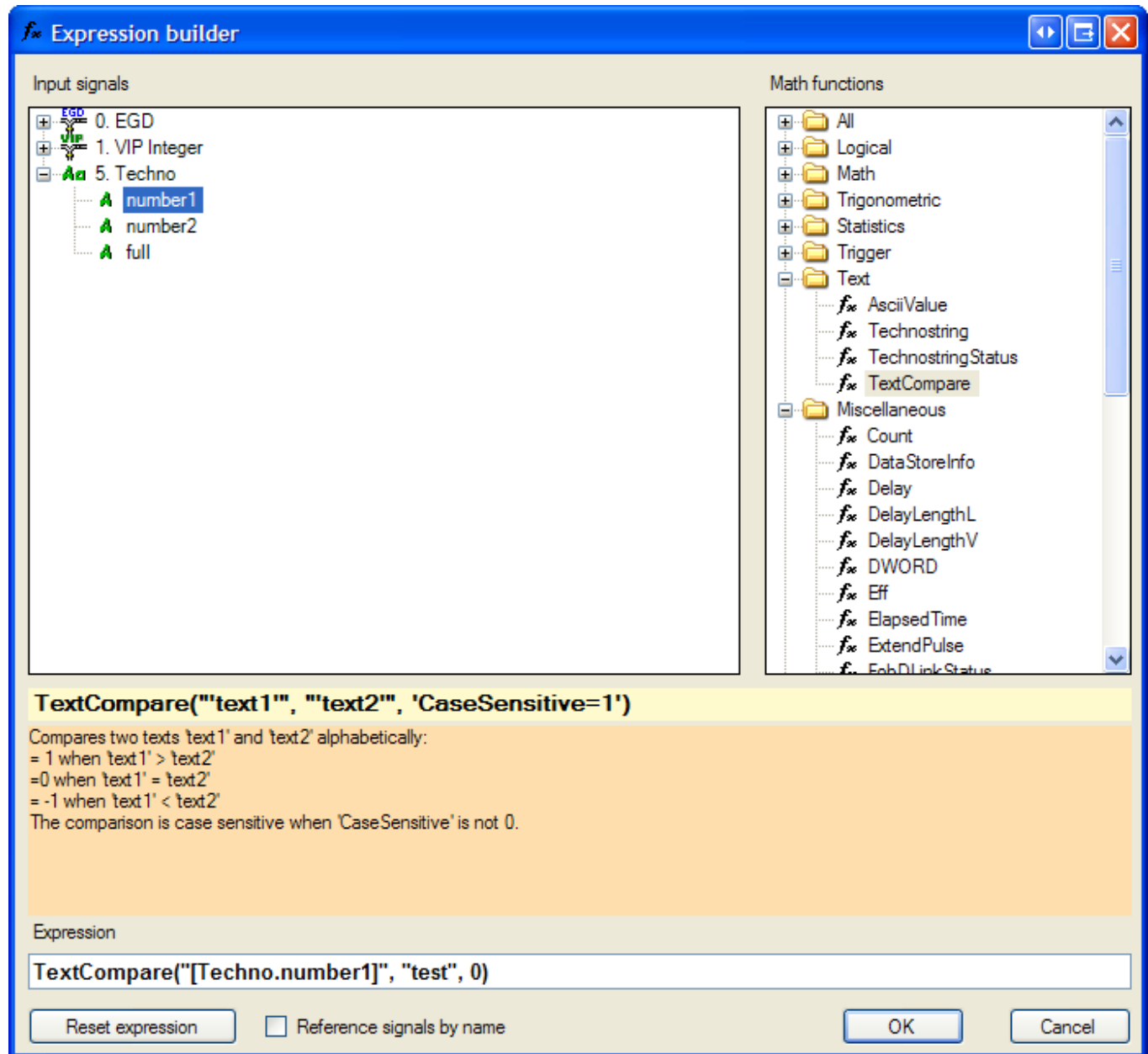
The red signal is the non-equidistant signal that uses the timestamps from the source signal. You see that the samples of the red signal correspond with samples of the original signal. When you choose source signal timestamps then the timestamp will be smaller or equal to the trigger timestamp.

The brown signal is the non-equidistant signal that uses the timestamps from the trigger signal. You see that the samples of the brown signal are the samples of the original signal shifted to the trigger timestamps.

The advantage of source timestamps is that the original samples are used. The disadvantage is that if you have multiple source signals with different timebases that the generated non-equidistant signals have samples at different timestamps. If you then do calculations with these non-equidistant signals then they will be resampled to equidistant signals and the calculations will be done on equidistant signals.

The disadvantage of trigger timestamps is that the original samples get shifted. The advantage is that all non-equidistant signals generated by the same profile have the same timestamps no matter what their original timebases were. If you then do calculations on the non-equidistant signals the calculations will be done on the non-equidistant signals and the result will again be a non-equidistant signal.

7 Text arguments in expressions

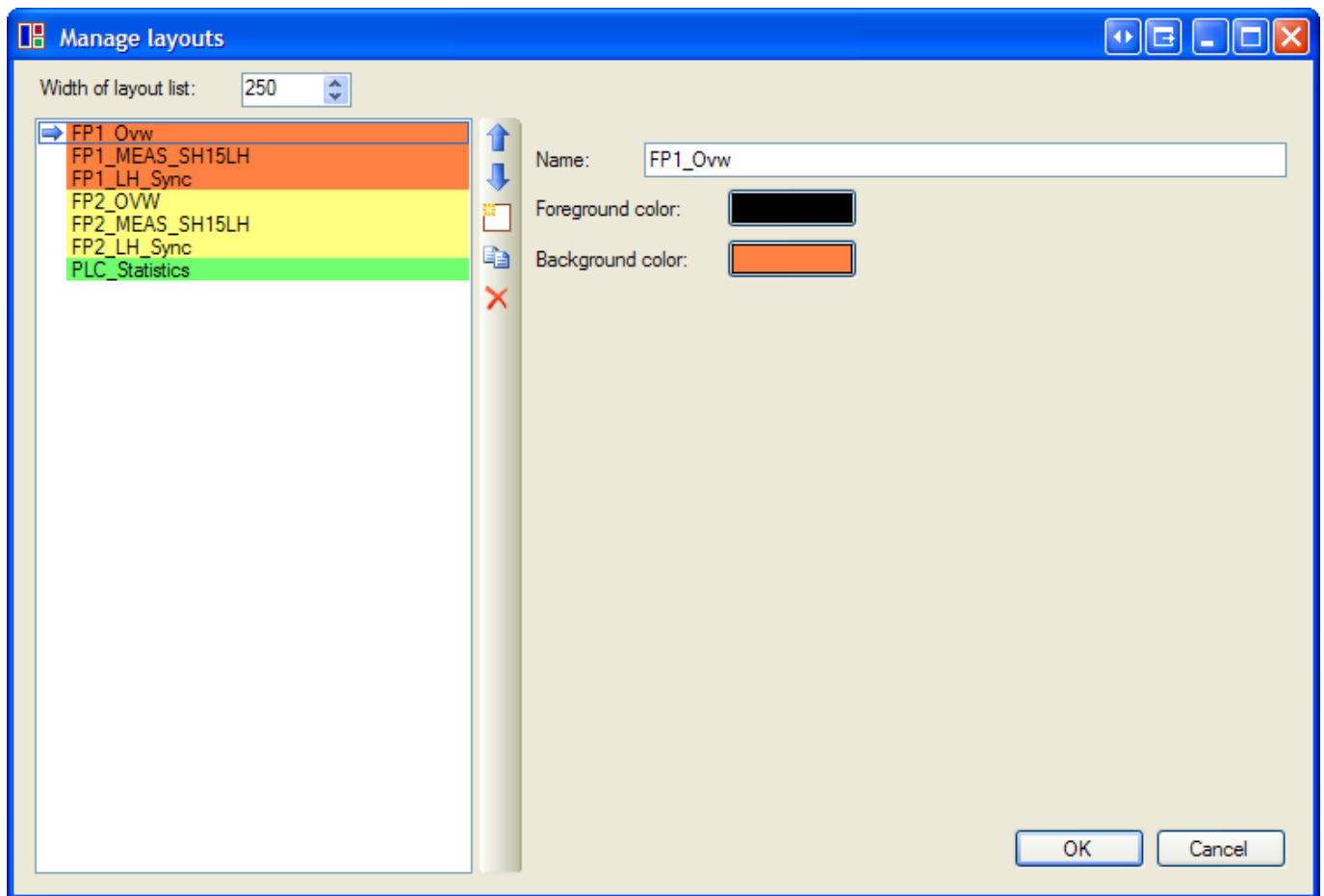


Functions in pda can now take string arguments. At the moment there is only one built-in function that uses text arguments: TextCompare. This function compares 2 texts alphabetically. String arguments are placed between double quotes ". You can use dynamic text by selecting a technostring section. It will be added between square brackets [] to indicate a dynamic text. You can also enter constant text by just typing the text between the double quotes.

The plugin interface dll has been extended with an extra interface so that plugin functions can also use string arguments. The details about this extra interface are explained in the updated ibaPda plugin manual.






8 Manage layouts

There is a new dialog that allows you to manage the layouts. This new dialog replaces the add and remove layout buttons and menu items.

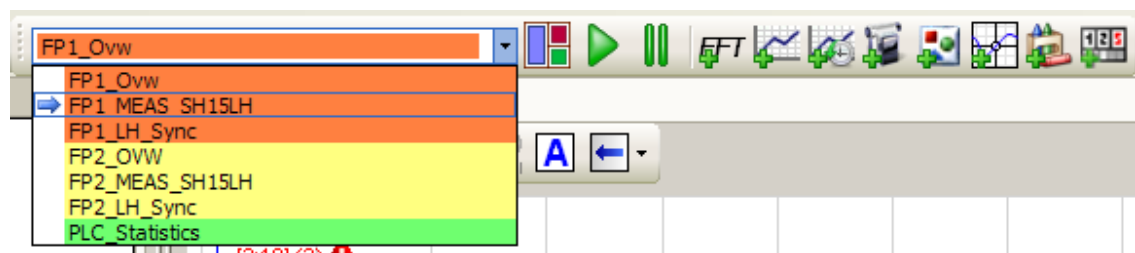


You can specify the width of the layout list on the toolbar. You can do this by entering the width in pixels in the numeric edit field or by dragging the right edge of the layout list. When you select a layout you can change its name on the right. You can also change the foreground and background color that will be used for this layout in the list. The list supports multi select so you change the colors of multiple layouts at once.

The toolbar buttons have the following function:

- : Move the selected layouts up
- : Move the selected layouts down
- : Create a new empty layout
- : Copy the selected layouts
- : Delete the selected layouts

The changes to the layouts will be applied when you click OK.

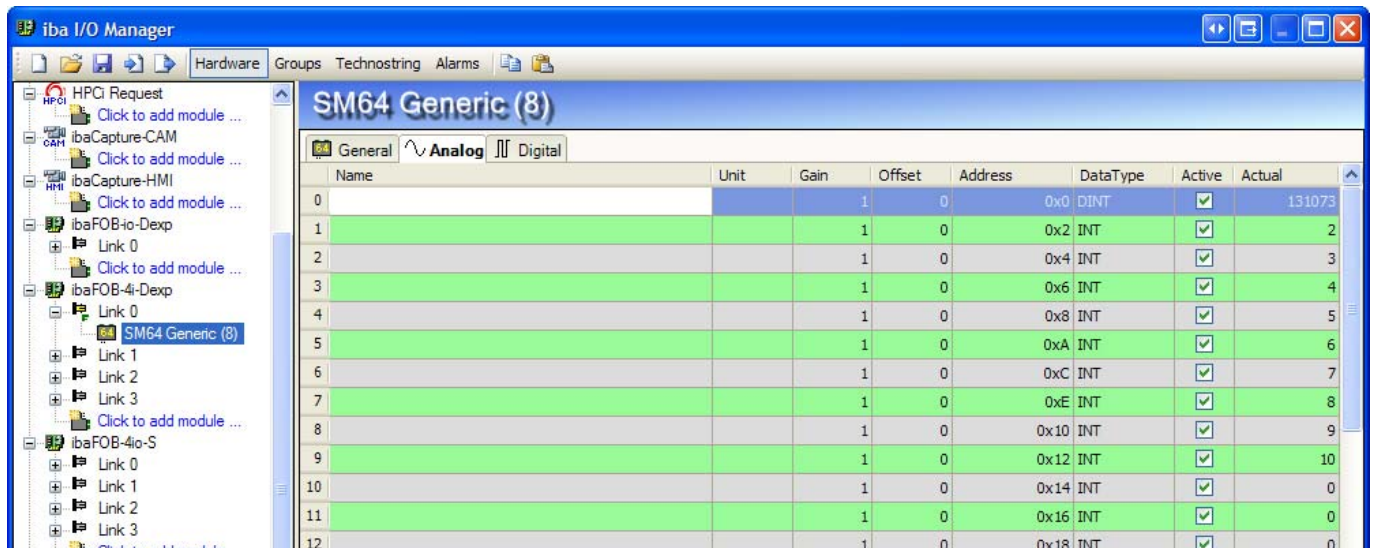


9 SM64 generic module

The SM64 generic module is a module that can be used to measure data on a 3.3Mbit/s FO link on FOB-S, FOB-X and FOB-D boards. The difference with a normal SM64 module is that you can determine for each byte of the 3.3Mbit/s message what it means. Therefor you can specify the number of analog and digital signals.



The 3.3Mbit/s message contains 136 bytes in integer mode and 264 bytes in real mode. The address range for the analog signals goes from 0 to 0xFF (255). The address range for the digital signals goes from 0x100 (256) to 0x107 (263). For each analog signal you have to define the address within the 3.3Mbit/s message and the datatype. For each digital signal you have to specify the address within the 3.3 Mbit/s message and the bit number (0 – 15). You are allowed to define digital signals in the analog address range and analog signals in the digital address range.



iba I/O Manager

Hardware Groups Technostoring Alarms

SM64 Generic (8)

General Analog Digital

Name	Address	Bit no.	Active	Actual
0	0x100	0	<input checked="" type="checkbox"/>	0
1	0x100	1	<input checked="" type="checkbox"/>	0
2	0x100	2	<input checked="" type="checkbox"/>	0
3	0x100	3	<input checked="" type="checkbox"/>	0
4	0x100	4	<input checked="" type="checkbox"/>	1
5	0x100	5	<input checked="" type="checkbox"/>	0
6	0x100	6	<input checked="" type="checkbox"/>	0
7	0x100	7	<input checked="" type="checkbox"/>	0
8	0x100	8	<input checked="" type="checkbox"/>	0
9	0x100	9	<input checked="" type="checkbox"/>	0
10	0x100	10	<input checked="" type="checkbox"/>	0
11	0x100	11	<input checked="" type="checkbox"/>	0
12	0x8	0	<input checked="" type="checkbox"/>	1