OPAL-RT TECHNOLOGIES

FROM IMAGINATION TO REAL-TIME

RT-LAB Solution for Real-Time Applications

OP101 : Getting started

RT-LAB Advanced Features

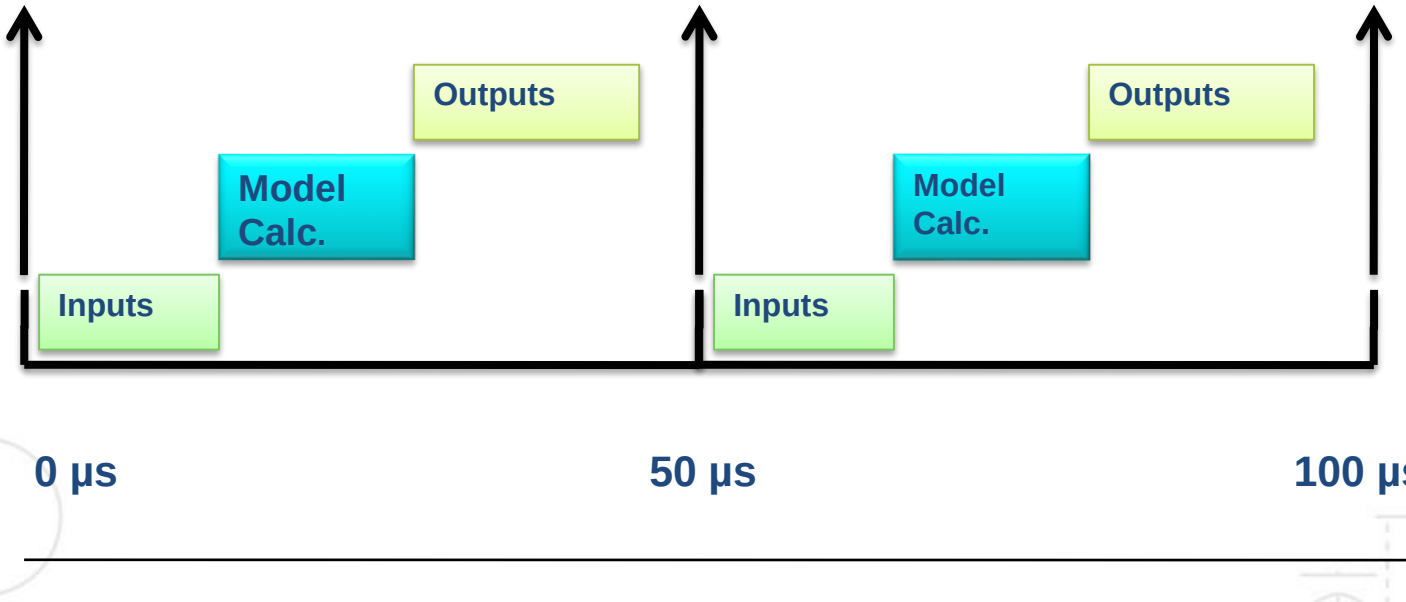
Training Services

Outline

1. ***Model monitoring***
2. Variables table
3. Probe Control
4. Dealing with acquisition data loss
5. Data logging features
6. Dynamics Signal acquisition

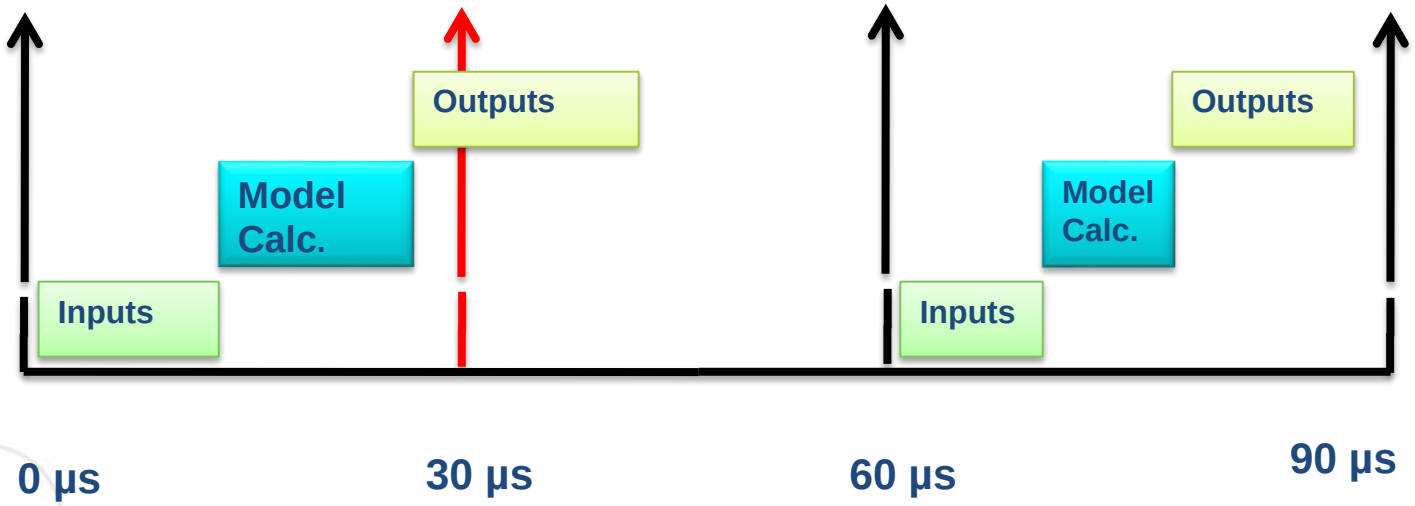
Model monitoring

- Overruns
  - Normal Condition :



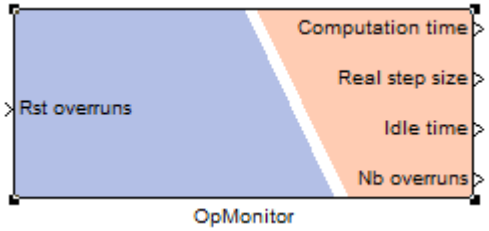
Model monitoring

– Overrun Condition :



Model monitoring

- OpMonitor block
  - Allows you to retrieve timing information from the model
  - Location in Simulink Library : RT-LAB / Monitoring / OpMonitor

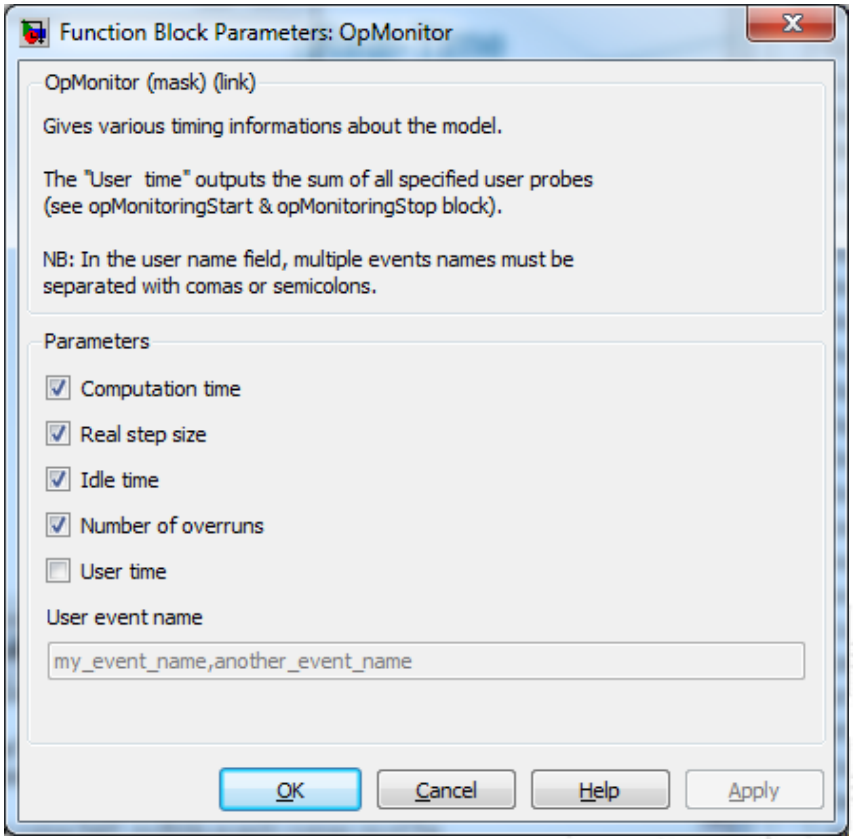


Outputs

- Computation Time
- Real Step Size
- Idle Time
- Number of overruns
- User Time

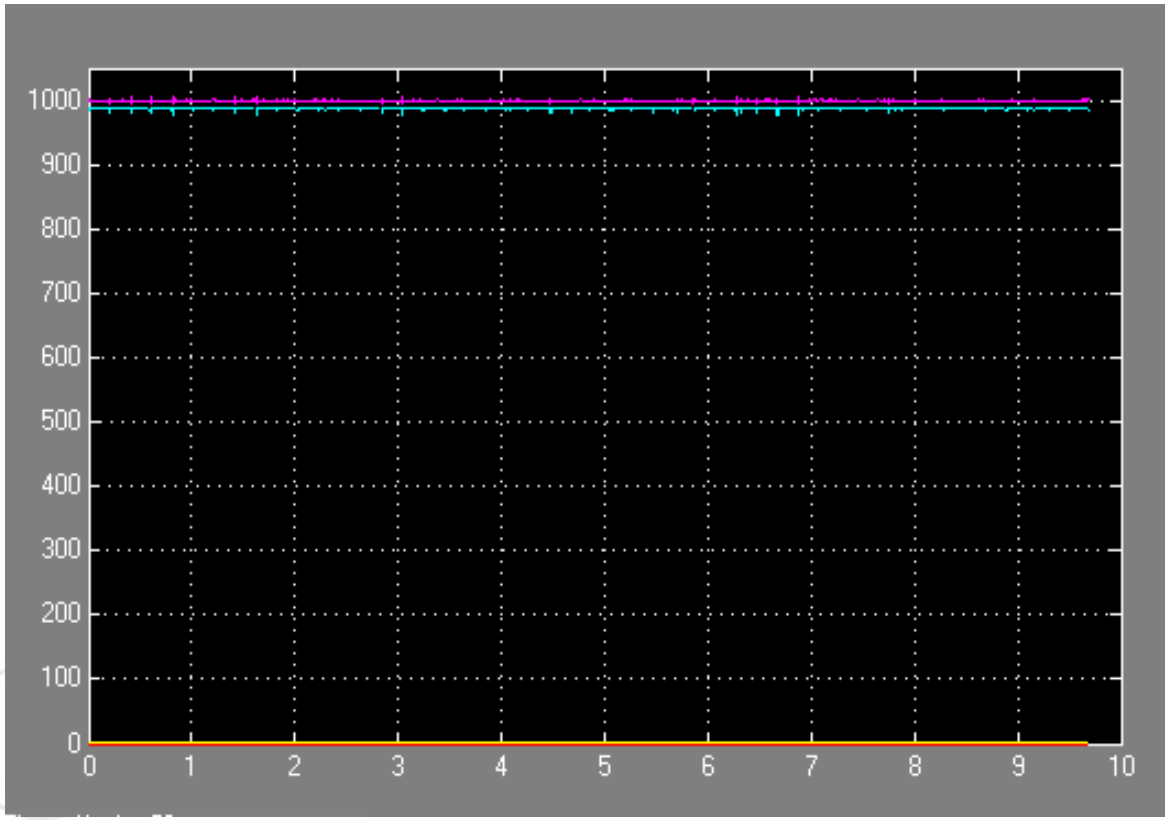
Inputs

- Reset Overruns



Model monitoring

Definition of monitoring values, time in microseconde



Real Step Size

Idle time

Model Idle Time (wait for synchro)

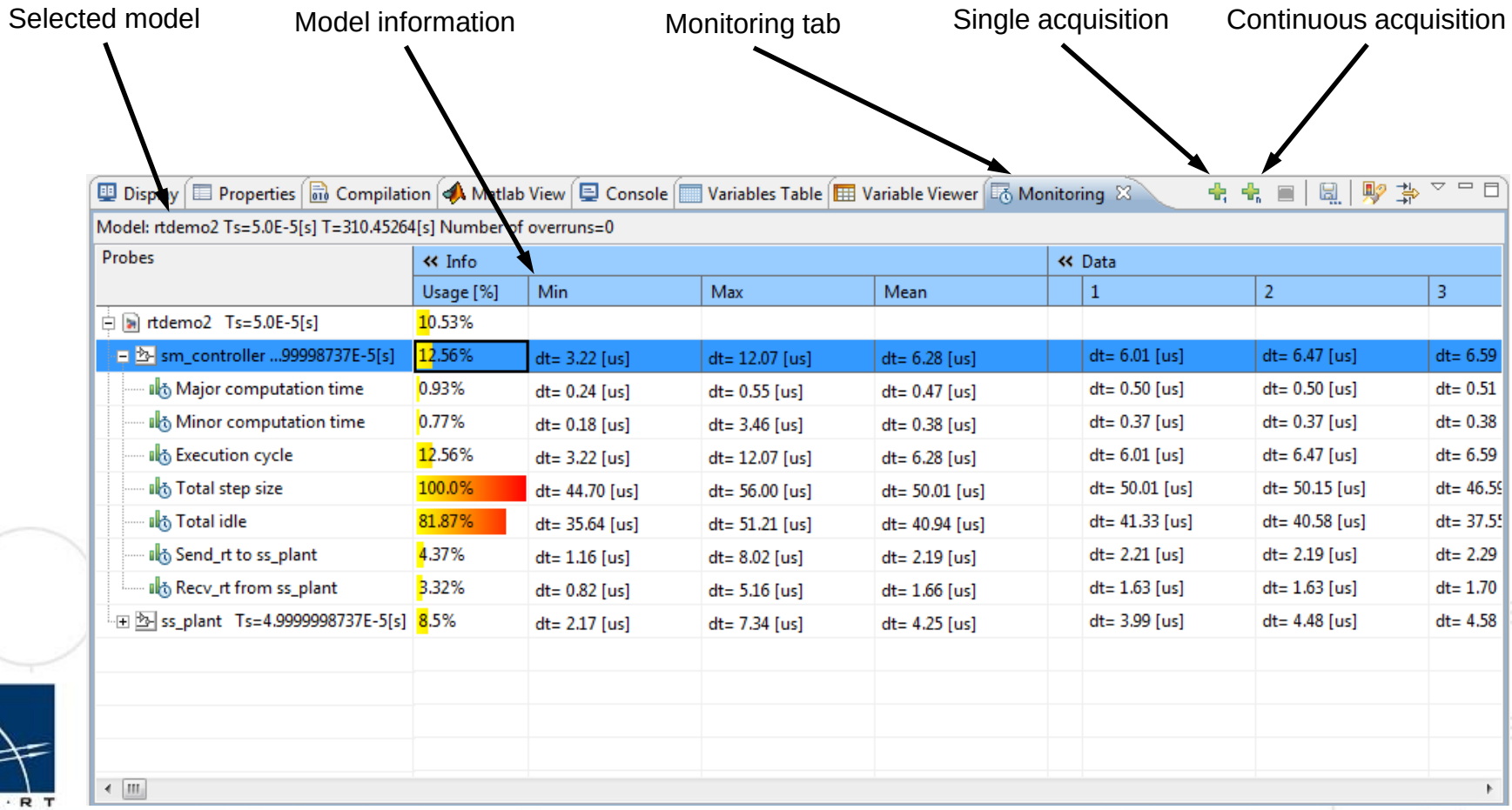
Computation Time

Computation

Number of overruns

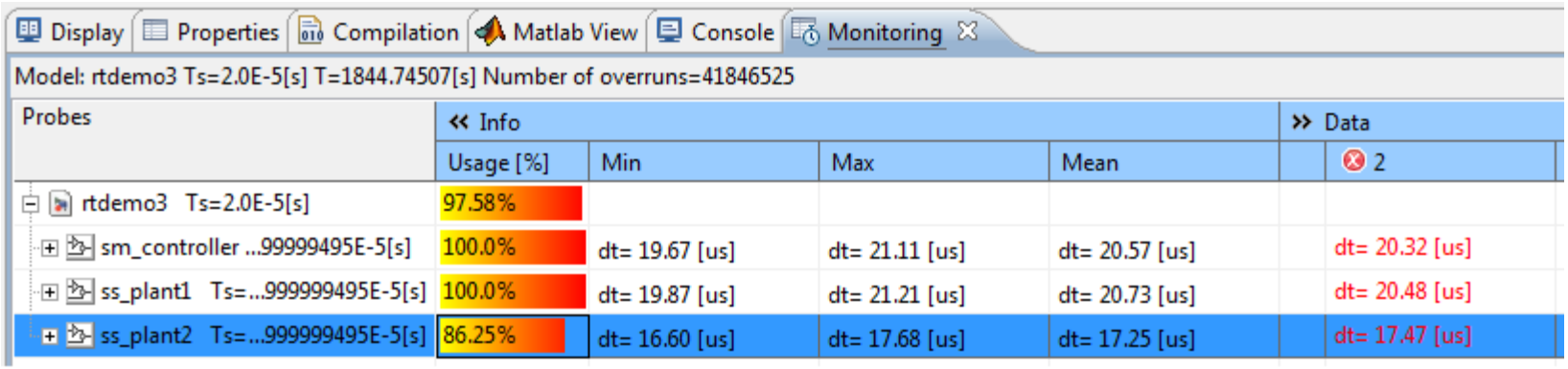
Model monitoring

- RT-LAB Monitoring View (Model must be runnning)
  - Overrruns will be in red

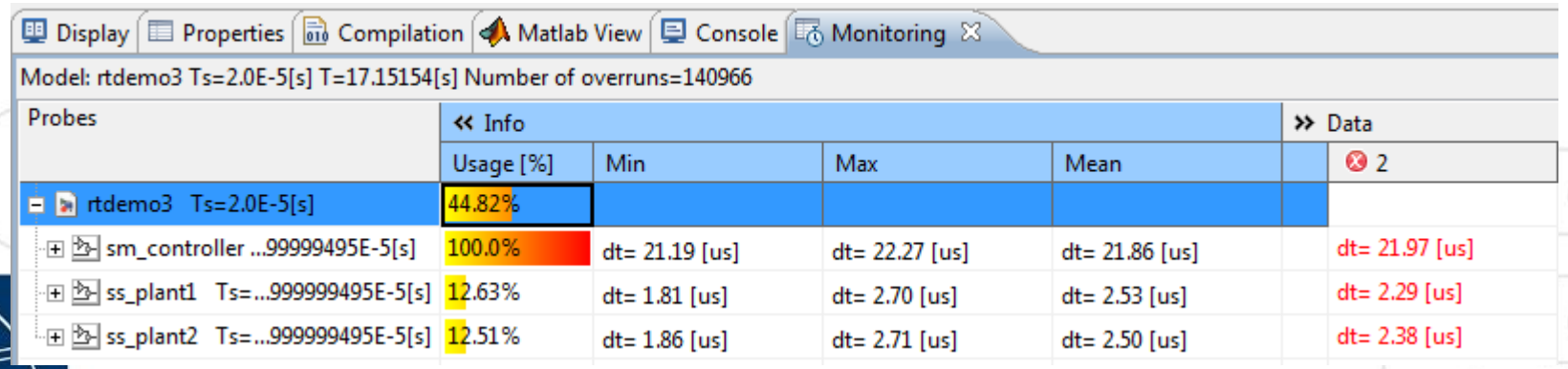


Model monitoring

Monitoring view of overrun for ”ss_plant1” that create an overrun of “sm_controller” (Cascading)



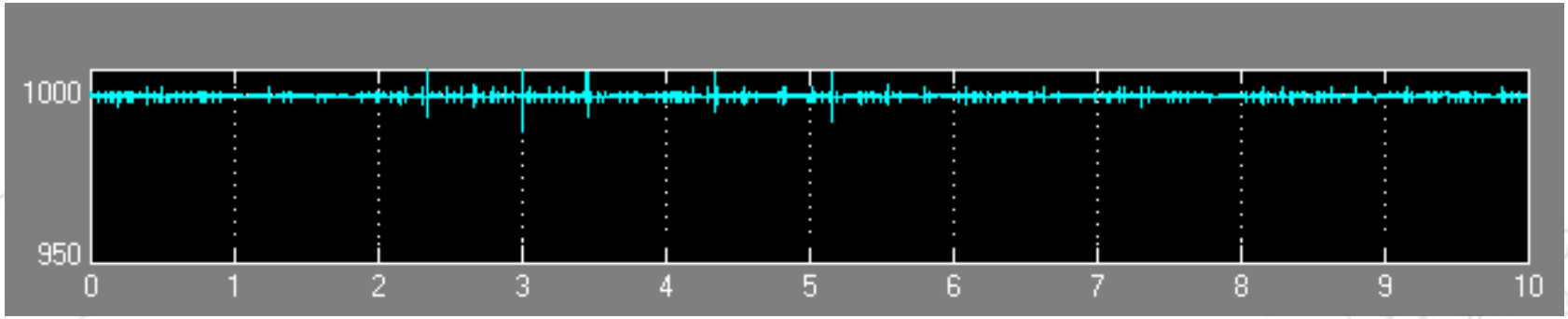
Monitoring view in overrun for ‘’sm_controller’’ only



Model monitoring

- Jitter :

  - Real variation of the step time
  - Good jitter is around 7us (with RT-OS like QNX and Redhat)
  - If model run time is too close of the step time, jitter variation could cause random overrun

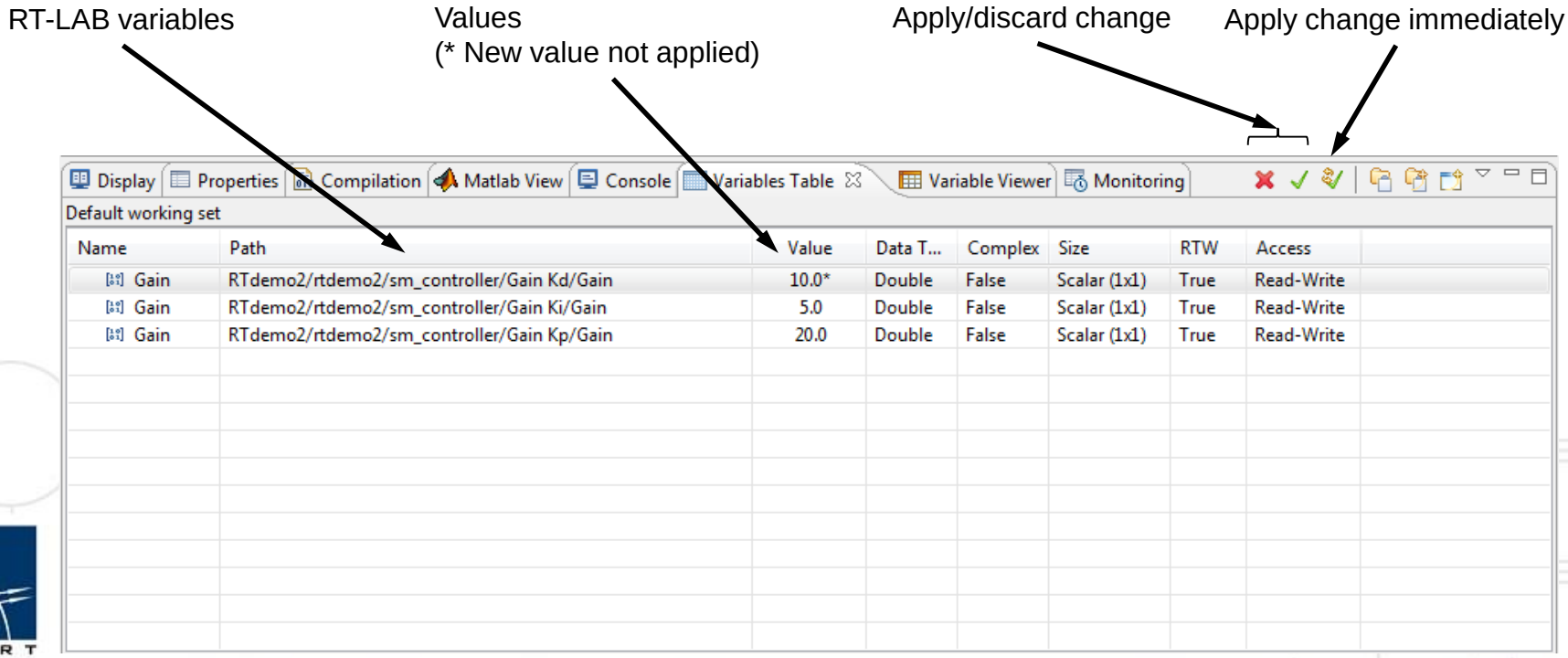


Outline

1. Model monitoring
2. ***Variables table***
3. Probe Control
4. Dealing with acquisition data loss
5. Data logging features
6. Dynamics Signal acquisition

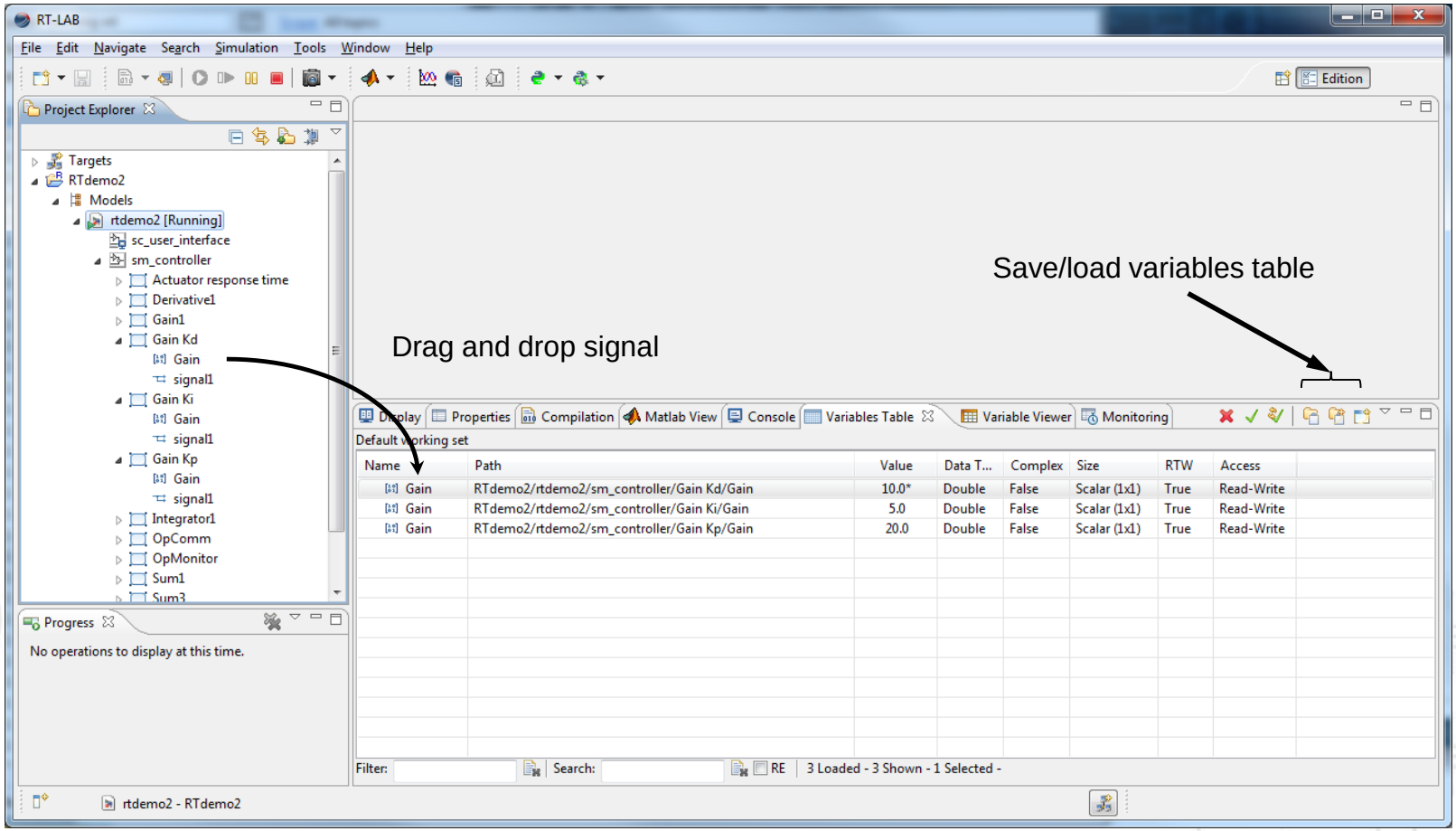
Variables Table

- Display one or more set of RT-LAB variables
- Static or dynamic acquisition
- Change values of a variables in a running model



Variables Table

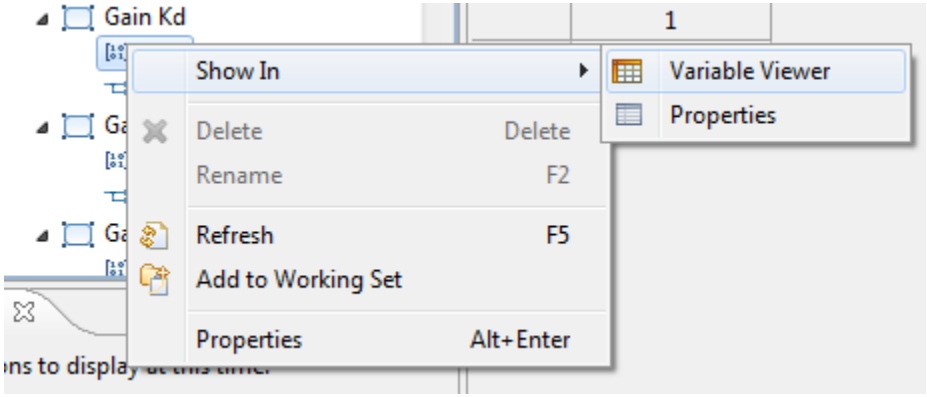
- Drag and drop signal from the model to the variables table
- Variables table could be save as a working set

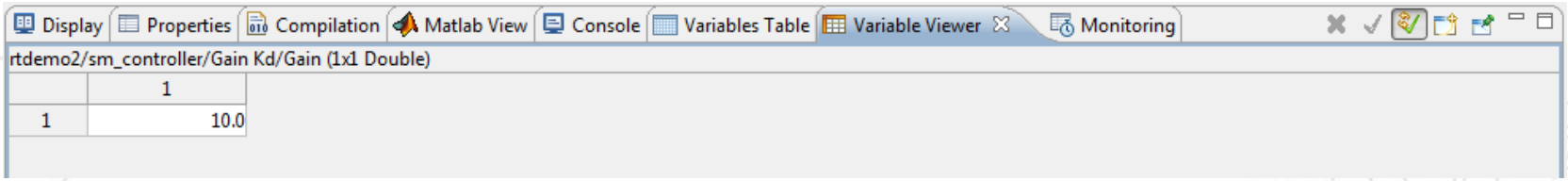


Variables Table

Variable Viewer

- Display selected variables from the explorer
- Right click > Shown In > Variable Viewer



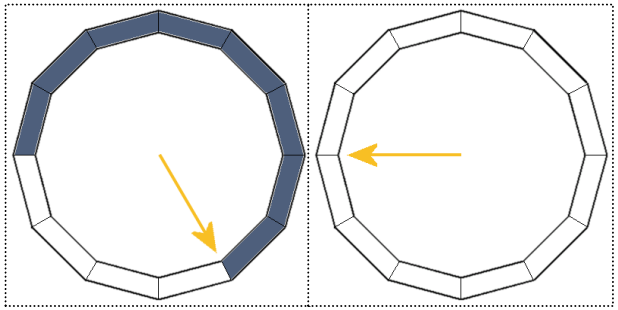


Outline

1. Model monitoring
2. Variables table
3. ***Probe Control***
4. Dealing with acquisition data loss
5. Data logging features
6. Dynamics Signal acquisition

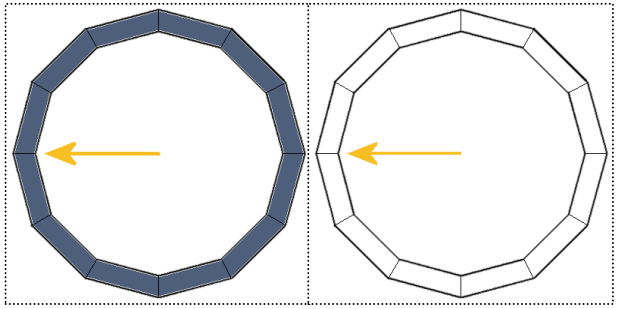
Probe Control

- How does acquisition work : The model starts and the acquisition fills the first buffer with the values collected at every step.



8th step

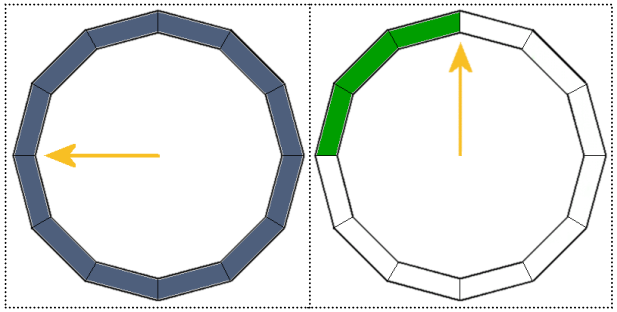
- How does acquisition work : The first frame (in the first buffer) is ready. We signal the *sender* to send, when possible, the first frame to the console.



12th step / 1st frame

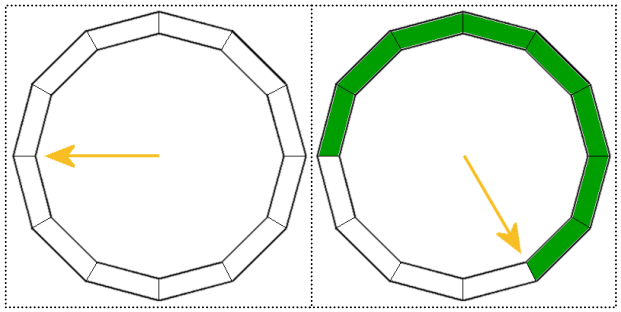
Probe Control

- During that time, the next frame is prepared in the other buffer. The sender hasn't had a chance to run yet.



15th step

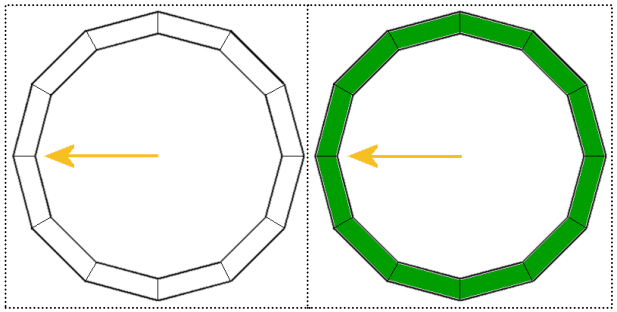
- The *sender* has run in the background and the first frame has been sent. We continue to prepare the second frame in the second buffer.



20th step

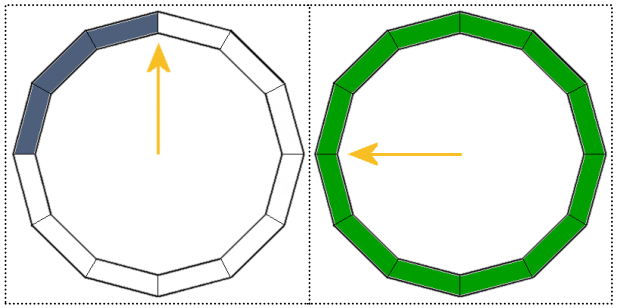
Probe Control

- The second frame is ready. We signal the *sender* to send, when possible, this frame to the console



24th step / 2nd frame

- During that time, the next frame is prepared in the other buffer. And so on…



27th step

Probe Control

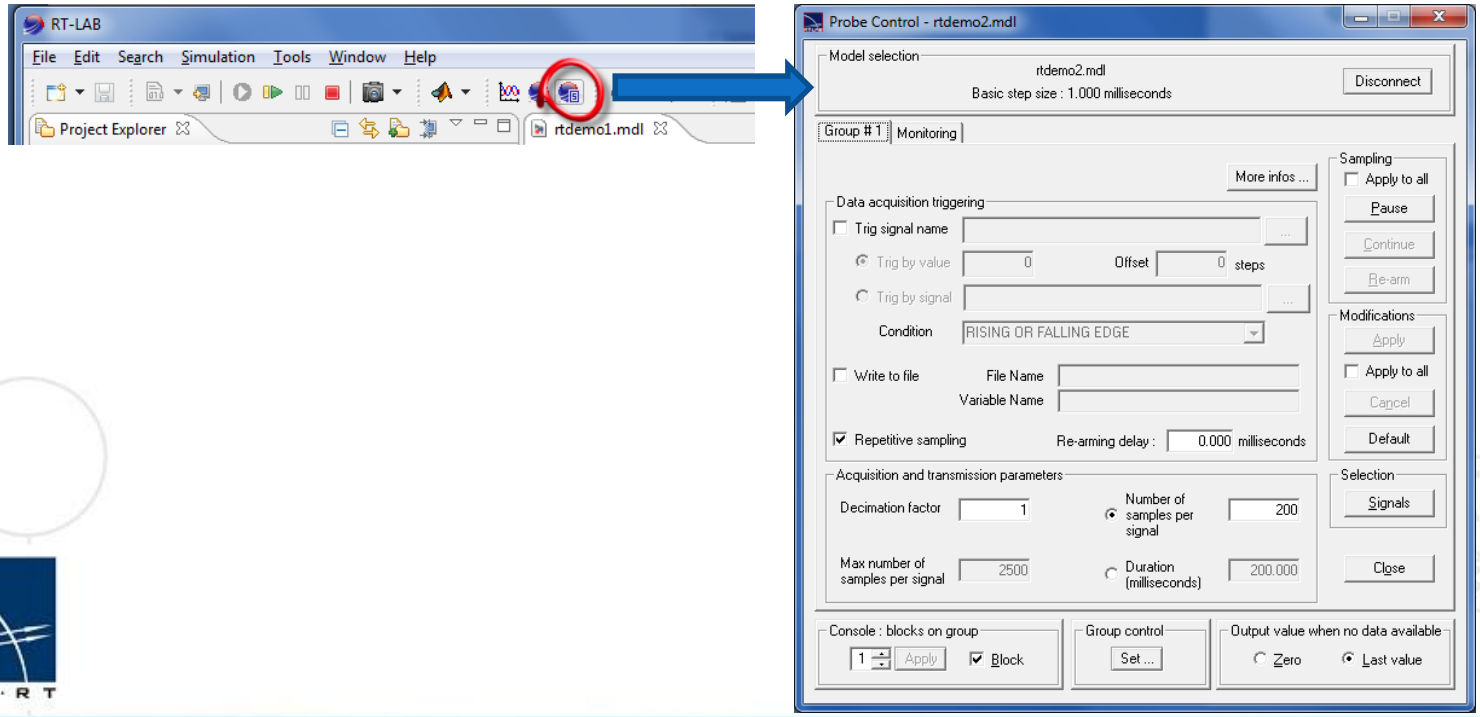
- Acquisition and transmission parameters impact only the user interface, not the model execution

- Together, the parameters determine how much data will be gathered before it's sent to the command station
  - Too much and the display will be updated in visible bursts.
  - Too little, and you risk overloading your computation node and loosing data.

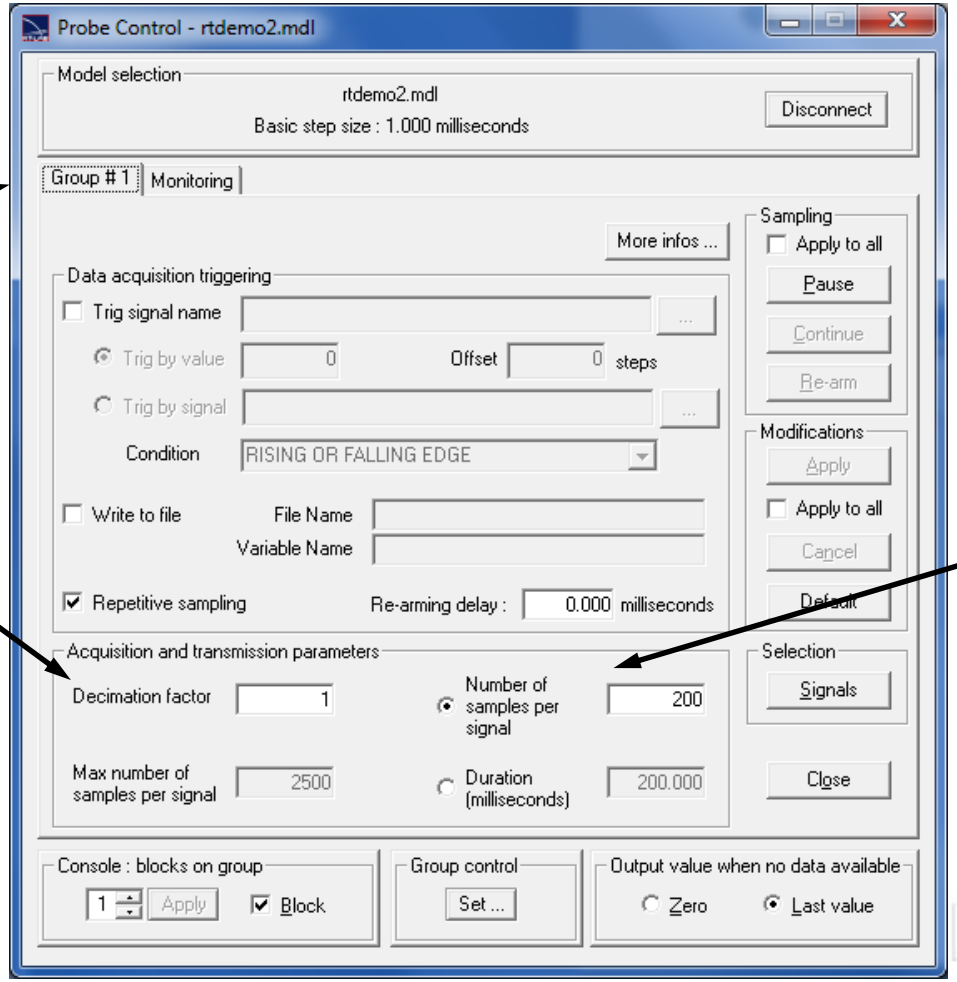
- **The solution is always a compromise.**

Probe Control

- The acquisition and transmission parameters are set per “acquisition group”
  - One OpComm block (in the console) = One Acquisition groups.
  - Decimation factor
  - Number of samples per signal
  - Duration



Probe Control



Group tabs

Decimation factor

Number of samples per signal

Probe Control

- The decimation factor determines how often on the computation node we collect a data point (for each requested signal) to put into the memory buffer

Decimation factor = 1

Decimation factor = 2

Decimation factor = 3

Probe Control

- The number of samples per signal (NS/s) represents the size of the acquisition window. It defines how many data point are gathered in the buffer before being sent to the command station

NS/s = 1

NS/s = 2

NS/s = 2, Decimation factor = 2

Probe Control

- The duration is an alternate way of choosing the number of samples to gather before sending. Instead of specifying the number of data points, the duration specifies the window size in milliseconds

- The general equation is the following:

Duration = Decimation x NS/s x Model sample time (Ts).
4ms = 2 x 2 x 1ms.

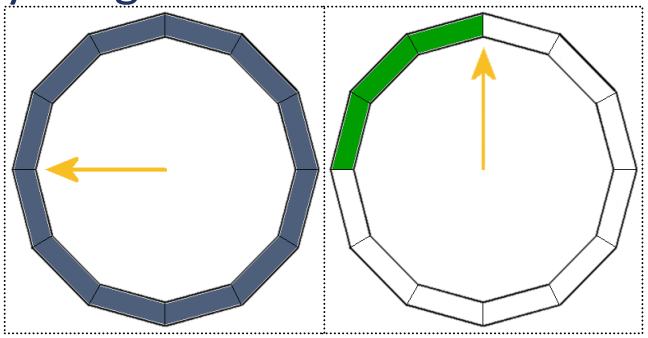
N/s = 2, Decimation factor = 2 → Duration = 4 ms

Outline

1. Model monitoring
2. Variables table
3. Probe Control
4. ***Dealing with acquisition data loss***
5. Data logging features
6. Dynamics Signal acquisition

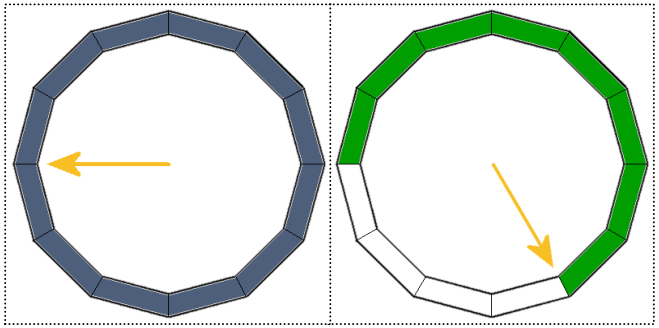
Dealing with acquisition data loss

- How data loss happens : A frame has already been prepared and is waiting to be sent. Until now, everything is normal…



15th step

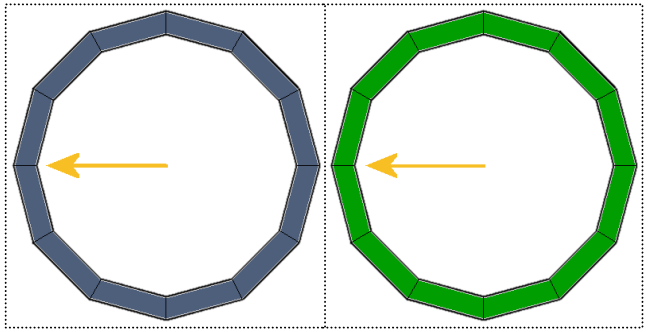
- How data loss happens :The *sender* still hasn't run the first frame is still waiting to be sent. We continue to prepare the second frame in the second buffer.



20th step

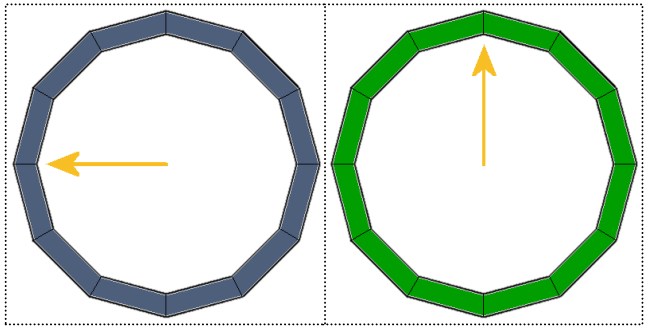
Dealing with acquisition data loss

- How data loss happens : The second frame is also ready. For some reason, the *sender* is still behind. We start to lose data.



24th step / 2nd frame

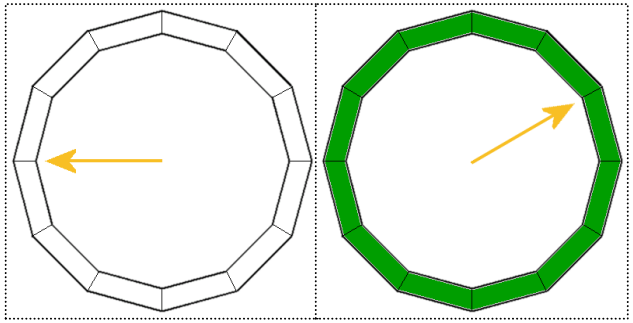
- How data loss happens : While we wait for the first frame to be sent, we overwrite the data points in the second buffer.



27th step / 3 data points lost

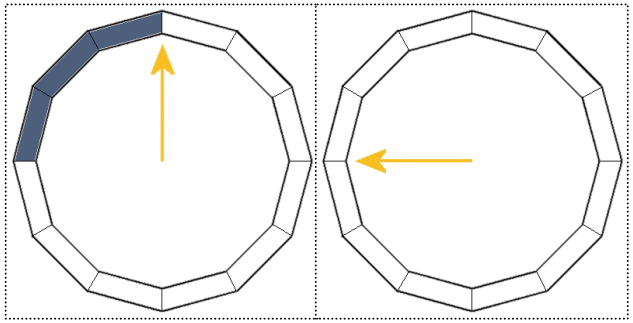
Dealing with acquisition data loss

- How data loss happens : The first frame has been sent. We resume collecting data points in the first buffer. In all, 5 data points have been lost. We still wait for the second frame to be sent.



24th step / 2nd frame

- How data loss happens : The second frame has also been sent. We are back on track…



27th step

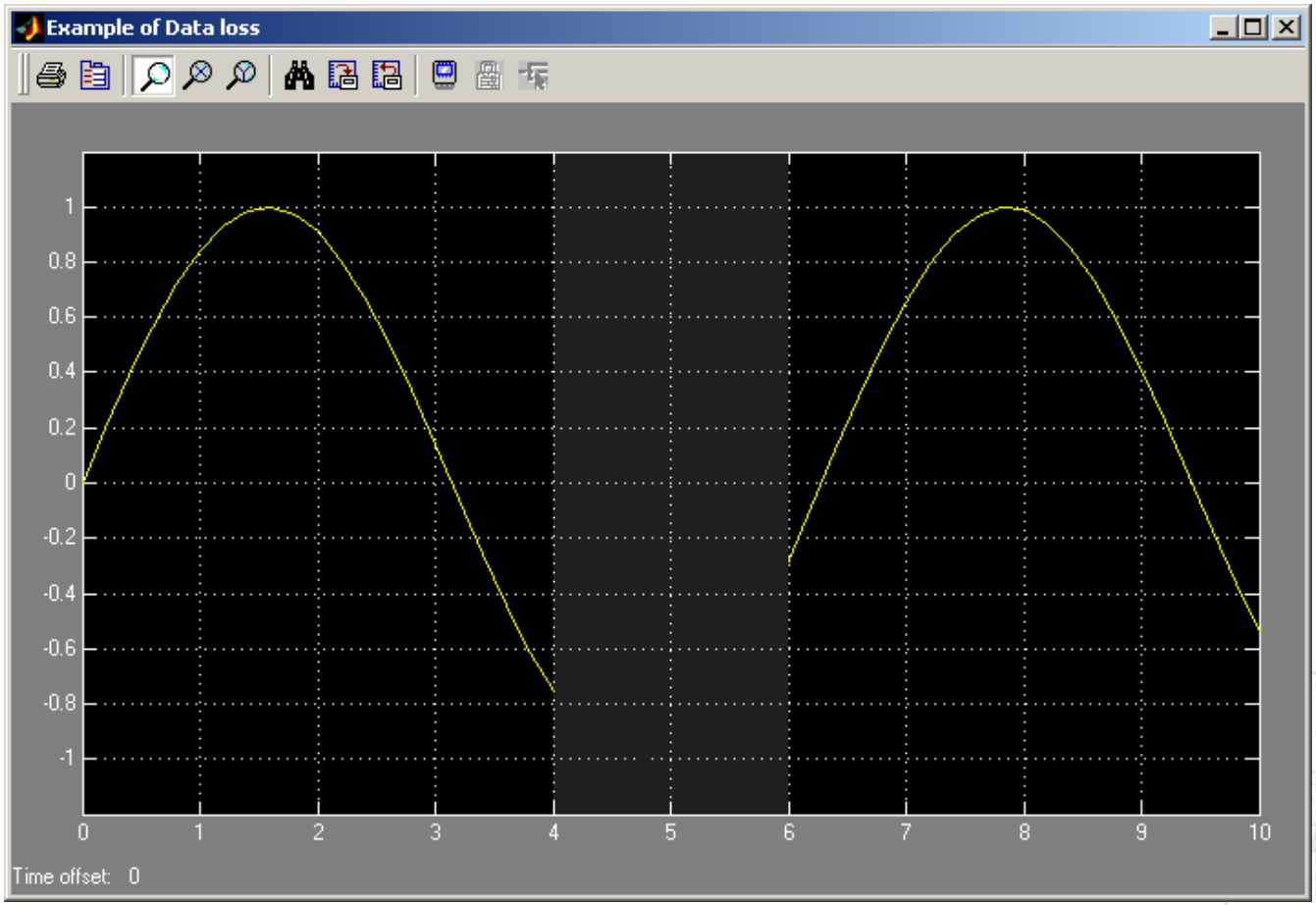
Dealing with acquisition data loss

RT-LAB will always transfer acquisition frame in totality. Data loss will be seen as jumps between received frames. To minimize data loss, depending on the cause, you can:

- The target CPU is busy
  - Increase the step size of the model
  - increase the processing power of the system
  - Use a larger frame size
- The network bandwidth is exceeded
  - Use a larger decimation factor or frame size.
  - A faster network (if possible) is always best
  - See also the OpWriteFile option discussed later.
- The console CPU is busy
  - Use a larger frame size
  - Faster processor
  - Faster GUI

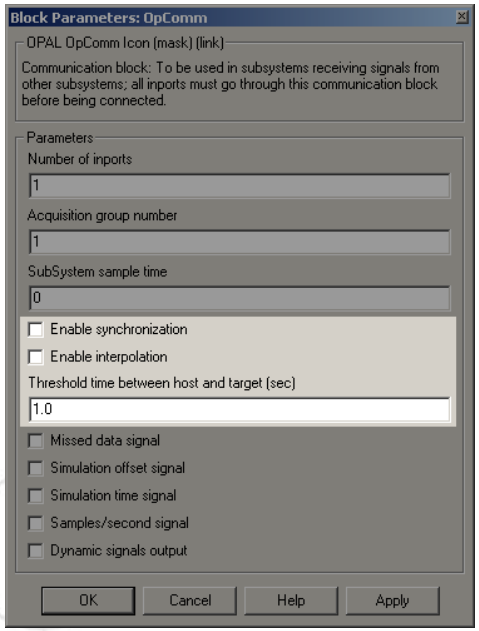
Dealing with acquisition data loss

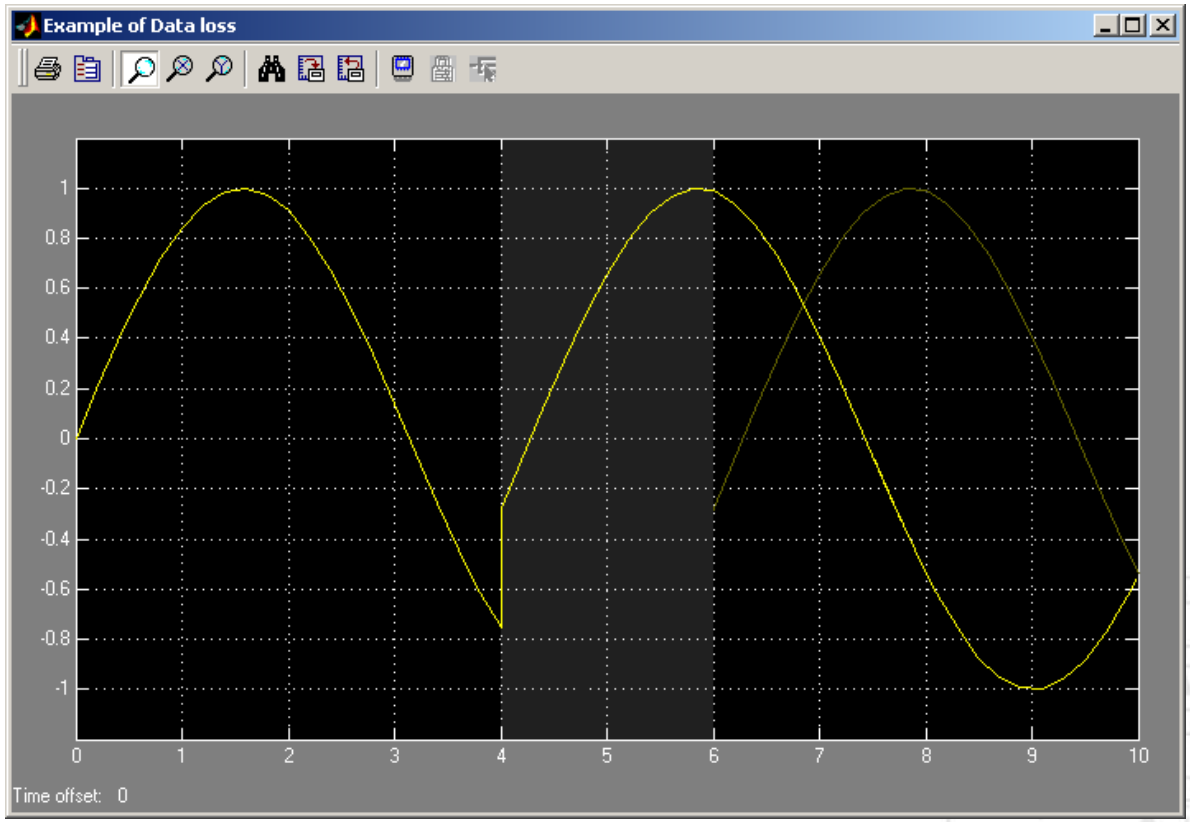
RT-LAB has many ways to deal with lost data. Let's look at an example where 2 seconds worth of data would be lost (regardless of the cause)



Dealing with acquisition data loss

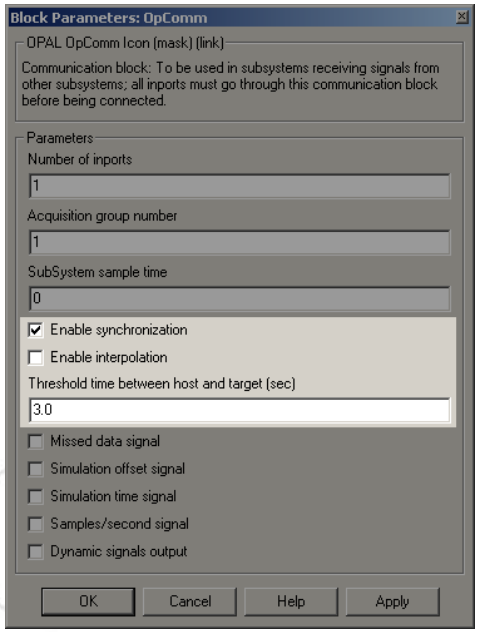
Without any compensation, the signal trace would look like this because Simulink uses received data to set it's time, the Simulink clock would be two seconds behind the model (real-time) clock

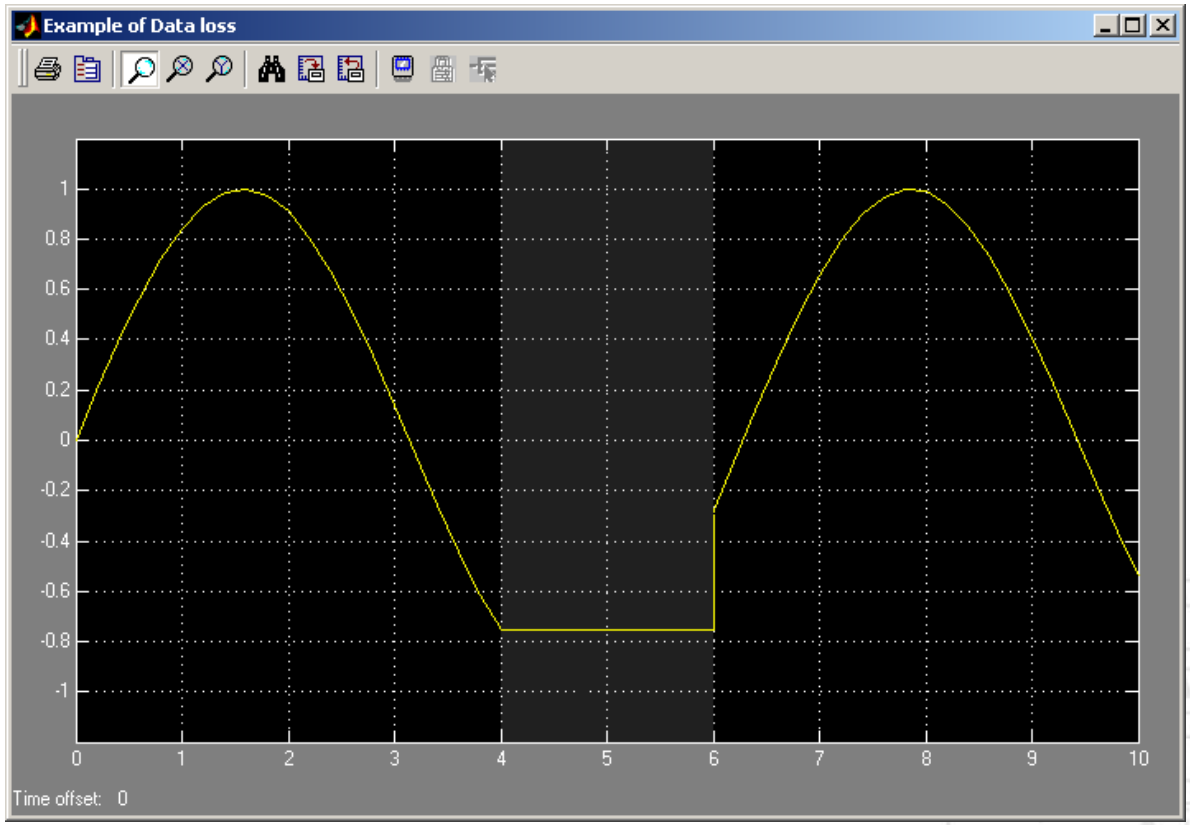




Dealing with acquisition data loss

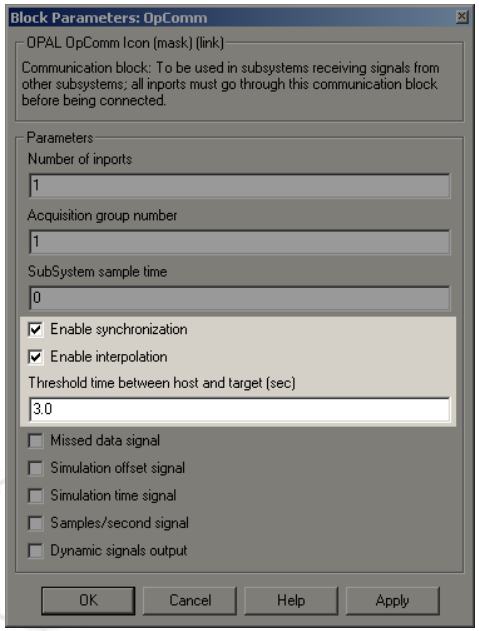
In order to keep the Simulink clock in sync as much as possible, RT-LAB has a synchronization algorithm which will insert dummy data points when data is lost and the loss in less than the threshold time

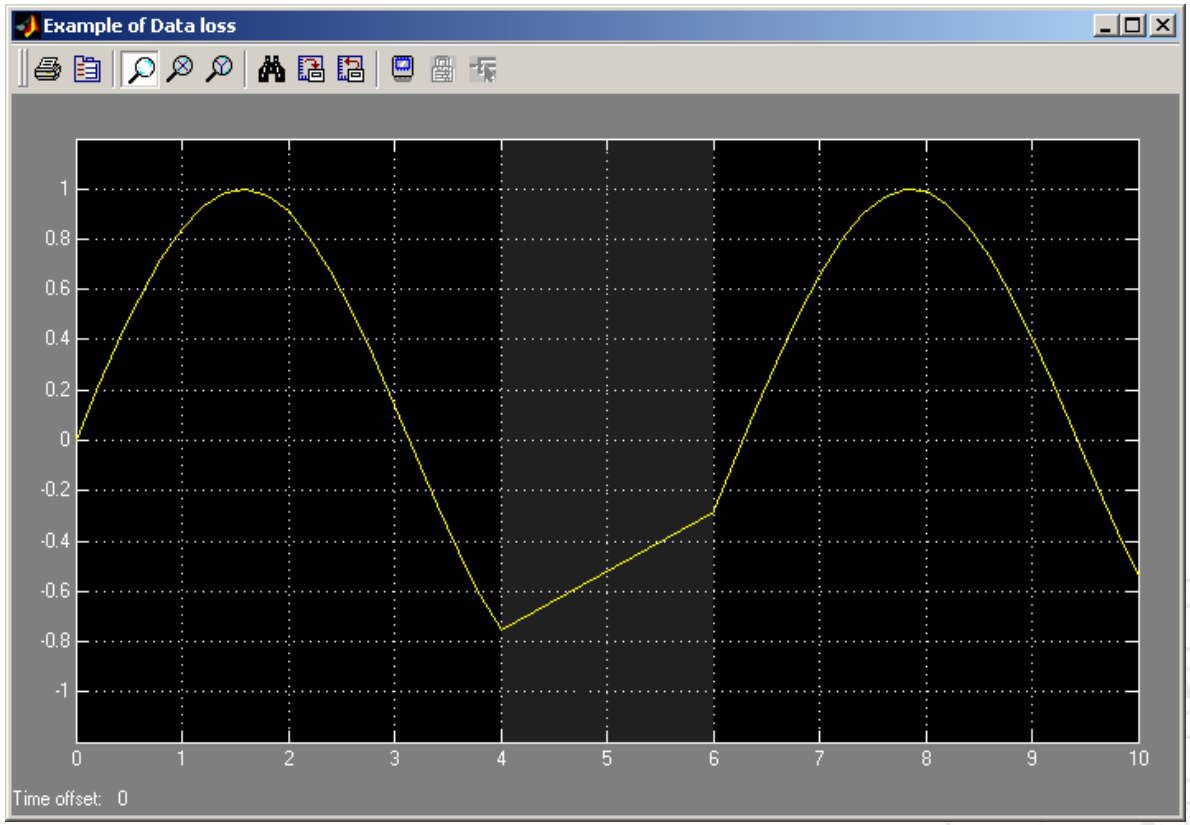




Dealing with acquisition data loss

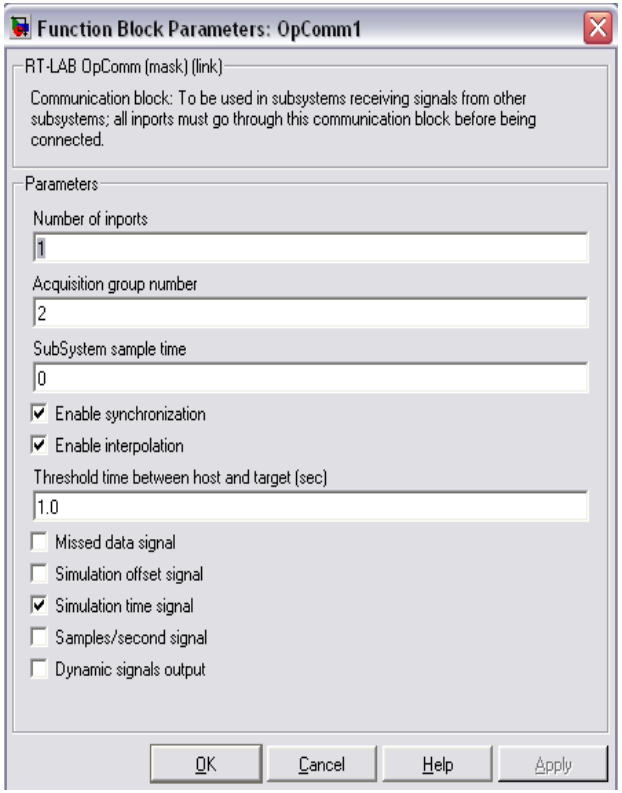
Optionally, RT-LAB can interpolate between the last and new data points to smooth out the curve (not very useful here).



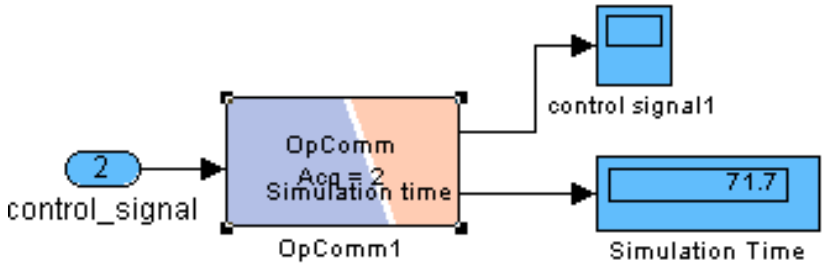


Dealing with acquisition data loss

Regardless of synchronization or interpolation, you can always get the exact time of the simulation from the real-time model by checking the "*Simulation Time Signal*" in the console OpComm block



This will add an output with the time value to the OpComm.

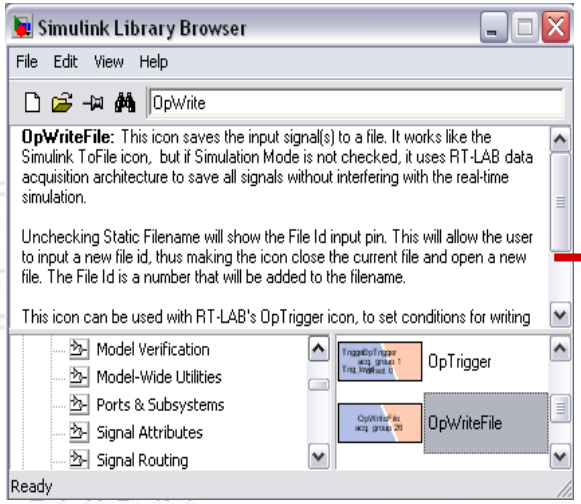


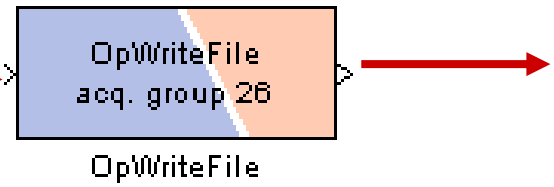
Outline

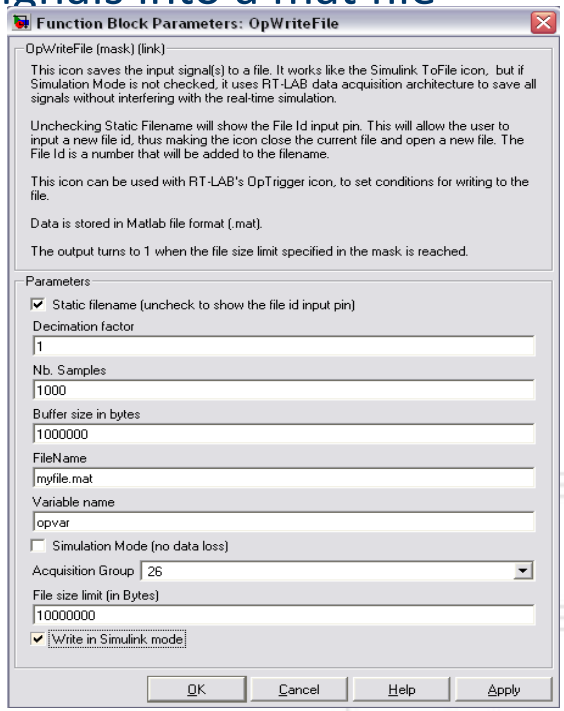
1. Model monitoring
2. Variables table
3. Probe Control
4. Dealing with acquisition data loss
5. ***Data logging features***
6. Dynamics Signal acquisition

Data logging features

- RT-LAB allows acquisition to be done directly to a file instead of going through the network to the console.
- Define dedicated acquisition groups (26-30) that only write to file :
  The OpWriteFile block, from the RT-LAB block library in Simulink allows you to wire signals to be recorded to file. It contains the same acquisition parameters discussed earlier. You can record multiple signals into a mat file using a “Mux” block to create a vector.







Data logging features

- You can have up to five OpWriteFile blocks in a model. Each block has a unique filename that is entirely user-defined

- Data structure is :
  - first array for time
  - Other arrays for signals, same order as signals enter the block

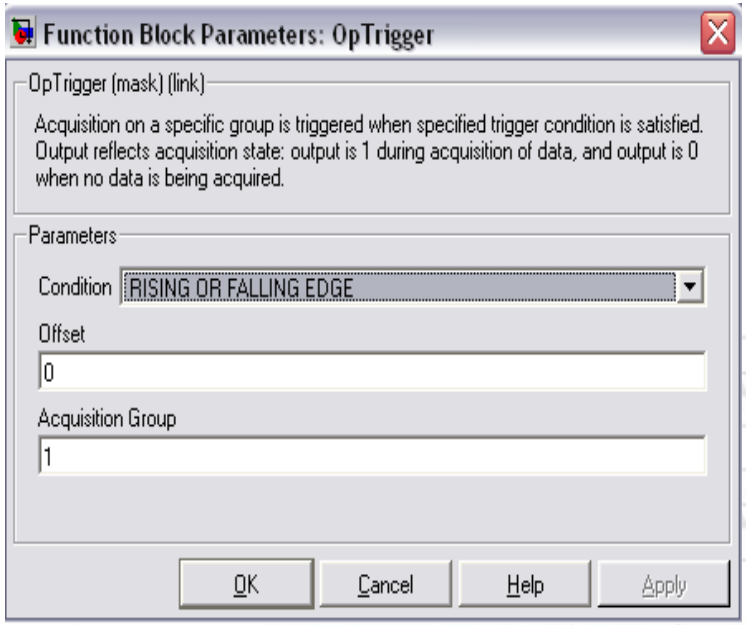
- This block is also able to record data offline by checking the option “Write to Simulink mode”

Data logging features

Triggering acquisition groups

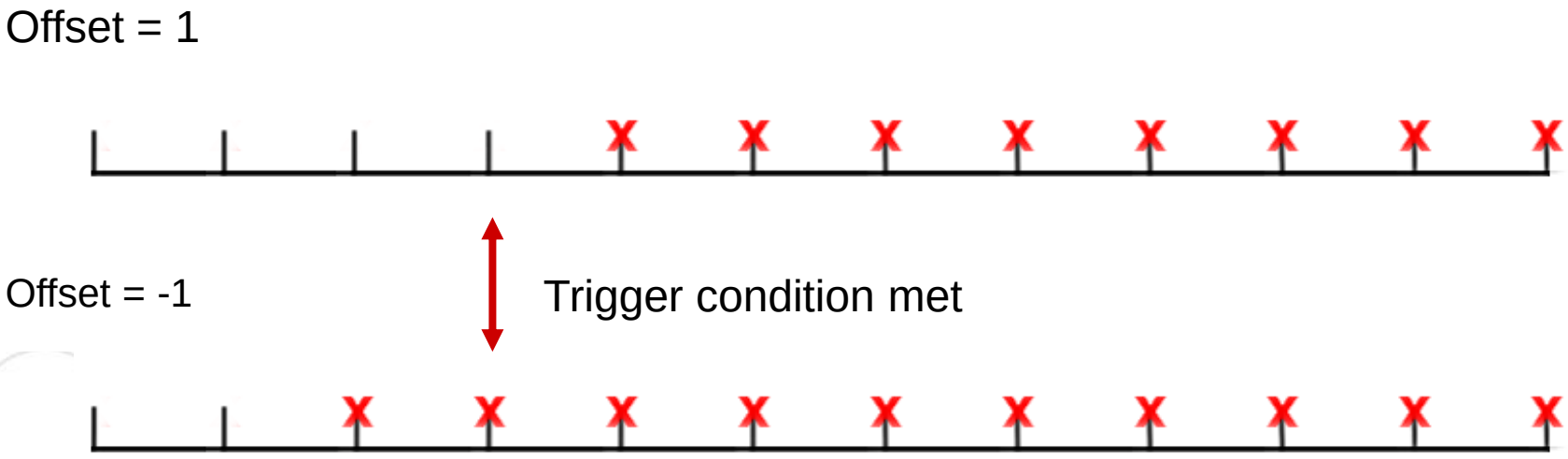
- RT-LAB's default behaviour is to continuously acquire signals connected to an acquisition group. The OpTrigger block allows the acquisition to be conditional.

- When triggered, an acquisition group will acquire only one frame and then waits for the next trigger



Data logging features

- OpTrigger block can trigger any acquisition, Opcomm from sc_ or OpWriteFile blocks
- Using the Offset parameter, it is possible to specify how many steps BEFORE the trigger the frame will start.



Outline

1. Model monitoring
2. Variables table
3. Probe Control
4. Dealing with acquisition data loss
5. Data logging features
6. ***Dynamics Signal acquisition***

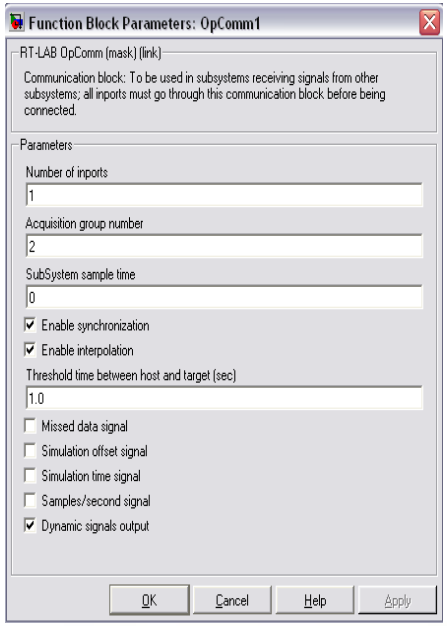
Dynamics Signal Acquisition

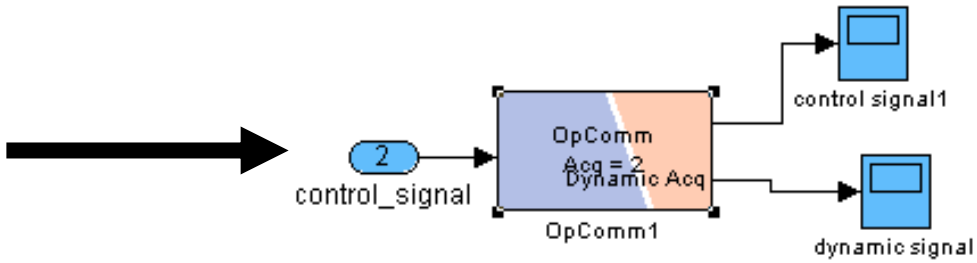
- Until now, all the signals we have managed to visualize in the console have been explicitly wired from the computation subsystems (SM or SS) to the console (SC subsystem).
- RT-LAB also allows the selection of signal for visualization to be done dynamically, at run time.
- This is called dynamic signal selection.

Dynamics Signal Acquisition

- In order to receive dynamic signals and be able to display them in the console, we must enable their reception in one or more acquisition group

- In the OpComm block of the acquisition group of your choice, simply check the “Dynamic signals output”. This will add an extra output to the block. You can connect any Simulink block to process the received signals

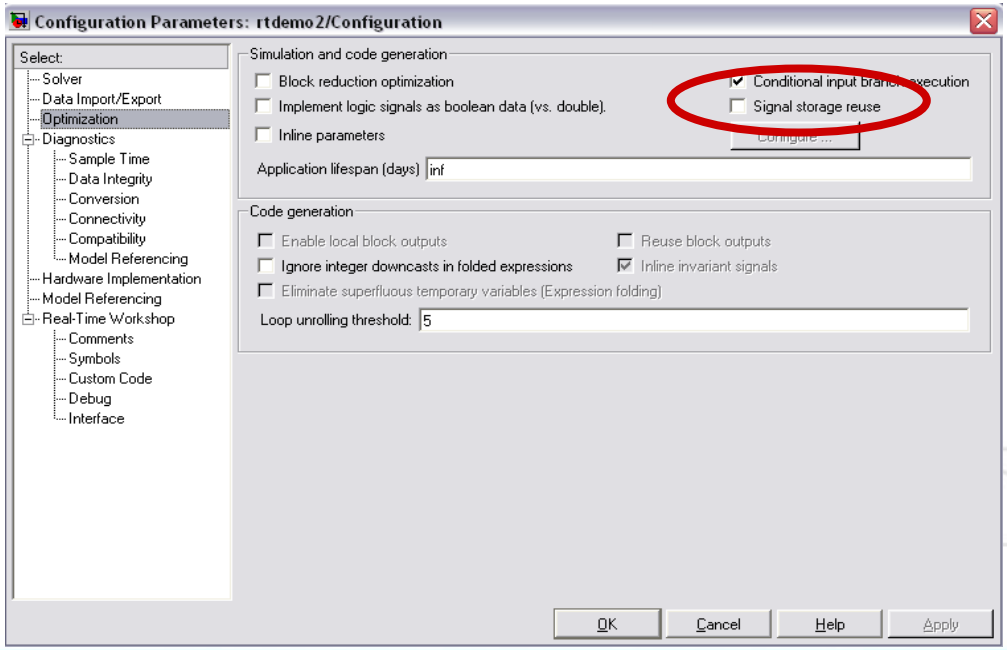




Dynamics Signal Acquisition

- By default, Simulink optimizes the code it generates with RTW. This optimization can lead to some blocks' outputs not being available for selection.

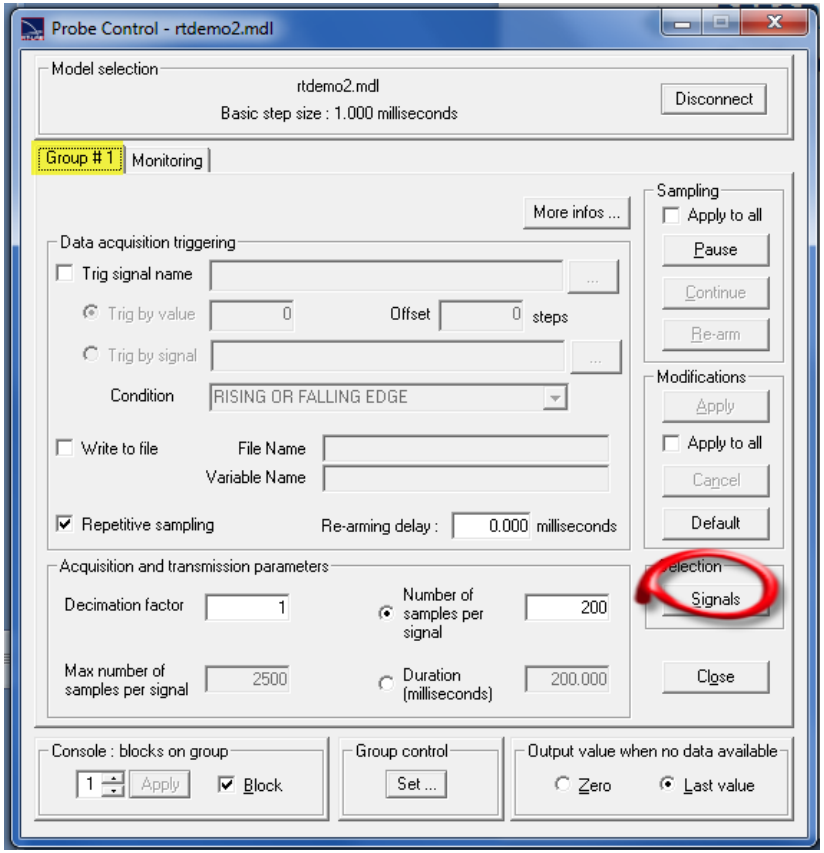
- In order to bypass this, we need to tell Simulink to not optimize away the outputs in the signal structure. This is done by setting "Signal Storage Reuse" to Off in the advanced Simulink settings.



Dynamics Signal Acquisition

- BEFORE LOADING :

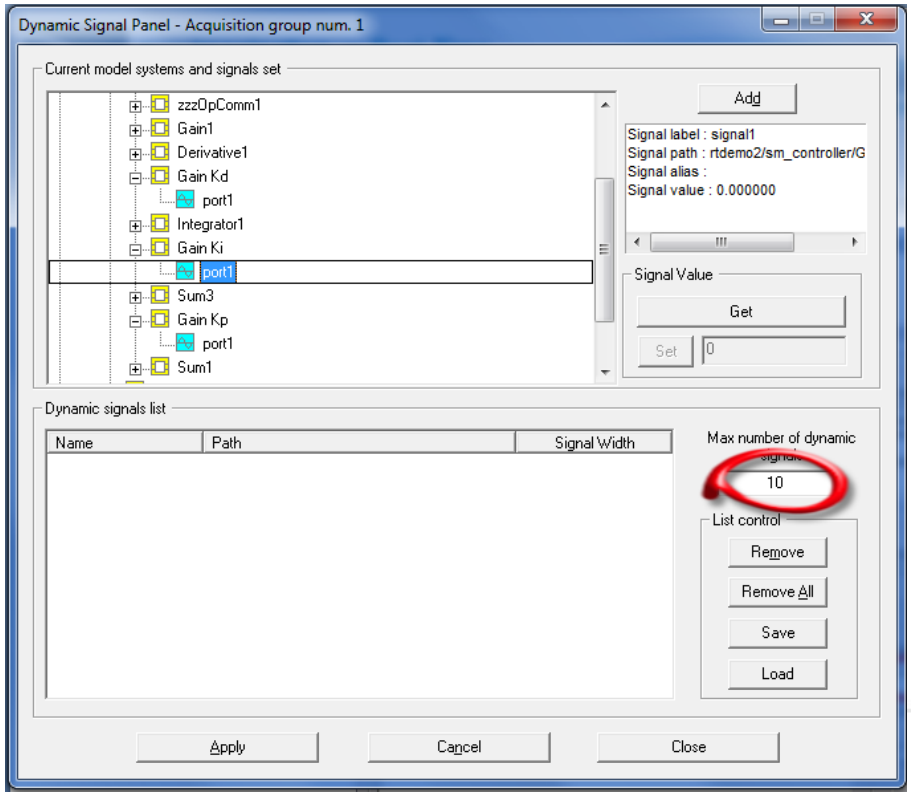
Bring up the dynamic signals dialog from the *Probe Control* panel.



Dynamics Signal Acquisition

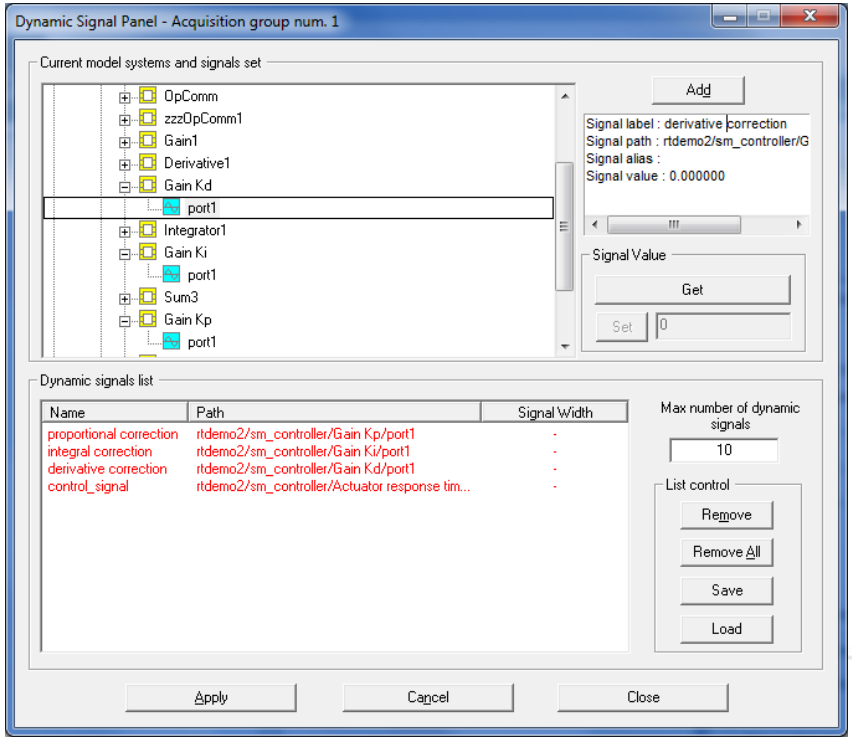
- BEFORE LOADING :

  You need to set the maximum number of dynamic signals you want to be able to manipulate. This is necessary before loading for the memory allocation of the acquisition buffer



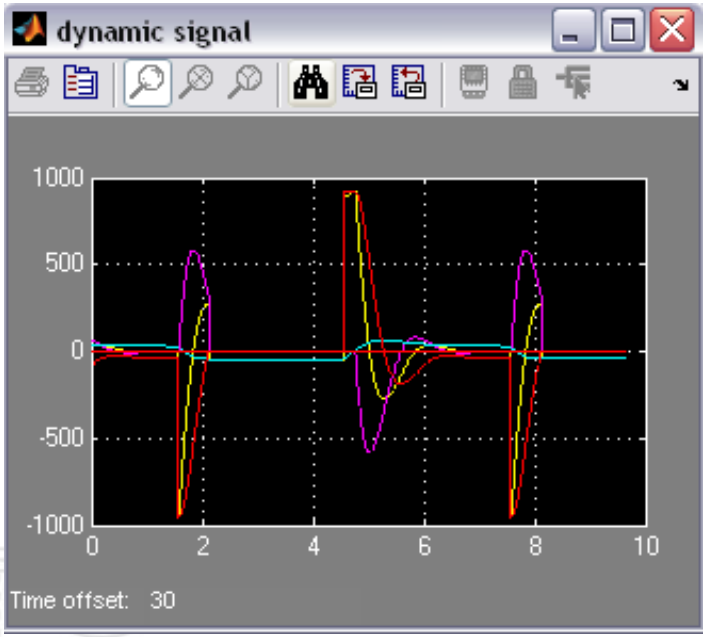
Dynamics Signal Acquisition

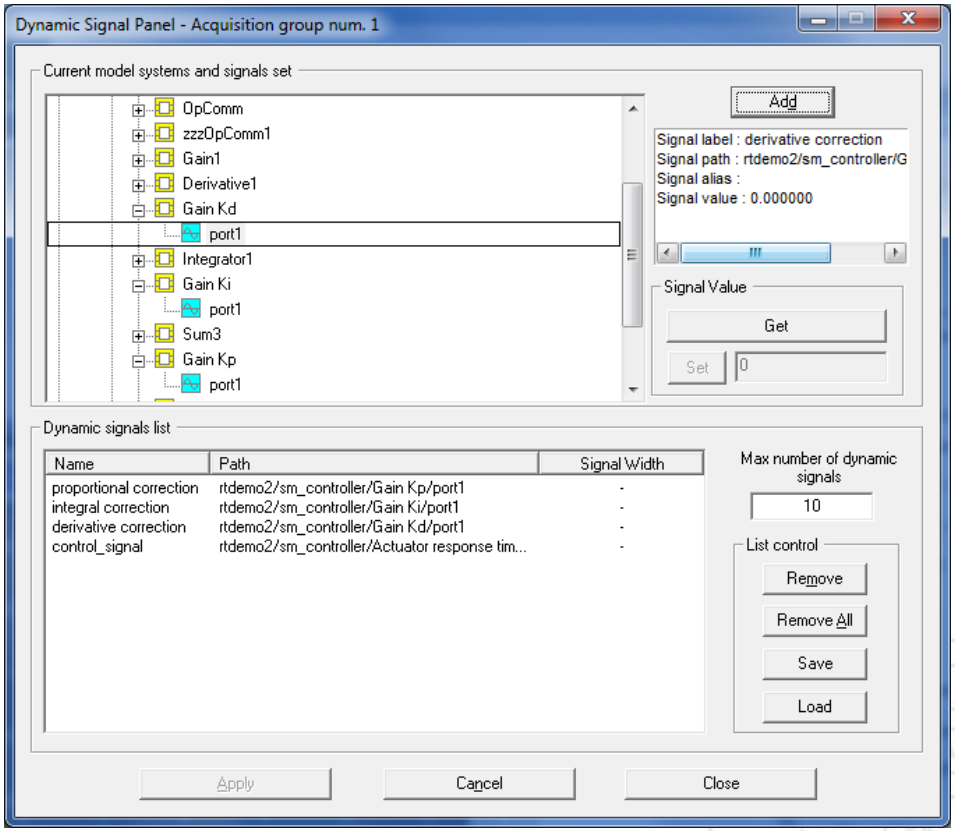
- RUN-TIME :
  - Select the signal you want to visualize from the tree list. The names that are listed are the signal names in the original Simulink model
  - Be sure to use explicit names to make it easier here



Dynamics Signal Acquisition

- Press “Apply” and you can immediately see the selected signals in the Simulink console





Questions ?

- Model monitoring
  - Running information about real time simulation
- Variables table
  - See value in the model
  - Change value on the fly
- Probe Control
  - Control the data acquisition of the console panel
- Dealing with acquisition data loss
  - Understanding the loss of data between the model and the console
- Data logging features
  - Write data in a .mat file
- Dynamics Signal acquisition
  - Probe your model without change