

IbaAnalyzer 5.10.0 new functionality description

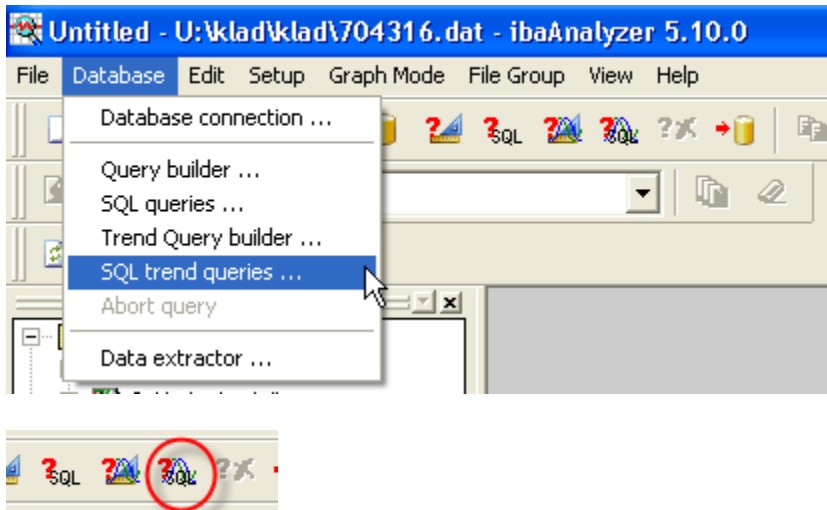
Trend Queries

General

You can query the *infofields* and/or *computed columns* in the file tables created by IbaAnalyzer to display a trend graph of that data. In principle also data from other database tables or views can be queried provided that these tables contain a timestamp column and numeric data. The results of such a query are signals with non equidistant data samples in it, similar as the output of the *XY function*. For each record in the query result a sample is added to the signal at the time indicated by the timestamp field and with the value of the numeric field. You can use these generated signals in further computations, but then they are resampled through linear interpolation to equidistant signals before being used in the given expression. The sample distance of the resampled signal will be the time interval between the closest two sample points in the non equidistant original signal.

SQL Trend queries

In the SQL Trend query dialog you can type an SQL instruction to be executed for fetching the trend graph data. You can access this dialog either from the database menu or by clicking the toolbar icon.

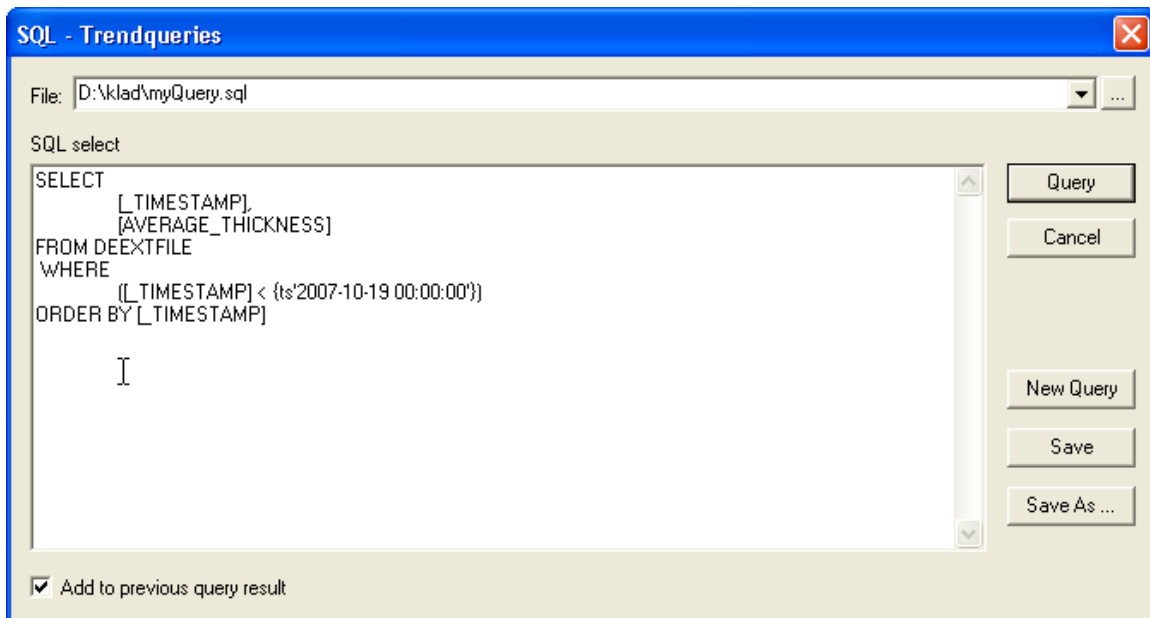


You can load a previously saved text file containing the instruction or save the query instruction you are currently using to a text file for further use similar as you did for an ordinary ibaAnalyzer file query. The instruction you are typing here should be valid SQL and should be valid for the database you are using. Also when executed, the instruction should have a result set with a timestamp field (you can have more than one, but only the first one will be referenced) and at least one numeric field. The instruction should also contain an ORDER BY clause on the timestamp field.

On executing the instruction by clicking the '*Query*' button, each numeric field in the result set will generate a signal with its name the same as the fieldname. The signal will be added to the signal tree under the '*Trendquery results*' node.

Except the first timestamp field, all non-numeric fields in the result set of the SQL instruction will be ignored.

If you check the checkbox labeled '*Add to previous query result*' the new signals will be added after any signals previously generated by trend queries under the '*Trendquery results*' node, otherwise any previously generated trend queries will be removed from memory and cleared in the signal tree before adding the new signals.



Trend query builder

Similar as the file database query builder, a *trend query builder* is available to assist you in creating your SQL instruction to perform a trend query.

You can access this dialog either from the database menu or by clicking the toolbar icon.

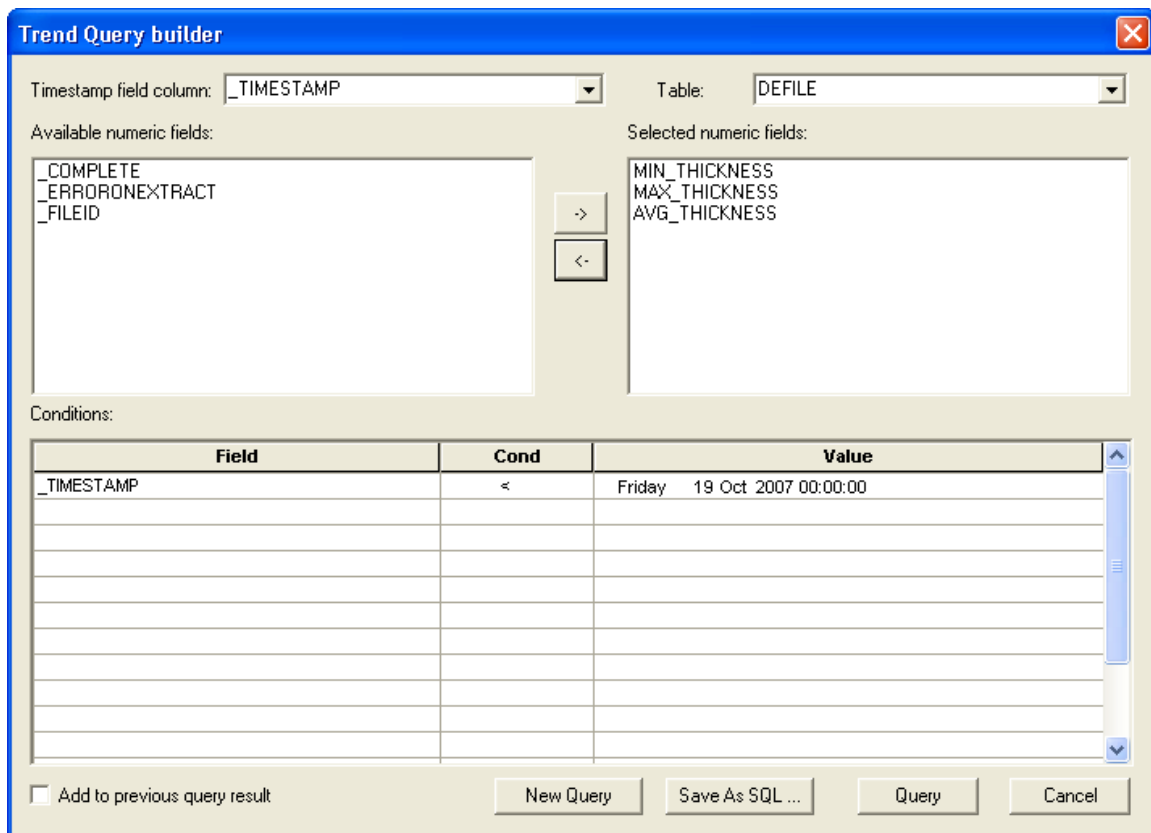


You can select the table you want to query, which field will serve as timestamp field and which numeric fields you want to query. You can also add conditions on the query similar as in the file database query builder.

You can choose to store the resulting SQL instruction in a text file by selecting the 'Save as SQL' button, for example to open it in the SQL trend query dialog.

You can execute the resulting SQL instruction by clicking the 'Query' button.

Again you can either check or uncheck the checkbox labeled 'Add to previous query result' to have the new signals either added to or replacing signals previously generated by trend queries.



Trend Query builder

Timestamp field column: Table:

Available numeric fields:

- _COMPLETE
- _ERRORONEXTRACT
- _FILEID

Selected numeric fields:

- MIN_THICKNESS
- MAX_THICKNESS
- AVG_THICKNESS

Conditions:

Field	Cond	Value
_TIMESTAMP	<	Friday 19 Oct 2007 00:00:00

☐ Add to previous query result

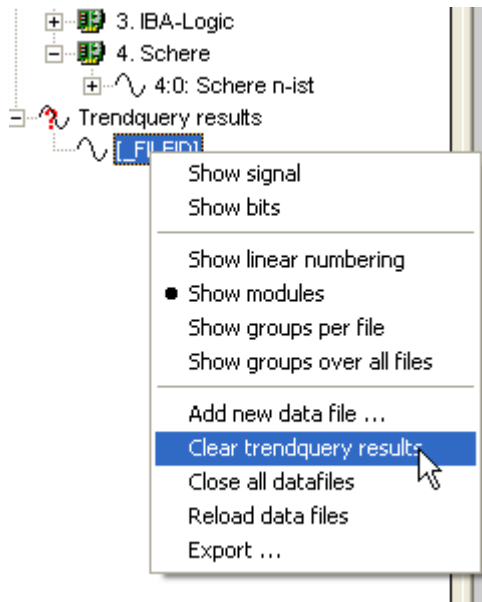
New Query Save As SQL ... Query Cancel

Signal tree

The generated signals for the queries are available in the signal tree under the 'Trendquery results' node. From here you can drag or double-click them to display them in the recorderview.



If you right click on a query result or the 'Trendquery results' node, you can remove the trendquery results from the signaltree and memory by selecting 'Clear trendquery results' in the contextmenu. The query results also get cleared by selecting 'Close all datafiles' from the signaltree context menu or main menu.



Command line switch

After having saved the SQL instruction of either the trend query builder or the SQL trend query dialog to a text file, you can start ibaAnalyzer with the query through the command line similarly you can start ibaAnalyzer with a file table query:

```
..\ibaAnalyzer.exe /trendsql:myFile.sql
```

Where myFile.sql is the saved SQL instruction

DB2 support

There is support for extracting to and querying from IBM DB2 UDB databases through Oledb. You can select the DB2-UDB provider from the list in the database setup dialog.

The screenshot shows the 'Database connection' dialog box. The 'Database login info' section contains the following fields:

- Database provider: DB2-UDB (selected from a dropdown menu)
- Database name: Sql-server, ODBC-database, Oracle, DB2-UDB (selected from a dropdown menu, circled in red)
- Authentication: DB2-UDB (selected from a dropdown menu, circled in red)
- Computer: Local machine (selected with a radio button)
- Specify authentication info: Username: db2admin, Password: [masked]
- Test database connection button

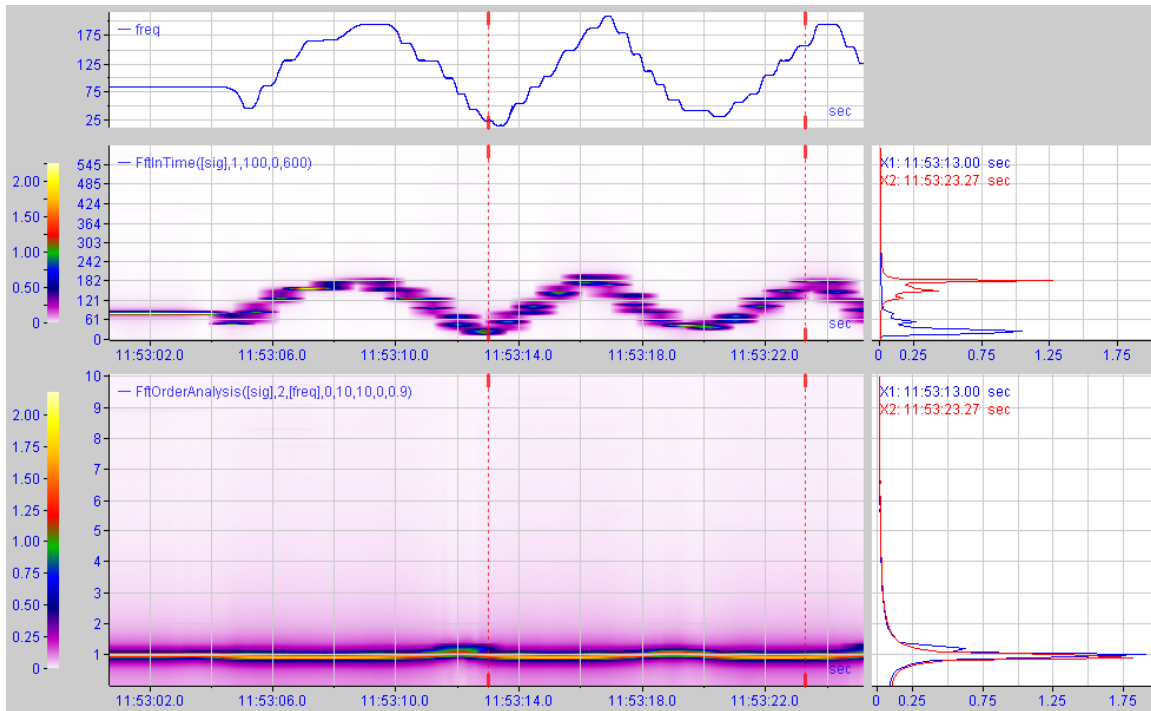
The 'Table names' section contains the following fields:

- File header: deFile
- Channel header: deChannel
- Segments: deSegment

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

Order Analysis

The new ibaAnalyzer *FFTOrderAnalysis* function works like the *FFTInTime* function; i.e. it returns FFTs of a signal varying in time. The difference is that instead of returning an FFT array with elements for given frequencies, it returns an FFT array with elements for given orders, where an order corresponds to a multiple of a reference frequency. For an ordinary *FFTInTime* function, if a rotational part of machinery speeds up or slows down, the peaks of the FFT will move respectively to the right and left, while for an *orderanalysis* the peaks will not move provided the change in speed has no other impact on the measured signal.



The parameters of the *FFTOrderAnalysis* function are:

- **Expression:** the expression to take the FFT of.
- **Time:** This parameter multiplied with the overlap factor determines the time (or length) interval at which a new FFT should be calculated each time. This parameter multiplied with the sample rate of 'Expression' also determines roughly the amount of samples that are taken into account for generating the FFT function.
- **Frequency:** The reference frequency signal corresponding with the first order (in Hz or 1/m)
- **MinOrder:** The minimal order that should be displayed, this is expected to be an integer value and will be rounded if not so.
- **MaxOrder:** The maximum order that should be displayed, this is expected to be an integer value and will be rounded if not so.
- **Order subdivisions:** You can display suborders if you wish. The number of subdivisions is actually one less than this parameter so that if this parameter is

for example x , then every $1/x$ order there is an FFT sample. This parameter can be omitted and has then as default value 1. The parameter is expected to be an integer and will be rounded if not so.

- Window: As with all FFT functions you specify a window function that is multiplied with the data before taking the FFT (rectangular or no window = 0, Bartlett = 1, Hamming = 2, Hanning = 3, Kaiser = 4). This parameter can be omitted and has a default value of 0 (no window).
- Overlap: parameter that determines how much the data for calculating each individual FFT overlaps, this is a value between 0 and 1 where 0 = no overlapping and 1 = complete overlapping. This parameter can be omitted and has a default value of 0 (no overlapping).

The order analysis is calculated by interpolating sample points at such locations so that the positions in the resulting FFT array correspond with the requested orders and suborders.

The function does not return values for a given time point if

- It would take more samples than there are available in an interval twice the size of the parameter '*Time*'.
You can remedy this by increasing the '*Time*' parameter provided that the '*Frequency*' and '*Expression*' signals do not change too drastically so that the FFTs are still meaningful taken over the increased '*Time*' interval.
Also decreasing the '*Order subdivisions*' parameter can remedy this problem.
- More than four points need to be interpolated between the same two sample points. You can remedy this by decreasing the '*MaxOrder*' parameter