Content

• Introduction
• Clocks
• Sample and Hold
• Sub- and Super-sampling
• Discretizing Continuous Blocks
• Modelica_DeviceDrivers library
• Conclusions
Introduction

• Synchronous language elements of Modelica 3.3 are “low level”:

  // speed sensor
  vd = sample(v, Clock(0.01));

  // P controller for speed
  u = K*(vref-vd);

  // force actuator
  f = hold(u);

• Modelica_Synchronous library developed to access language elements in a convenient way graphically:

Blocks to generate clock signals

Blocks operating on clock signals (e.g. sub-sampling a clock signal)

Blocks operating on clocked signals of type Real (e.g. sub-sampling a Real signal, PI block, FIR filter)

Blocks operating on clocked signals of type Integer

Blocks operating on clocked signals of type Boolean
Clocks

New base data type: **Clock**
Variables associated to a clock have only a value at the clock tick.

![Diagram of clock values](image)

- **c(t): Clock**
- **r(t): Variable associated to c**

Similar to Real, Integer, Boolean, introduced input/output **Clock connectors**:

```modelica
connector ClockInput = input Clock;
connector ClockOutput = output Clock;
```

Blocks that generate clock signals

**Generates a periodic clock with a Real period**

```modelica
parameter Modelica.SIunits.Time period;
ClockOutput y;
equation
  y = Clock(period);
```

**Generates a periodic clock as an integer multiple of a resolution (defined by an enumeration).**

Code for 20 ms period:

```modelica
y = superSample(Clock(20), 1000);
```

Clock with period 20 s
super-sample clock with 1000 period = 20 / 1000 = 20 ms

**Generates an event clock**: The clock ticks whenever the continuous-time Boolean input changes from false to true.

```modelica
y = Clock(u);
```
Sample and Hold

Discrete-time PI controller

Holds a clocked signal and generates a continuous-time signal. Before the first clock tick, the continuous-time output \( y \) is set to parameter \( y_{\text{start}} \): \( y = \text{hold}(u) \);

\[
y = \text{sample}(u, \text{clock});
\]

Samples a continuous-time signal and generates a clocked signal.

Purely algebraic block from Modelica.Blocks.Math

\[
y = \text{sample}(u);
\]

Sub- and Super-Sampling

Defines that the output signal is an integer factor faster as the input signal, using a “hold” semantics for the signal. By default, this factor is inferred. It can also be defined explicitly.

\[
y = \text{superSample}(u);
\]
Defines that the output signal is an integer factor slower as the input signal, picking every n-th value of the input.

\[ y = \text{subSample}(u, \text{factor}); \]

Several other blocks to change the clock of a signal, such as...
Discretizing Continuous Blocks

A clocked partition can consist of differential equations, provided an integrator is associated to the corresponding clock (the differential equations are solved at one clock tick with this integrator).

Especially, inverse, continuous-time models can be discretized (this is not possible with Modelica 3.2):

Especially inverse plant model (input $T_c$ becomes output)
output $c$ becomes input)
Modelica_DeviceDrivers library

- New, free library by DLR that interfaces hardware drivers.
- Cross platform (Windows and Linux)
- Realtime synchronization, UDP, joystick, keyboard, etc.
- Basic functionality provided with Modelica functions.
- Convenience blocks based either on Modelica_Synchronous library or on Modelica_3.1 when-clauses.
- Generic packaging system, e.g. to pack 8 Booleans on 1 Integer.
- Currently supported for UDP, shared memory, and (prototypical) CAN bus.

Conclusions (1)

Modelica_Synchronous library

- Main purpose:
  Encapsulating the Modelica 3.3 synchronous language elements with an easy to use graphical user interface (the code of most blocks is very simple!!)
  Makes definition of sampled-data systems much simpler and safer.
- Should work with every Modelica tool that supports Modelica 3.3.
- Shall be included in the Modelica Standard Library after an evaluation period.
- Available for MA members on internal svn server.
  Released version is stored publicly on MA web.
Conclusions (2)

Modelica_DeviceDrivers library

- Main purpose:
  Access device drivers on Windows and Linux PCs from a Modelica block.
- Should work with every Modelica tool that supports
  external Modelica functions (with C-code included in include annotation)
  synchronous elements of Modelica 3.3 (for Modelica 3.3 convenience blocks).
- Shall be included in the Modelica Standard Library after an evaluation period.
- Available for MA members on internal svn server.
  Released version is stored publicly on MA web.