StateMaker +

MicroGold Software

Why State Diagrams?

The StateMaker state transition program is useful for developers doing real-time or multitask programming who want functions or objects that act simultaneously in a system where modules of code have independent tasks. The state machine allows the programmer functions to have a memory of what it was doing when it leaves the task and later returns. State diagrams are a useful tool for dealing with this concurrency.

Why StateMaker?

StateMaker allows the developer the following advantages for rapid state diagram development:

1. A quick way to implement the state machine
2. A quick way to generate the code.
3. A good hard copy of documentation for the diagram
4. A case-like atmosphere.

Install Procedure

To install StateMaker or the Demo, run the install batch file on the floppy disk.
If you are installing on hard drive C, you should type **Install C:** at the DOS prompt
The program will create directory SM for your program just below the c:\ root directory.

When you have finished installing into DOS you must now install into Windows. Simply go to the **New** menu command in the Program Manager and browse for sm.exe under the SM directory of the drive you installed the program on. The program will be added to your applications group.

As an alternative, you can use the Windows **Setup** program to bring sm.exe into Windows in the same way.

Using StateMaker

The following is a detailed description of how the tool will function:

The Menu: The menu consists of 5 categories for operating StateMaker. They are File, Draw, View, Compile, and List.

File allows the user to create, open and save a file. In addition it will allow the user to print the diagram on the screen. Files must be saved with extension *.sm. Save is available on the palette as well.

Draw - This menu gives you the option of creating states and transitions from the menu. These functions are also available on the palette.

State brings up the state dialog box. The user can pick the name, shape and length of the state here. The allowable range for length of shape is 20 - 100. When the user clicks OK, the state placement tool will appear in the drawing window. The user can then click with the left mouse on the place where the user wishes the shape to appear. The user can then click on the state with the left mouse button and drag the state anywhere in the window. If the user double clicks with the left mouse button on the state, it will bring up the dialog box for that state. The user can then make changes to the name, shape, or length of the state if desired..

Transition allows the user to create a transition arrow for connecting two states together. Choosing the Transition feature in this menu(or on the palette) brings up the Transition Dialog Box. The transition dialog box provides editing for both the event and actions of the transition. To the right of these editing boxes is a reference to the event and action, respectively. The event is the "If" condition of the transition. No *if* is necessary here, however, if you are going to generate C code the condition must be supplied with the various C symbols and variables inside the *if* e.g.

```
( (time == 5 )   &&
(lever == UP) ) ||
(temperature != 30)
```

If the user wishes to use pseudocode instead, simply substitute English statements in place of the conditions:

```
either time reaches 5 seconds and
the lever is pressed  or the temperature
is not 30 degrees.
```

Actions are the *then* part of the *if* statement. The same rules for syntax that applied to the events dialog, also apply to the actions dialog. You also need to terminate actions with a semicolon for single or multiple actions e.g:

```
turnOnHeat();
resetTimer();
```

The assignment for transitioning between states is an action handled by the program.

After the user clicks OK, the transition drawing tool appears. Simply, click the left mouse button on the state you wish to connect the source of the transition to, and drag to the state you wish to be the destination of the transition.

Ref Num - These Numbers are used to reference the transitions in the diagram. They are generated automatically for you, but can be changed if the user wishes.

Events are in the form E #.
Actions are in the form A#.

Transitions are dragged by clicking on the reference text with the left mouse button and pulling the transition to the desired location. The transition can be stretched by clicking on either the head or tail of the transition. Double clicking with the left mouse button on the transition's text will bring up the dialog box for that transition.

Delete On - This Option turns the delete tool on, allowing the user to delete a state or transition from the diagram. When the delete "X" comes up, use the left mouse button to click on the undesired shape. Go to this menu option to toggle the delete mode off again.

View- Zooming is performed in this file option or by the buttons provided on the palette. Zoom in magnifies the diagram, where zoom out shrinks it.

Show Text - This Menu Option allows the user to view the actual text of the transition in the diagram, instead of the references. This option is toggled on and off each time it is selected. Note: The diagram will appear on the print out the same way it appears on the screen.

Show Grid - This Menu Option allows the user to view a grid on the screen. Moving shapes inside this grid will cause the states and transitions to align with the grid points. This option is also toggled on and off upon selection.

Generate- This allows the user to generate C/C++ from the diagram. When the user picks this option, the C code is generated, and the user is prompted with an options dialog. After the user choses the options they wish, the program will generate the code and prompt the user to save the file in a *.c or *.cpp ASCII file. The file can be edited by any ascii editor (e.g. notepad.exe). Upon examining the file, one observes the C code takes the form of a switch statement. The cases have the various state names chosen by the user. Inside the case block is the transitional information for events that trigger the transition, actions that result from triggering a particular transition, and a transitional assignment to the next state.

Options Dialog - This dialog is brought up after the user choses to generate C/C++ code. The dialog allows the user to choose whether or not they want C or C++, and set the respective function name for C and the class and method name for C++.

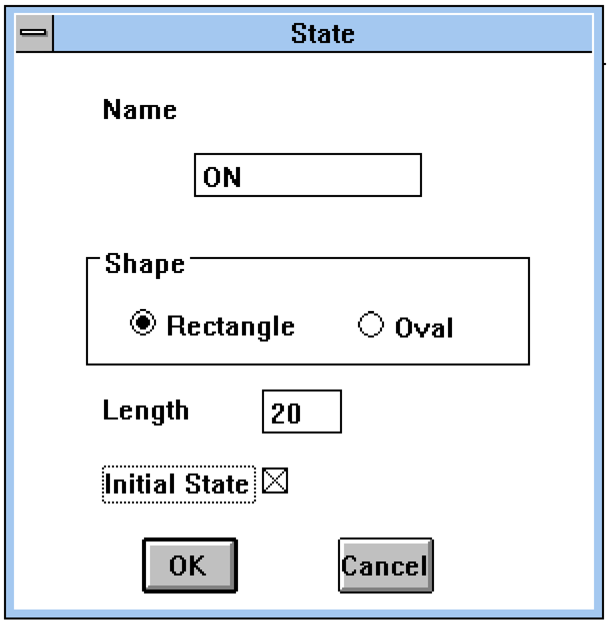
Reverse - This option allows the user to take the generated C or C++ code and turn ones code into a
graphical diagram. If the user writes their own C code in the same switch-case
fashion as StateMaker does, they can then reverse their own code and display it as a
diagram on the screen.

Add Event - This option brings up an event dialog box in which the user can add event constants.
These events will appear as an enumerated member of the type Event.

List- allows the user to generate a XRef list of the transitional information.
Action generates a list of all actions in the state diagram and their references.
Event generates a list of all events and conditions in the state diagram and their references.

The following is an example that takes you step by step through creating a state diagram with StateMaker: The diagram we will produce is a simple on/off switch state machine:

First let's create the states. Click on the state tool located on the upper left hand side of the palette. A state dialog will come up. Enter ON into the text field and click the Rectangle radio button.



Name: ON

Shape: [● Rectangle] ○ Oval

Length: 20

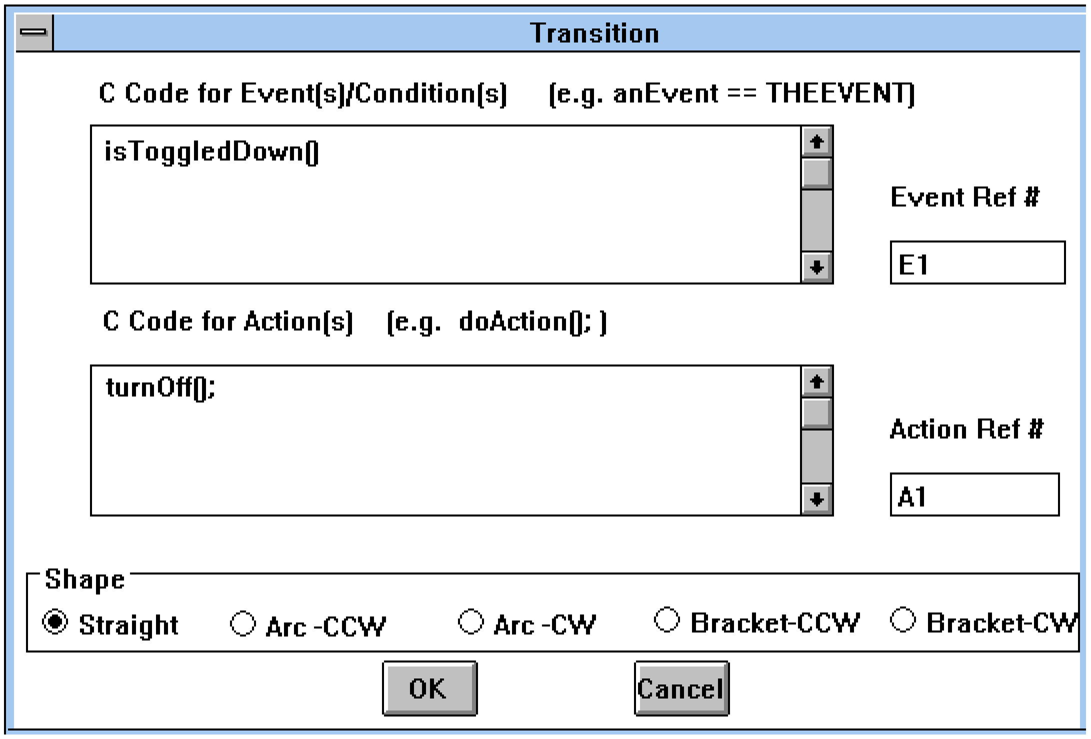
Initial State ☒

Click OK and the ON state will appear in the corner of the window. Drag the state out to center of the window by clicking on the state with the mouse and moving the mouse.

Now repeat the creation process for the OFF state and drag the OFF state underneath the ON state.

Next we will make the transitions going to and from the states. Bring up a transition dialog by clicking on the transition tool on the upper right hand side of the palette. Now fill in the edit fields as shown:

Fig 1.



C Code for Event(s)/Condition(s) (e.g. anEvent == THEEVENT)

```
isToggledDown()
```

Event Ref #: E1

C Code for Action(s) (e.g. doAction();)

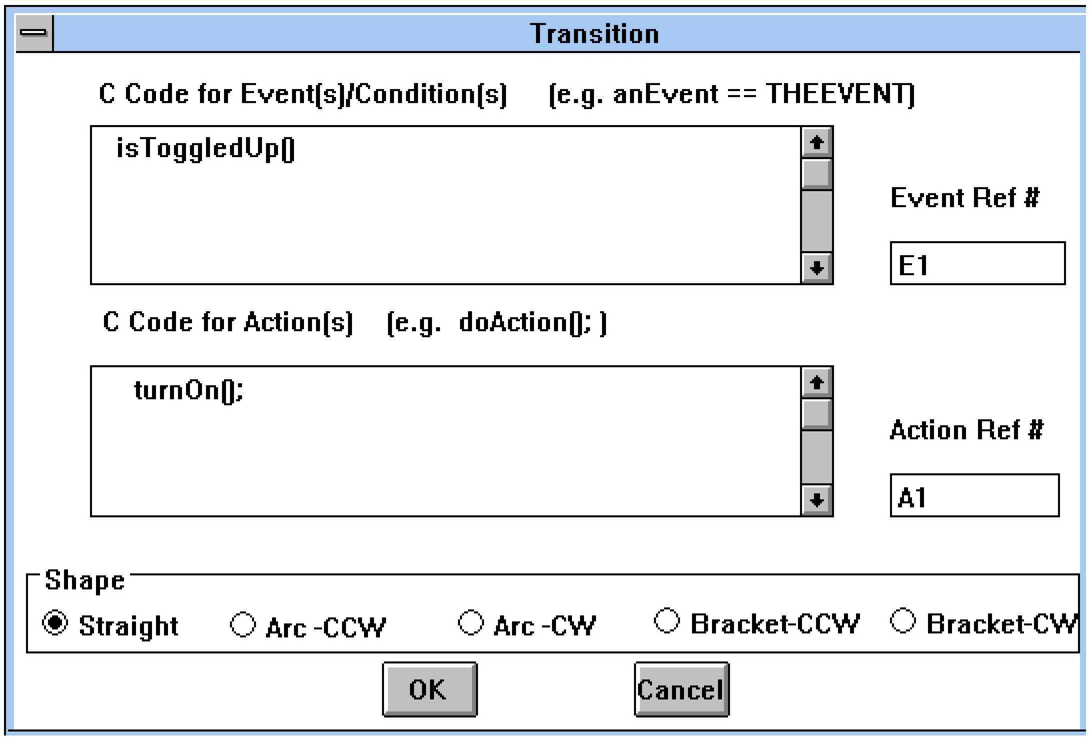
```
turnOff();
```

Action Ref #: A1

Shape: [✓ Straight] Arc -CCW Arc -CW Bracket-CCW Bracket-CW

Click OK and an arrow should appear in the upper left hand corner of the screen. Drag the arrow in between the ON and OFF states by clicking on the reference text in the center of the arrow, and moving the mouse in between the two states. Stretch the arrow to the ON and OFF states by first clicking on the endpoint of the tail of the arrow, and dragging it inside the ON state. Then click on the head of the arrow and drag it inside the OFF state. The arrow will automatically be sized by the program to fit between the states.

Create another transition by filling in the edit fields in the transition dialog as shown:
Fig 2.



C Code for Event(s)/Condition(s) (e.g. anEvent == THEEVENT)

```
isToggledUp()
```

Event Ref #: E1

C Code for Action(s) (e.g. doAction();)

```
turnOn();
```

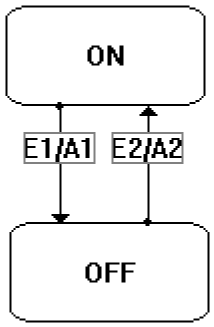
Action Ref #: A1

Shape: [✓ Straight] Arc -CCW Arc -CW Bracket-CCW Bracket-CW

Drag this transition in between the ON and OFF state, and drag the tail to the OFF state and the head of the arrow to the ON state.

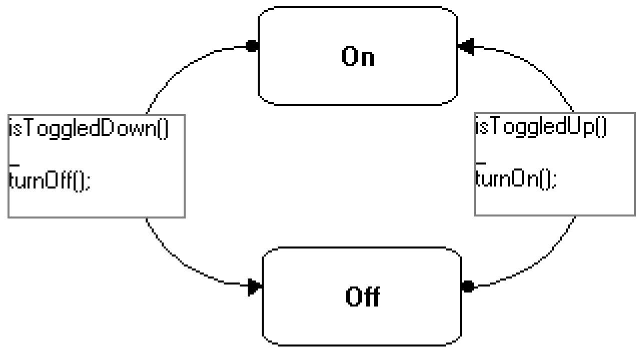
The final diagram will look something like the following:

Fig 3.



By playing with some of the display features you can have the same diagram appear like this:

Fig 4.



You have now finished a simple state diagram. If you own a printer, print out the diagram by going to the file menu and choosing the print option. You will be prompted for a title which will be printed at the top of your diagram.

Save the file by going to the save option in the lower right hand corner of the palette. Save the file as aSwitch.sm.

Compiling aSwitch:

Now that you have drawn your diagram, you will want to generate a C file. Simply choose the generate option in the lower left hand corner of the palette. After the C code is generated, you will be prompted to save the file with extension *.c. Save the file aSwitch.c. If you examine with the editor you should see the following listing:

listing 1.

```
enum Event { NULLEVENT};
enum State { Off, On};

int process(int anEvent)
{
static int aState = Off;

 switch(aState)
 {
 case On:
   if (isToggledDown())
     {
       turnOff();
       aState = Off;
       return aState;
     }

   break;
 case Off:
   if (isToggledUp())
     {
       turnOn();
       aState = On;
       return aState;
     }

   break;

 default:
   return aState;
   }

 return aState;
}
```

Or with C++ option turned on in the Options dialog you generate the following:

listing 2.

```
class LightSwitch
{

public:
enum State { Off, On};
enum Event { NULLEVENT};

private:
State aState;

public:

LightSwitch(State InitState = Off){aState = InitState;}
~LightSwitch(){}
int process(Event anEvent = NULLEVENT);
};


int LightSwitch::process(Event anEvent)
{
 switch(aState)
 {
 case On:
   if (isToggledDown())
    {
      turnOff();
      aState = Off;
      return aState;
    }

   break;
 case Off:
   if (isToggledUp())
     {
       turnOn();
       aState = On;
       return aState;
     }

   break;

 default:
    return aState;
  }

 return aState;
}
```

This code can now be used by a C compiler to generate object code. Note the user needs to provide the C functions such as turnOn(), and isToggledUp(), but once these are accessible by aSwitch.c, the code will compile.

Xreferencing aSwitch

Choose the List option in the menu and pick Event. Save the cross reference as aSwitch.els. The following xref list is produced:

Conditions

E1	isToggledDown()
E2	isToggledUp()

Next choose the List option for Action. Save the cross reference as aSwitch.als. The following xref list is produced:

Actions

A1	turnOff();
A2	turnOn();

<u>**Ordering Info**</u>

To Order Call 1 - 800 - 845 - 9348
Visa/MC accepted.

Or send order to:
MicroGold Software
696 Birch Hill Drive
Bridgewater, NJ, 08807

1 - 908 - 722 - 6438