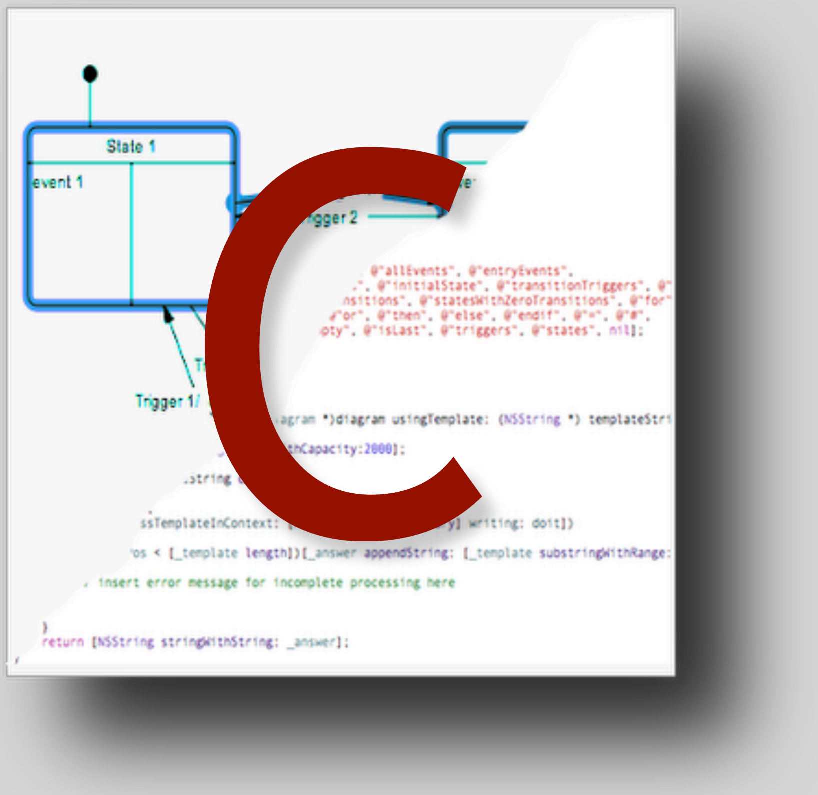
A Template for C-Code Generation using Switch Statements



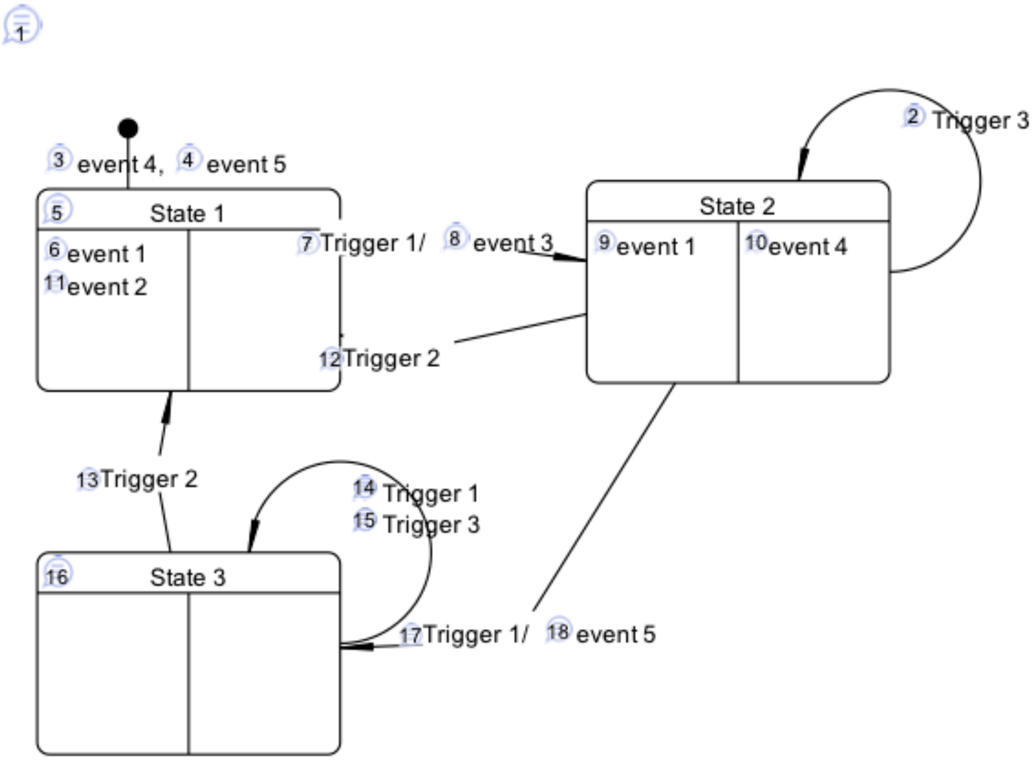
Code Templates for the State Diagram Editor

License

This product, a Template for the State Diagram Editor is copyrighted shareware, not public-domain software. You may use the unregistered version with certain restrictions for personal, non-commercial purposes.
Upon request, you will receive an unlimited license for a period of thirty days. To use the software after this period for commercial purposes, you have to register it. The commercial license is bound to one specific machine,but may be transferred at no cost to another one.
Stefan Sachs does not warrant that State Diagram Editor will meet your specific needs or requirements,, nor that the operation of State Diagram Editor will in all instances be uninterrupted or error-free. Because software is inherently complex and may not be completely free of errors, you must verify all results before relying on them.
Stefan Sachs shall not be liable for any direct, indirect, consequential, special, incidental, or other damages, including but not limited to damages for lost profits, interruption of business, lost or corrupted data or programs, system crashes, or diversion of system or other resources, arising out of or relating to this agreement or the performance, quality, results, use of, or inability to use the State Diagram Editor or any support, advice, products, or services which are or should have been provided to you, even if Stefan Sachs has been advised of the possibility of such damages.
State Diagram Editor may contain technical protections that restrict use to certain equipment, to a single personal computer or workstation, or to a certain time period, and may provide other restrictions, all of which shall be consistent with the License. Stefan Sachs shall not be liable for any damages incurred as a result of such technical protections or the failure of State Diagram Editor to operate outside the scope of the license.
Stefan Sachs shall not be liable for any claim asserted against you by any other party, including but not limited to claims for infringement of copyright or other intellectual property rights.
Some jurisdictions do not permit the exclusion or limitation of implied warranties or consequential damages, so some of the above exclusions or limitations may not apply to you.
Stefan Sachs may terminate the License with or without cause at any time immediately and without notice if you materially breach any term of this Agreement or infringe any of Stefan Sachs’s rights in State Diagram Editor.
Stefan Sachs may terminate the License upon ten days written notice if Stefan Sachs believes that your use of State Diagram Editor may infringe any third party rights. If such termination occurs within 24 months of your first installation of State Diagram Editor, Stefan Sachs will provide you a modified, non-infringing version of State Diagram Editor and reinstate the License.
Immediately upon termination of the License, you must stop using State Diagram Editor and destroy all copies of State Diagram Editor within your possession, custody, or control.
You may not assign, sublicense, or otherwise transfer the License or any other rights arising under this Agreement. Any purported transfer in violation of this paragraph shall be void

A Template for C-Code generation using Switch Statements

The template C-switch.codetemplate is used to generate linear C-code using for each trigger a distinct function. Each of those functions consists of a switch statement with cases for each state, in which the trigger either causes at least one event to be trigggered or a transition to another state (or both).
There is no special treatment for zero transitions.
For diagrams triggering a lot of events this representation supports easy debugging, it may be advisable, to have the original diagram as reference during debugging since the code does not reflect the states table as clearly as the table based C-code.



Comments:
1 Standard example diagram to demonstrate code generation
2 Trigger 3 results in a zero transition in state 2
3 event 4 triggered when initializing the state machine
4 event 5 triggered when initializing the state machine
5 This is the initial state
6 event 1 triggered when entering state 1
7 Trigger 1 results in the transition from state 1 to state 2
8 event 3 is triggered, when the transition from state 1 to state 2 takes place
9 event 1 triggered when entering state 2
10 event 4 triggered when leaving state 2
11 event 2 triggered when entering state 1
12 Trigger 2 results in the transition from state 2 to state 1
13 Trigger 2 results in the transition from state 3 to state 1
14 Trigger 1 results in a zero transition in state 3
15 Trigger 3 results in a zero transition in state 3
16 This is state 3
17 Trigger 1 results in the transition from state 2 to state 3
18 event 5 is triggered, when the transition from state 2 to state 3 takes place

The example diagram

C-Header generated from the Example Diagram:

```
// Code generated by StateDiagramEditor (c) Copyright 2010 Stefan Sachs www.ssachs.de
// Full version, temporarily licensed to Daisy Duck
// commercial use of this generated code is illegal
// (c) Copyright for template 2009  Stefan Sachs
// generated from diagram demo at 2010-10-19 21:18:05 +0200
// Standard example diagram to demonstrate code generation
#ifndef _demo_H
#define _demo_H

void demo_init(void);

void  Trigger1(void);
void  Trigger2(void);
void  Trigger3(void);

typedef enum {
       demo_State1,  // This is the initial state
       demo_State2,
       demo_State3  // This is state 3
       } demo_state_id;
#endif
```

C-Code generated from the Example Diagram:

```
// Code generated by StateDiagramEditor (c) Copyright 2010 Stefan Sachs www.ssachs.de
// Full version, temporarily licensed to Daisy Duck
// commercial use of this generated code is illegal
// C Code using switch statements
// (c) Copyright for template 2009  Stefan Sachs
// generated from diagram demo at 2010-10-19 21:18:05 +0200
// Standard example diagram to demonstrate code generation
#include <demo.h>

static demo_state_id actualState;

void demo_init(void)
{
   actualState = demo_State1;
   event4();
// event 4 triggered when initializing
// the state machine
   event5();
// event 5 triggered when initializing
// the state machine
   event1();
// event 1 triggered when entering
// state 1
   event2();
// event 2 triggered when entering
// state 1
}

void Trigger1(void)
{
    switch(actualState)
    {
      case demo_State1:
// Trigger 1 results in the transition from state 1 to state 2
          actualState = demo_State2;
// event 3 is triggered, when the
// transition from state 1 to state 2
// takes place
          event3();
// event 1 triggered when entering
// state 2
          event1();
          break;
      case demo_State2:
```

```
// event 4 triggered when leaving state
// state 2
          event4();
// Trigger 1 results in the transition from state 2 to state 3
          actualState = demo_State3;
// event 5 is triggered, when the
// transition from state 2 to state 3
// takes place
          event5();
          break;
     default:
          break;
    }
}

void Trigger2(void)
{
    switch(actualState)
    {
      case demo_State2:
// event 4 triggered when leaving state
// state 2
          event4();
// Trigger 2 results in the transition from state 2 to state 1
          actualState = demo_State1;
// event 1 triggered when entering
// state 1
          event1();
// event 2 triggered when entering
// state 1
          event2();
          break;
      case demo_State3:
// Trigger 2 results in the transition from state 3 to state 1
          actualState = demo_State1;
// event 1 triggered when entering
// state 1
          event1();
// event 2 triggered when entering
// state 1
          event2();
          break;
     default:
          break;
    }
}

void Trigger3(void)
{
    switch(actualState)
    {
     default:
          break;
    }
}
```