

Stateflow 7.1

Agenda

- Complex support
- Multi-output graphical functions
- Absolute time temporal logic
- Temporal count operator
- Bidirectional traceability between model and generated code

Complex Data Support in Stateflow

Challenge

- Signal processing designers typically work with complex data.

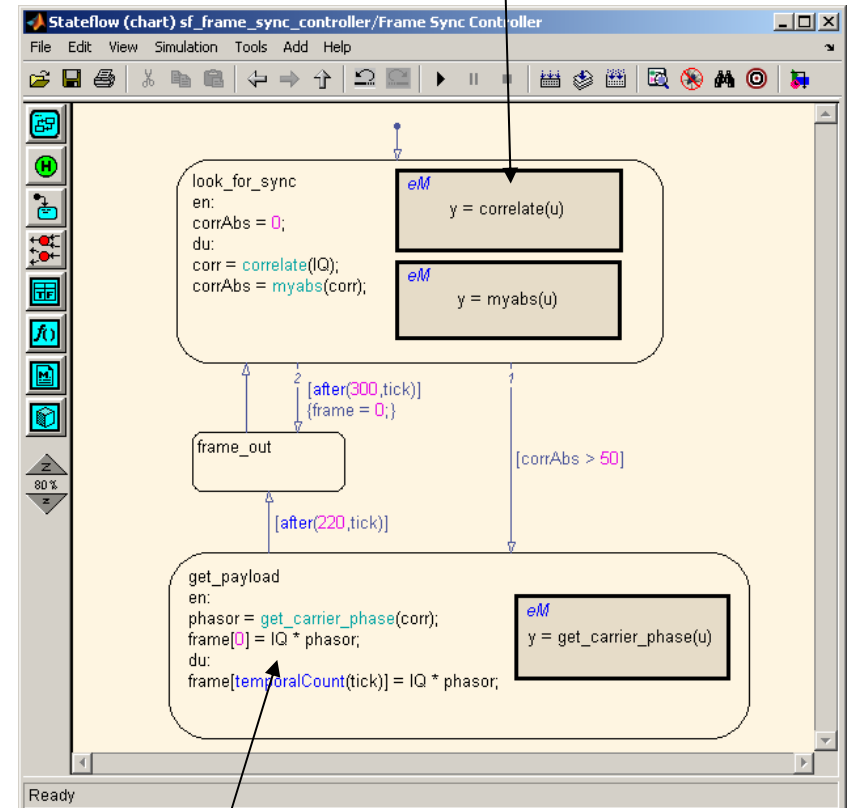
Solution

- Stateflow now supports complex data for inputs, outputs, locals, parameters, and functions (graphical, Embedded MATLAB, and truth table).
- Stateflow also supports a minimal set of operations of complex variables directly on the states and transitions:
 - + , - , * , == , != , =
 - Accessing real and imaginary parts of complex variable
 - Constructing a complex variable from real and imaginary parts

Benefit

- Use Embedded MATLAB functions in Stateflow to design signal processing applications that require mode switching and complex data support

Complex support in Embedded MATLAB functions



Basic arithmetic support for complex data

Multi-Output Functions

Challenge

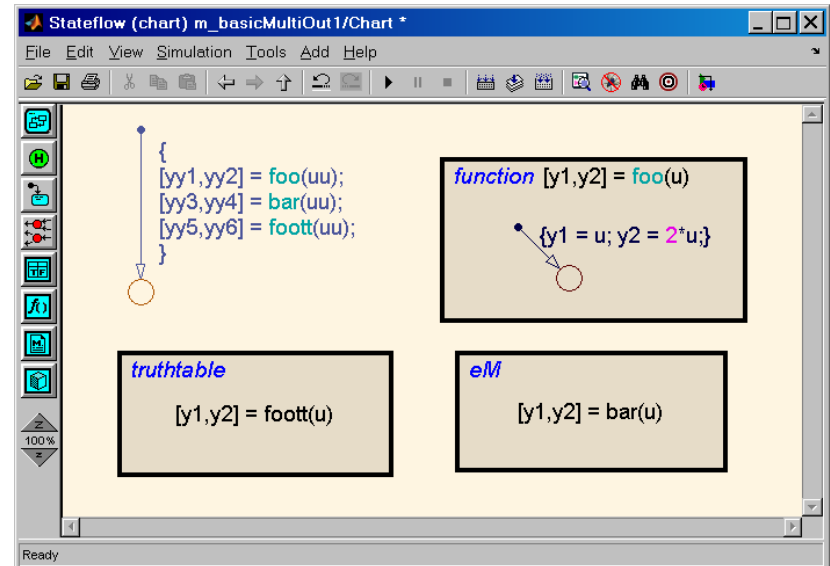
- Users who create functions in Stateflow need to define multiple output variables for their functions.

Solution

- Stateflow now supports functions (graphical, Embedded MATLAB, and truth table) that can have multiple output data. The calling syntax is identical to the syntax used in MATLAB:
 - `[x,y] = my_sort(u)`

Benefit

- No need to define multiple functions when only multiple outputs are required
- Increased efficiency of the generated code



Absolute Time Temporal Logic

Challenge

- Many applications, especially in automotive vehicle electronics, require specifying timing logic (timeouts, alarms, and so forth) using absolute time units.

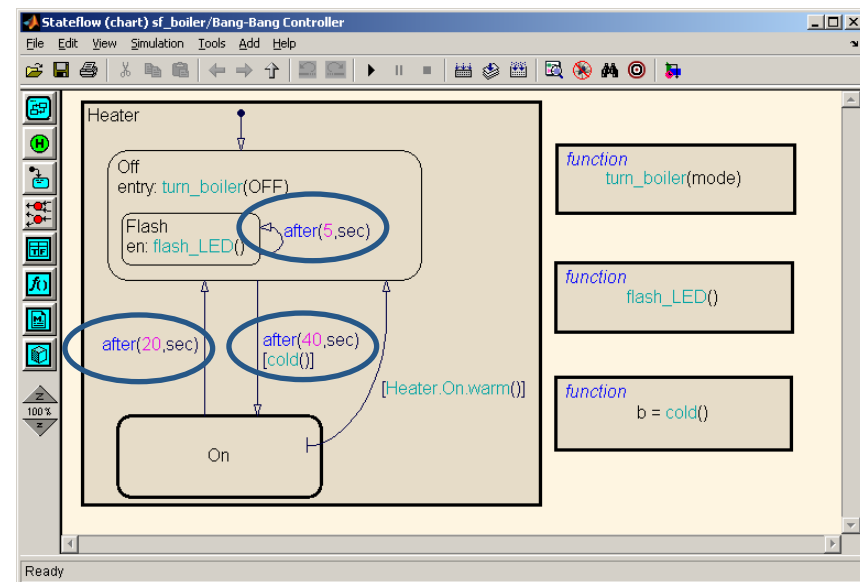
Solution

- New keyword “sec” for temporal operators provides support for “absolute time temporal logic constructs” in transition conditions. For example:

- `after(3.5,sec)`
- `before(x,sec)`
- `at(K,sec)`

Benefit

- No overhead in setting up input events from Simulink
- Makes design easier to read



`>>sf_boiler`

Temporal Count Operator

Challenge

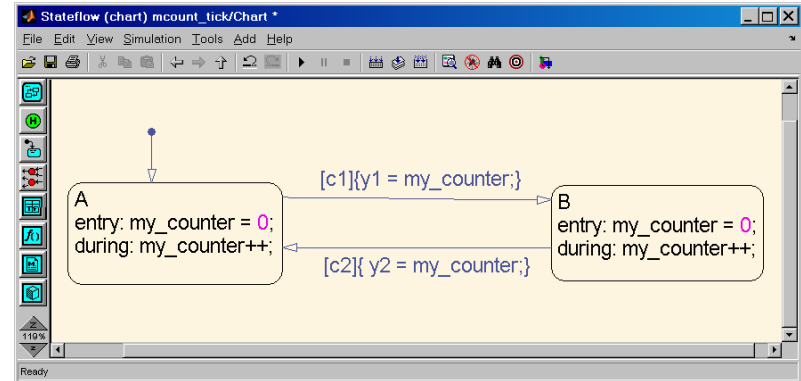
- Implementing counters in states (e.g., count how many times a chart wakes up while in a state) can be difficult to set up and requires overhead.

Solution

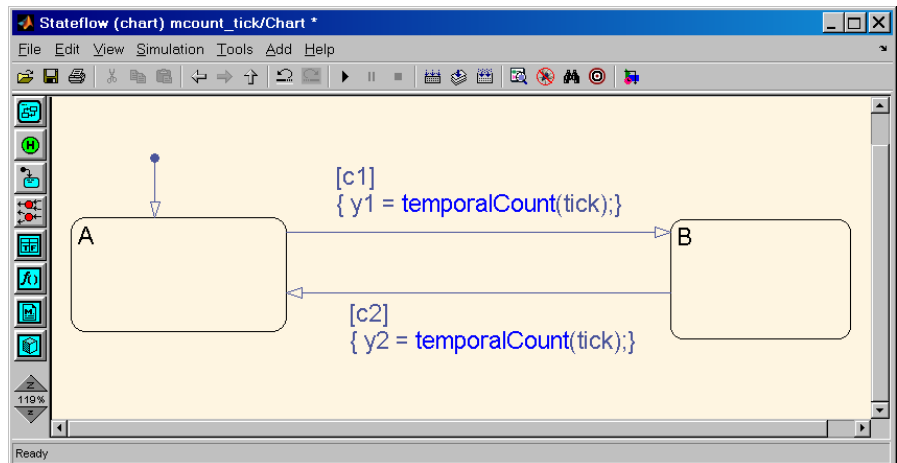
- New `temporalcount()` operator implicitly maintains state counters:
 - `X = temporalcount(tick)` tells you how many times the chart woke up since the current state was entered.

Benefit

- Less overhead in setting up and debugging state counters
- Makes design easier to read



Upon exiting A and B, `y1` and `y2` get the number of “ticks” for which the states A and B were active. Explicit definition of counters was required before R2008a.



In R2008a, there is no need to define and maintain explicit counters.

`>>sf_frame_sync_controller`

Stateflow and Embedded MATLAB Code Traceability (with Real-Time Workshop Embedded Coder)

Challenge

- It was difficult to trace the specific code that gets generated from each part of a Stateflow chart and vice versa.

Solution

- Bidirectional code traceability is now supported (support for Simulink in R2007b).

Benefit

- Know exactly what code is generated for each Stateflow object and vice versa
- Navigate easily from chart to code and back

