New Features in ibaLogic v5.6.0

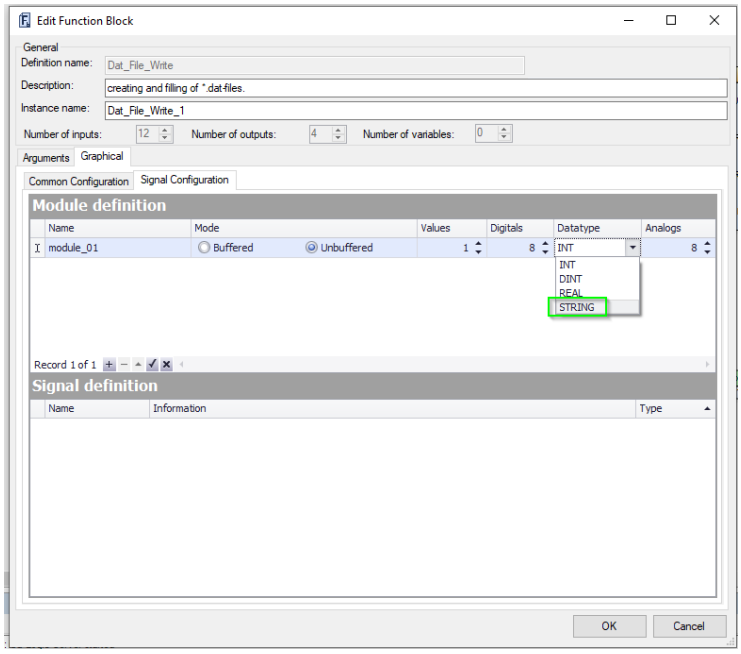
Author: iba AG Fürth

Date: 22/10/2021

Content

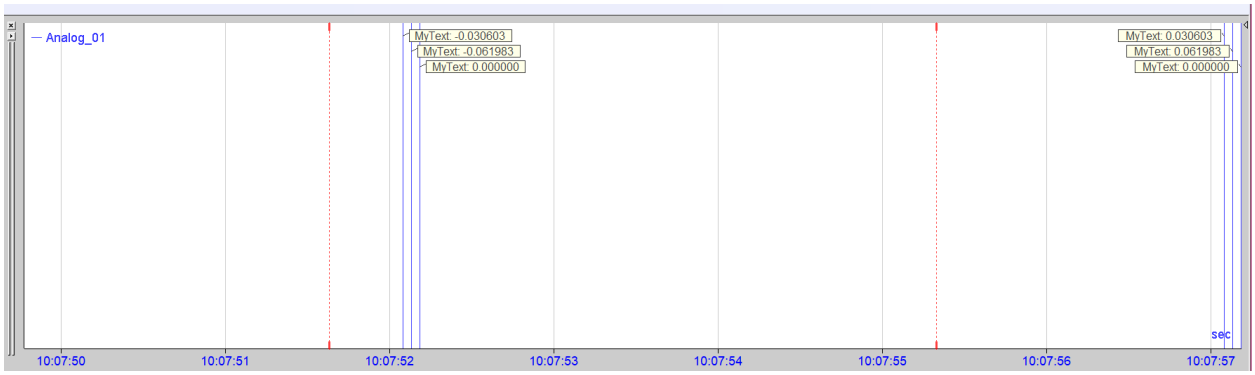
1 DatFileWrite with text channels

The DatFileWrite block now also has the capability to write text channels.
The selection of data types has been expanded for this purpose:



Attention: This is only possible for unbuffered values. This is not permitted for buffered signals and a corresponding message is displayed.

In ibaAnalyzer, a text channel can look like this. A signal flag is always displayed when the text has changed.



2 Password-protected function blocks (FBs) and macros (MBs)

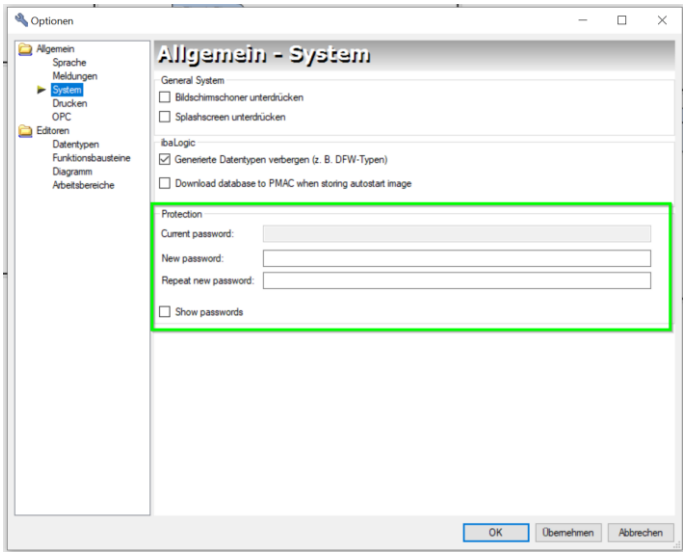
To protect the content of your own FBs and MBs, ibaLogic now offers the possibility to create a password protection.

This password is valid within a workspace.

With protected blocks, only the interface area (inputs/outputs) is visible, but the program part is no longer visible. In the case of a MB, the graphical content cannot be opened.

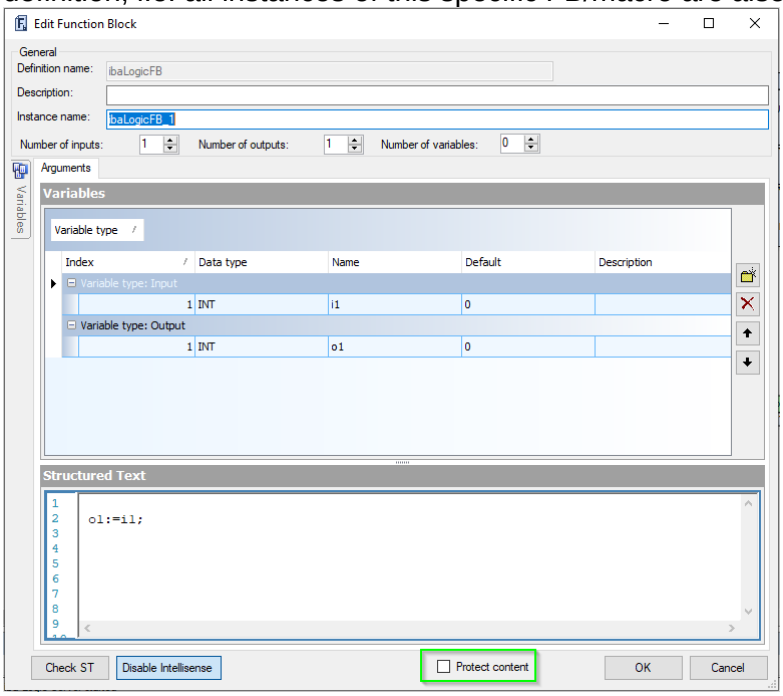
However, a protected module (instance and definition) can be deleted at any time without asking for the password.

To activate the protection, go to CONFIGURATION-OPTIONS-SYSTEM. Here it is possible to enter a password for the protection.

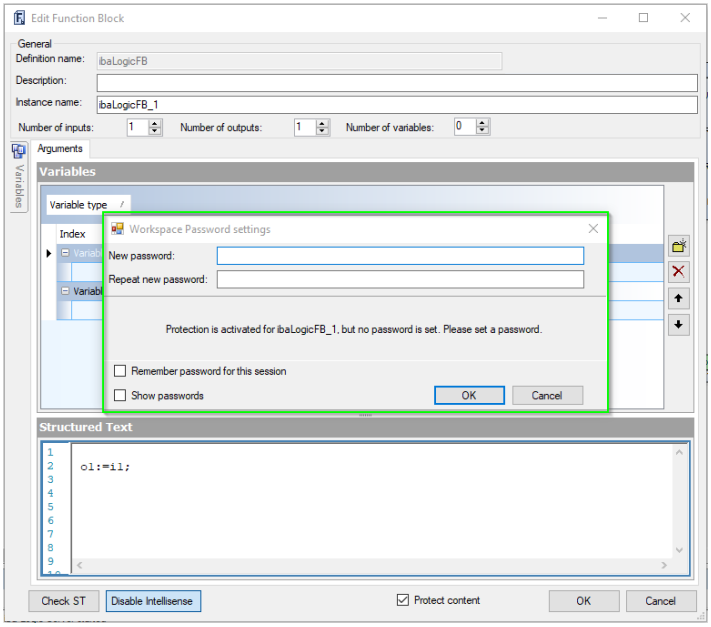


To deactivate the password protection, change the password to an empty one.

In every FB or macro there is the possibility to protect it. The protection always refers to the definition, i.e. all instances of this specific FB/macro are also protected.

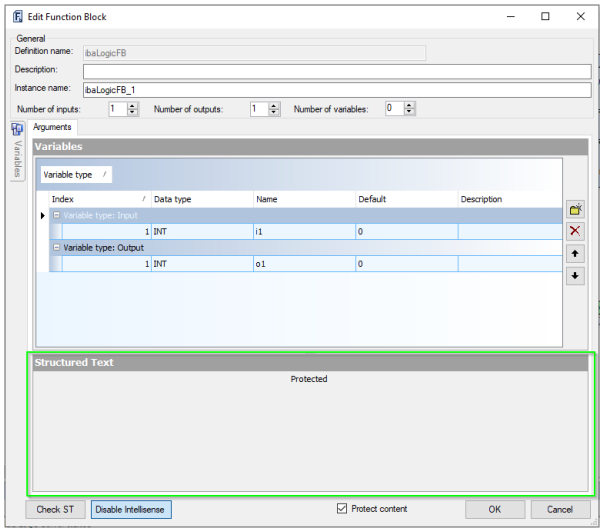


If you select this option and have not yet set a password for the workspace, you will be prompted to enter a password.



You can also decide whether the password should be remembered for the entire session. This means that as long as you do not close the client or the current workspace, you can edit all password-protected blocks.

A protected FB looks like this:



The content can no longer be seen.

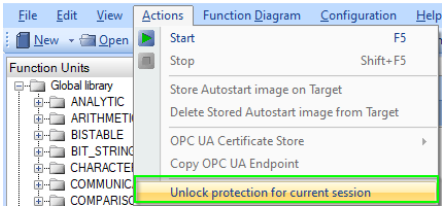
If a macro is protected, you can no longer access the internal macro plan.
If the protection of a macro is activated, any open macro contents (separate program tabs) will be closed.

If you want to edit the module again, you have several options:

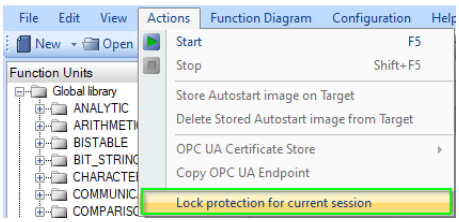
- You enter the password and you can edit the block. If you close the FB block again then the protection is immediately activated again. For the MB, you have to close the tab with the macro content so that the macro is protected again.

- You remove the protection from the block and set the protection again after editing. To do this, the password must be entered each time.

- A possibility to remove the protection for an editing session in general can be reached by the following menu:



The password must be entered once and now all protected FBs or MBs can be edited.
If you want to activate this protection in general, you can select this menu item again.



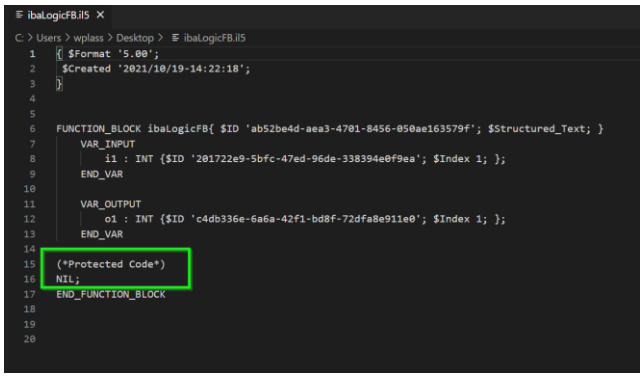
When selected, the protection is immediately reactivated.

- Also, when the client is closed, the session is terminated, and thus the protection will be active again the next time the client will be opened.

When exporting protected modules, or programs or projects that contain these modules, please note that they will only be exported completely if the password is known.

If you import such an export, it is not completely executable because it will be missing the protected contents.

Export example:



3 Playback changes (variable speed, playback of length-based signals)

Some innovations have been introduced in the playback handling.
Since previous versions it is already possible to play different playback files (of the same type) in a controlled way. I.e. read in, start, pause, etc.

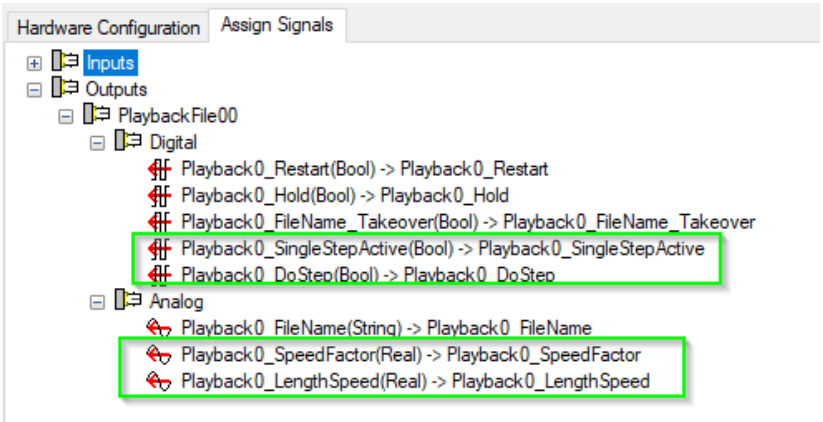
The following functions and changes have now been added:

- **Expressions contained in Dat-files** (= signals generated with the ibaAnalyzer) are now also
  available and can be used with the standard signals.
  Attention: This is not possible for the PADU-S-IT2x16.
- **Playback of length-based signals**
  Up to now, only time-based data files could be played. Now it is also possible to play back length-based files with the help of a speed signal.

  Please note that all length-based signals in a dat file can only be played back by a single speed signal. If there are several "length measuring points" in the file, each with its own speed, you have to choose one measuring point and its length signals.
- **Playback of playback files with variable speed**
  This makes it possible, for example, to synchronise a playback file with a current process by playing the playback slower or faster.
- **Playback in single-step mode = next measured value from data file**
  With this function it is possible to read out all measured values one after the other in order to carry out calculations with every sample in the data file. Or to carry out simulations more quickly with every sample. E.g. play back hourly values / daily values in a 100ms cycle.

For these functions, the inputs/outputs of the playback have been extended.

Outputs:



The marked outputs are new:

SingleStepActive: False - Dat-File will be played in a normal way
True - next sample from the dat file will be read when DoStep toggles

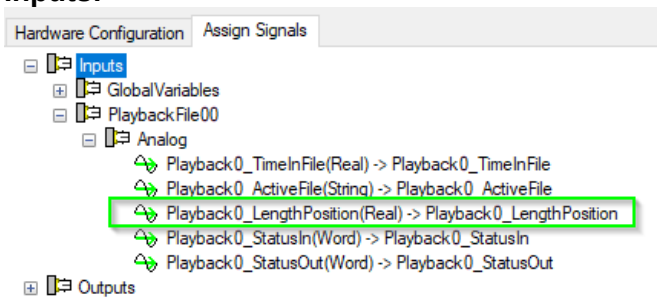
DoStep: Toggle advances the playback by one step.
1.Toggle after a Playback Restart jumps to position 0 !

SpeedFactor: 1 = normal playback of all time-based signals taken i.e. 1x speed. Other factors accordingly.

LengthSpeed: For the time-correct playback of length-based signals, the speed signal (or a static speed) must be set here.

Note: For SpeedFactor and LenghtSpeed, a 0 is internally set to a factor of 1 at the start. If a 0 is written by the running program, the last value remains internally.

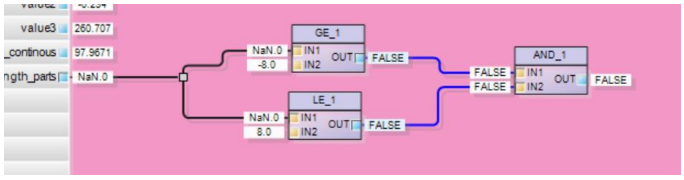
Inputs:



LengthPosition: Current length position when playing back length-based signals

The parameterisation for the individual cases:

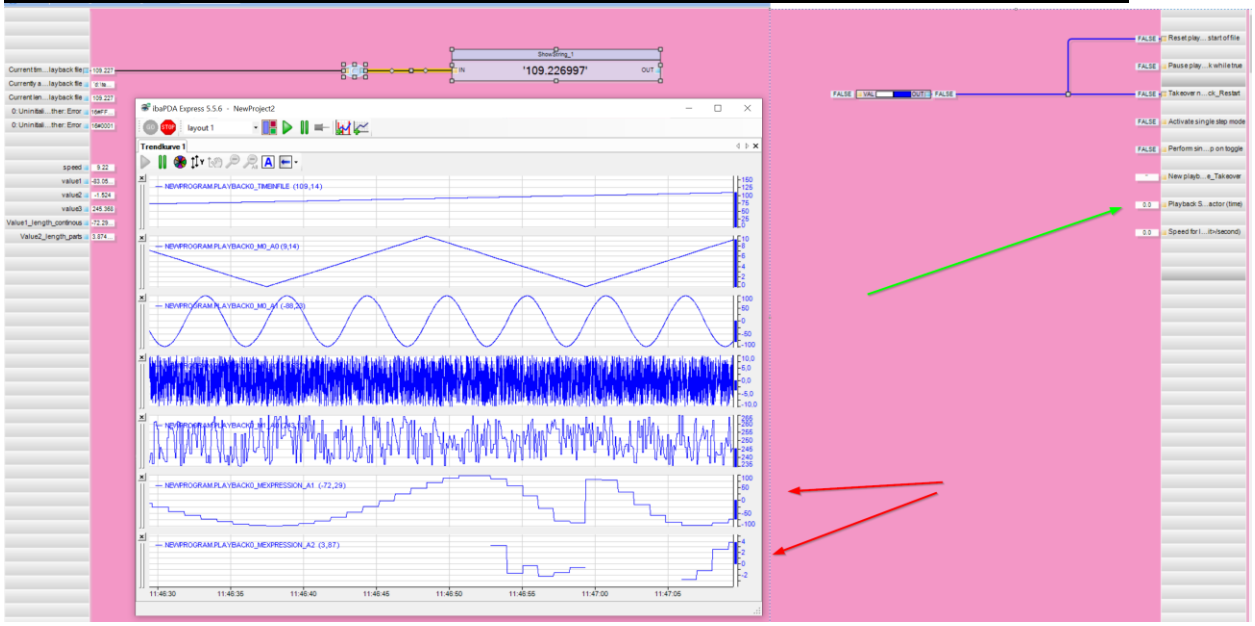
For the playback of time-based or length-based signals within a dat file, it should be noted that these each have their own independent parameterisation.
It may be that, for example, all the time-based data has already been played to the end, while the length-based signals have not yet been completed. In this case, no measured value would be returned for the time-based signals (NaN = not a number). The same applies if there are gaps in the signal.
If someone wants to query these gaps, this can only be done by querying whether the signal is in the expected range.



Note: For INTEGER values, a 0 stat NaN is currently returned for internal reasons.

Here are examples of the new functions

Example1: Simple playback of signals (time-based and length-based signals available)



There is no speed factor set(green arrow), so the data file is played in its original speed. The internal speed factor is 1. Any other factor influences the playback speed, e.g. 2 = 2 times the speed, 0.5 = half the speed, etc.

The length-based signals are also played with an internal length factor of 1 (red arrows). Usually this references a speed of 1m/s, according to the recording setup. The length signals with gaps are also played back correctly.

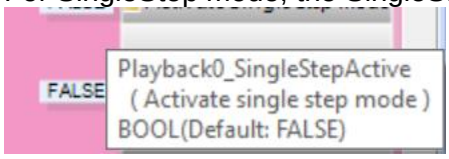
In these gaps, the lower length signal in the graph has the value 

If you want to play back the length-based signal with the correct time, the time-based speed signal on which the length conversion was based must be contained in the same dat file. Then you can send this signal directly to the LengthSpeed output. You can then run it faster and slower via a multiplied factor.

If you use the speed factor setting for anything other than 1, you should multiply the length factor with the speed factor so the length progress is kept in sync.

Example2: SingleStep for timebases signals

For SingleStep mode, the SingleStepActive output must be set to TRUE.



In this case, a value change at the DoStep output causes the playback to advance by the timebase of the IO-configuration. To jump directly to the next sample, the output speed factor must be configured accordingly.

The speed factor is calculated from the ibaLogic timebase in the IO-configuration and the sampling rate of the signal.

Speed factor = sampling rate in ms / timebase in ms

Examples:

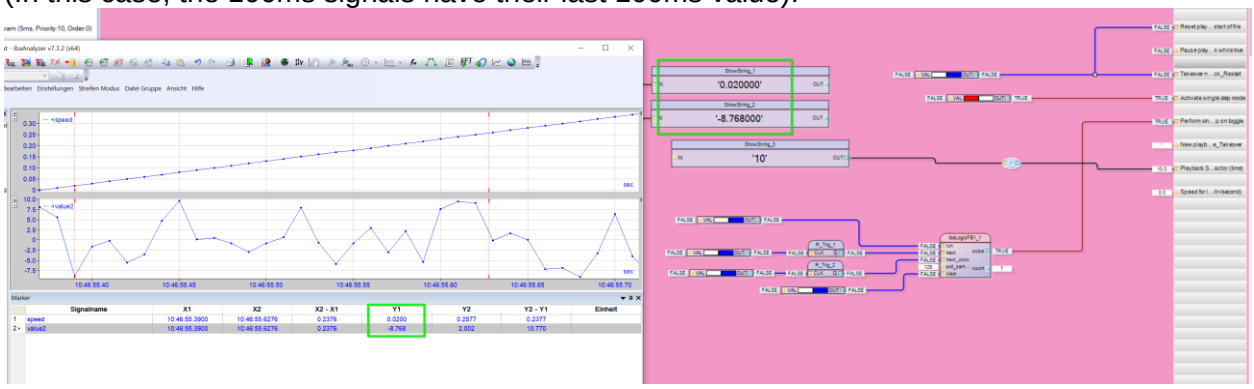
Sampling rate 10ms timebase 1ms speed factor = 10
Sampling rate 10ms timebase 5ms speed factor = 2

Example: The data file contains 50ms and 100ms signals. You can either query the measured values of the 50ms signals one after the other or those of the 100ms signals.

If you want the 50ms signals, set the value 0.05 at the output speed factor.

With each toggle you get the next value. The current time in the DatFile is also reported in the TimeInFile input.

(In this case, the 100ms signals have their last 100ms value).



Here, 10ms values with a timebase of 1ms are queried. For each toggle, the data file position is advanced 10ms and the appropriate values are set to the inputs.
(Note: the function block in the layout is only used to make a toggle per keystroke or to automatically "toggle through" until a set time is reached).

Example3: Singelstep for lengthbased signals

The same applies here as in example 2.

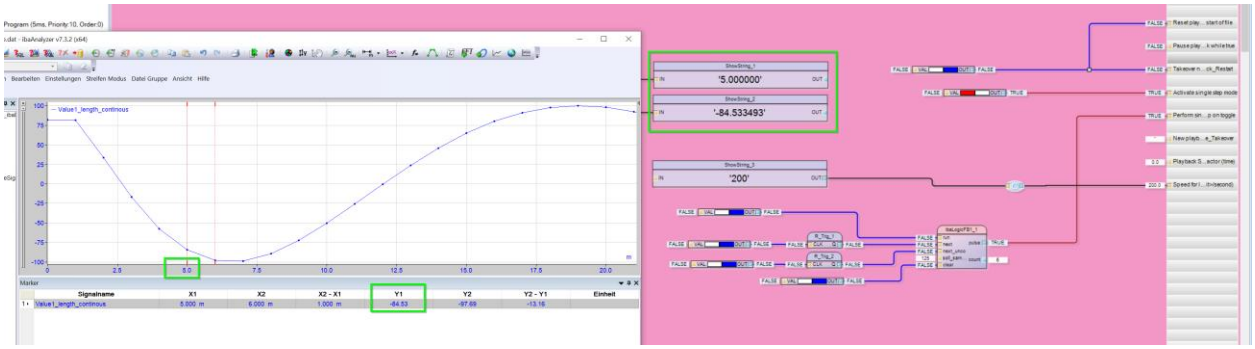
But instead of the speed factor, the output LenghtSpeed must be set.

This factor is calculated from the ibaLogic timebase in the I/O-configuration and the length resolution of the signal in mm.

LengthSpeed = length resolution in mm/ timebase in ms

Examples:

Length resolution 1m timebase 1ms Speed factor = 1000
Length resolution 1m timebase 5ms speed factor = 200



Here, 1m values with a timebase of 5ms are queried. For each toggle, the data file position is advanced by 1m and the appropriate values are set to the inputs. The current position metre is displayed by the LenghtPosition input.