F. Wagner April 2005

Moore or Mealy model?

Definitions

Mealy and Moore models are the basic models of state machines. A state machine which uses only Entry Actions, so that its output depends on the state, is called a **Moore** model. A state machine which uses only Input Actions, so that the output depends on the state and also on inputs, is called a **Mealy** model. The models selected will influence a design but there are no general indications as to which model is better. Choice of a model depends on the application, execution means (for instance, hardware systems are usually best realised as Moore models) and personal preferences of a designer or programmer. In practise, **mixed** models are often used with several action types.

On an example we will show the consequences of using a specific model. As the example we have taken a Microwave Oven control presented already in another paper on our web site. This time we will make a very thorough analysis of the behaviour exposing the specifics of the model used.

The oven has a momentary-action push button Run to start (apply the power) and a Timer that determines the cooking length. Cooking can be interrupted at any time by opening the oven door. After closing the door the cooking is continued. Cooking is terminated when the Timer elapses. When the door is open a lamp inside the oven is switched on, when the door is closed the lamp is off. During cooking the lamp is also switched on. The cooking period (timeout value) is set by a potentiometer which supplies a voltage to the control system: the voltage is represented by a numeric value 0..4095 which is scaled by the Ni object to 1799. This arrangement allows the maximum cooking time to equal 1799 seconds, i.e. 30 minutes. The solution should also take into account the possibility that the push button Run could get blocked continuously in the active position (which is easy to demonstrate if testing the system in SWLab, which has only the two-positions buttons): in such a case cooking must not start again until it is deactivated when the cooking is terminated (otherwise our meal which we wanted to heat for instance for 5 minutes could be burned until we discover that the button has got stuck in the active position). In other words, each cooking requires intentional activation of the Run button.

The control system has the following inputs:

Run momentary-action push button - when activated starts cooking,

Timer - while this runs keep on cooking,

Door sensor - can be true (door closed) or false (door open).

And the following outputs:

Power - can be true (power on) or false (power off),

Lamp - can be true (lamp on) or false (lamp off).

Moore model

Using Moore model we get a state machine whose state transition diagram is shown in Figure 1. This solution requires 7 states. Figure 2, Figure 3 and Figure 4 show state transition tables for three of those states: Init, Cooking and CookingInterrupted. The state machine uses only Entry actions. Other states can be studied in the provided file MWaveOven_Moore.fsm.

While specifying that state machine the states dominate. We think in the following manner: if the input condition changes the state machine changes its state (if a specific transition condition is valid). Entering the new state, the state machine does some actions and waits for the reaction of the controlled system. In a Moore model the entry actions define effectively the state. For instance, we would think about the state Cooking: it is a state where the Timer runs and the state machines waits for the Timer OVER signal.

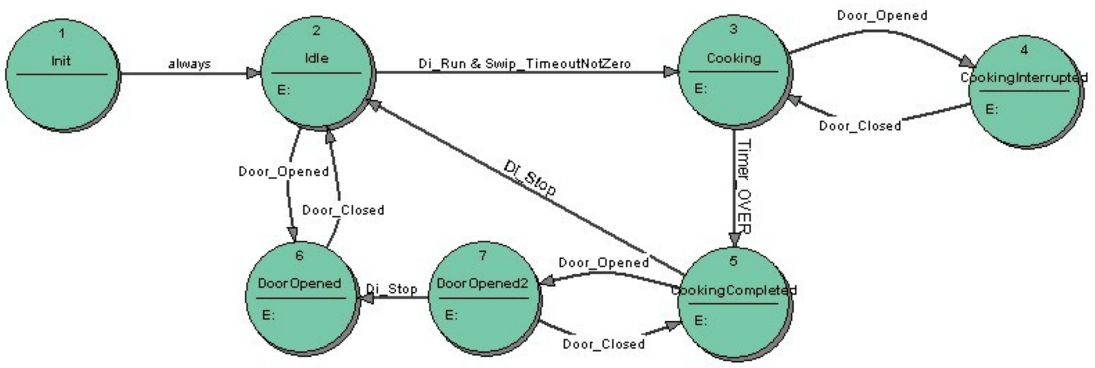
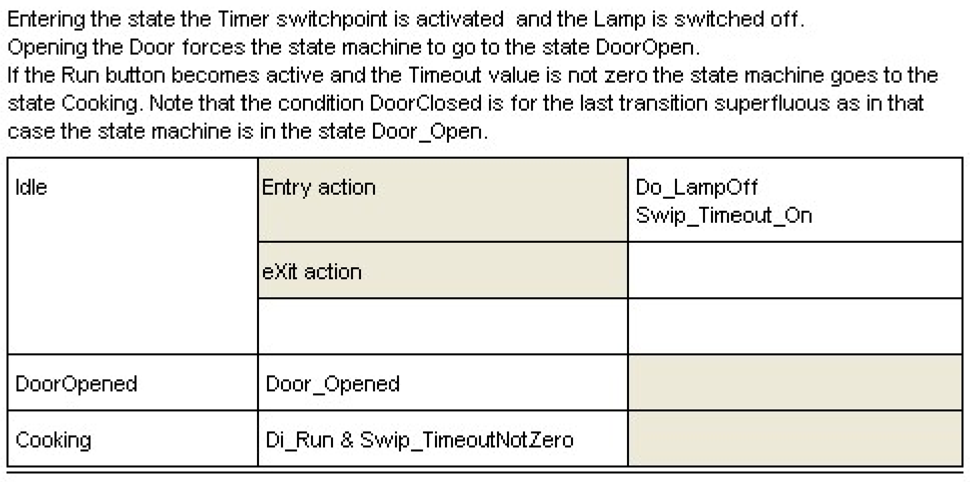


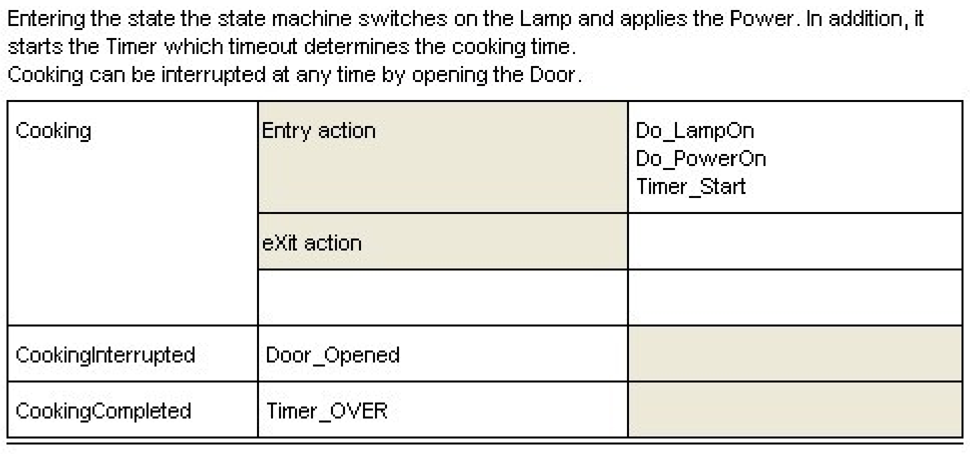
Figure 1 Moore model



Entering the state the Timer switchpoint is activated and the Lamp is switched off.
Opening the Door forces the state machine to go to the state DoorOpen.
If the Run button becomes active and the Timeout value is not zero the state machine goes to the state Cooking. Note that the condition DoorClosed is for the last transition superfluous as in that case the state machine is in the state Door_Open.

Idle	Entry action	Do_LampOff Swip_Timeout_On
	eXit action	
DoorOpened	Door_Opened	
Cooking	Di_Run & Swip_TimeoutNotZero	

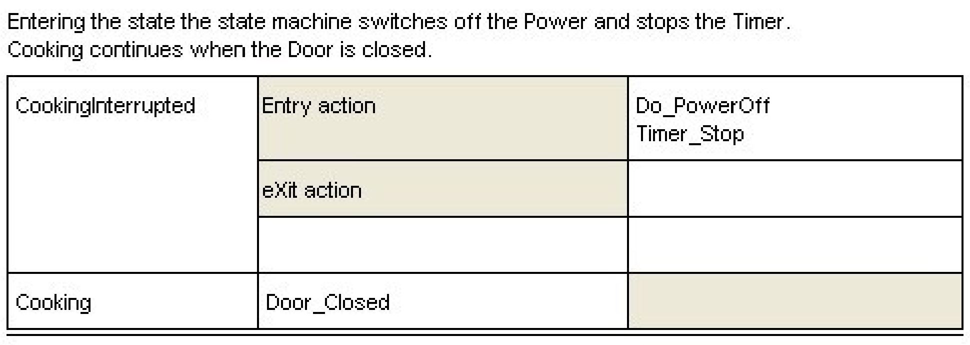
Figure 2 Moore model: state transition table for the state Idle



Entering the state the state machine switches on the Lamp and applies the Power. In addition, it starts the Timer which timeout determines the cooking time.
Cooking can be interrupted at any time by opening the Door.

Cooking	Entry action	Do_LampOn Do_PowerOn Timer_Start
	eXit action	
CookingInterrupted	Door_Opened	
CookingCompleted	Timer_OVER	

Figure 3 Moore model: state transition table for the state Cooking


Entering the state the state machine switches off the Power and stops the Timer.
Cooking continues when the Door is closed.

CookingInterrupted	Entry action	Do_PowerOff Timer_Stop
	eXit action	
Cooking	Door_Closed	

Figure 4 Moore model: state transition table for the state CookingInterrupted

Note the necessity of having two states DoorOpened. The DoorOpened2 is required to keep the information about the Run button: until it stays active in the state CookingCompleted the state machine must not return to the state Idle.

The Moore model seems to be relatively easy to understand due to its simple design philosophy based on states with Entry actions.

Mealy model

The Mealy model is shown in Figure 5. It requires only 5 states. The states: Idle, Cooking and CookingInterrupted for that model (see Figure 6, Figure 7 and Figure 8) illustrate its features. Other states can be studied in the provided file MWaveOven_Mealy.fsm. All activities are done as Input actions, which means that actions essential for a state must be performed in all states which have a transition to that state. The Timer must be now started in both states: Idle and CookingInterrupted. This may be considered as a disadvantage: the functioning becomes a bit confusing.

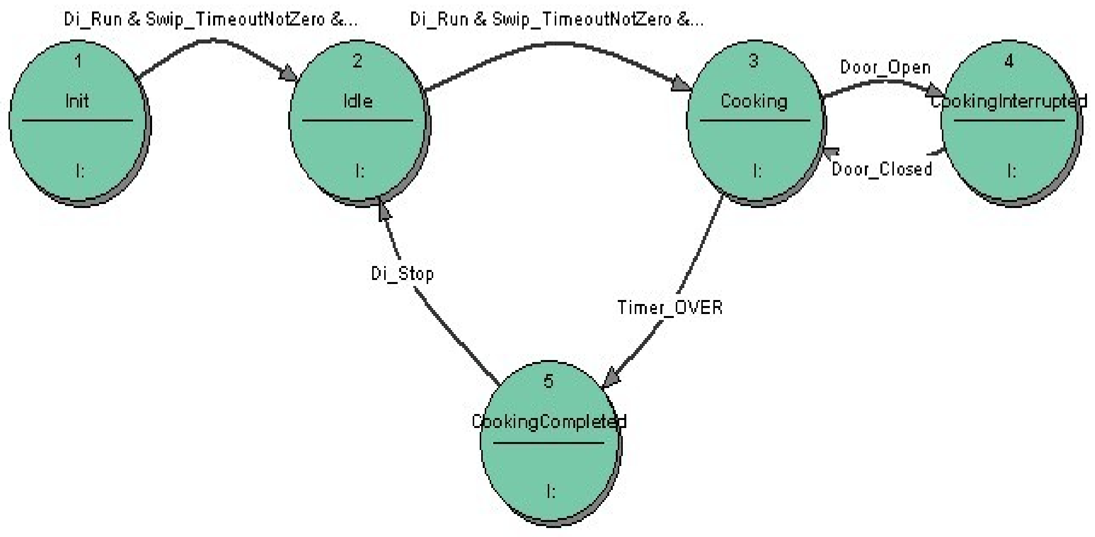
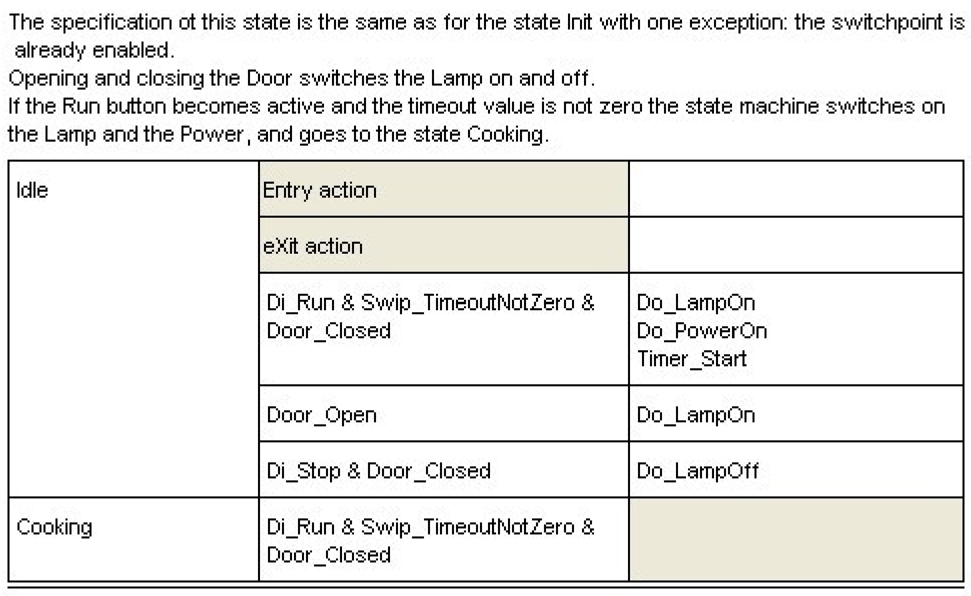


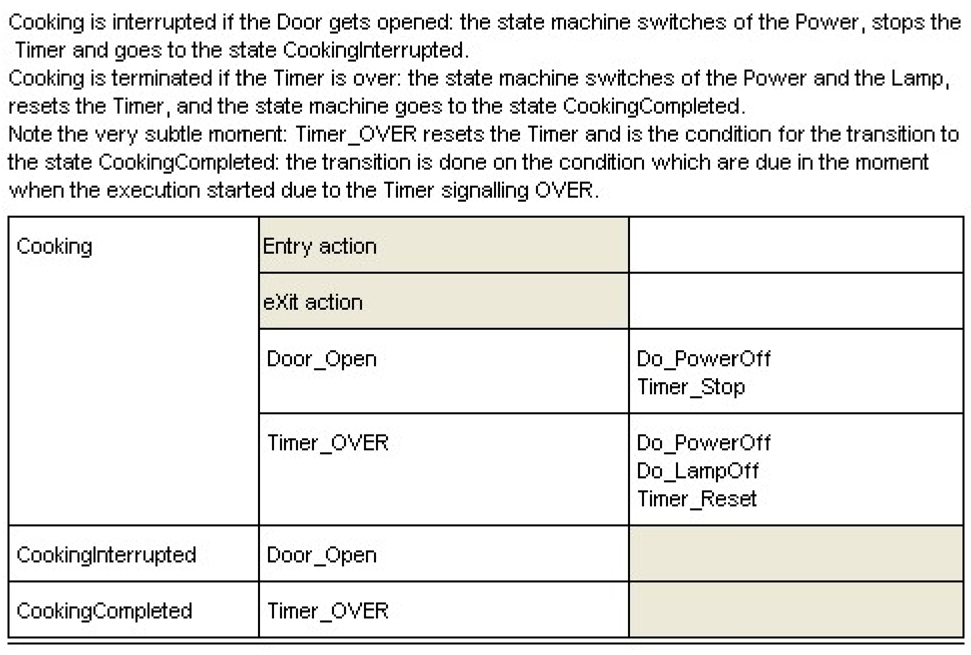
Figure 5 Mealy model

In a Mealy model the inputs dominate our considerations during specification. We think in the following manner: if an input condition changes the state machine reacts to that change performing some action(s); it may also change the state. In a Mealy model the inputs in a given state determine less the present state but rather the next states. For instance, the Timer which decides about the state Cooking must be started in both neighbouring states: Idle and CookingInterrupted. Therefore the meaning of states is more hazy: the state Cooking is determined by the Timer but the Timer control is spread over several states.


The specification ot this state is the same as for the state Init with one exception: the switchpoint is already enabled.
Opening and closing the Door switches the Lamp on and off.
If the Run button becomes active and the timeout value is not zero the state machine switches on the Lamp and the Power, and goes to the state Cooking.

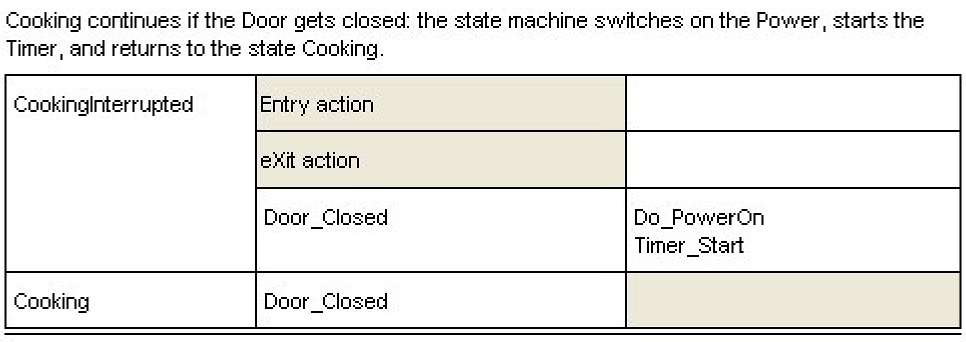
Idle	Entry action	
	eXit action	
	Di_Run & Swip_TimeoutNotZero & Door_Closed	Do_LampOn Do_PowerOn Timer_Start
	Door_Open	Do_LampOn
	Di_Stop & Door_Closed	Do_LampOff
Cooking	Di_Run & Swip_TimeoutNotZero & Door_Closed	

Figure 6 Mealy model: state transition table for the state Idle


Cooking is interrupted if the Door gets opened: the state machine switches of the Power, stops the Timer and goes to the state CookingInterrupted.
Cooking is terminated if the Timer is over: the state machine switches of the Power and the Lamp, resets the Timer, and the state machine goes to the state CookingCompleted.
Note the very subtle moment: Timer_OVER resets the Timer and is the condition for the transition to the state CookingCompleted: the transition is done on the condition which are due in the moment when the execution started due to the Timer signalling OVER.

Cooking	Entry action	
	eXit action	
	Door_Open	Do_PowerOff Timer_Stop
	Timer_OVER	Do_PowerOff Do_LampOff Timer_Reset
CookingInterrupted	Door_Open	
CookingCompleted	Timer_OVER	

Figure 7 Mealy model: state transition table for the state Cooking


Cooking continues if the Door gets closed: the state machine switches on the Power, starts the Timer, and returns to the state Cooking.

CookingInterrupted	Entry action	
	eXit action	
	Door_Closed	Do_PowerOn Timer_Start
Cooking	Door_Closed	

Figure 8 Mealy model: state transition table for the state CookingInterrupted

As a rule, the Mealy model requires fewer states than the Moore model but it seems to be more difficult to understand due to its less strict design philosophy based on a rule: do something in a state but the effects will be used in other state(s).

Mixed model

The mixed model shown in Figure 9 allows us to achieve an optimal solution: we just use the best features of both models: Moore and Mealy. The states Cooking (Figure 10) and CookingInterrupted (Figure 11) correspond to the Moore model. The states Idle and CookingCompleted (Figure 12) is a mixture of both models where we use both: Entry and Input actions. Other states can be studied in the provided file MWaveOven.fsm.

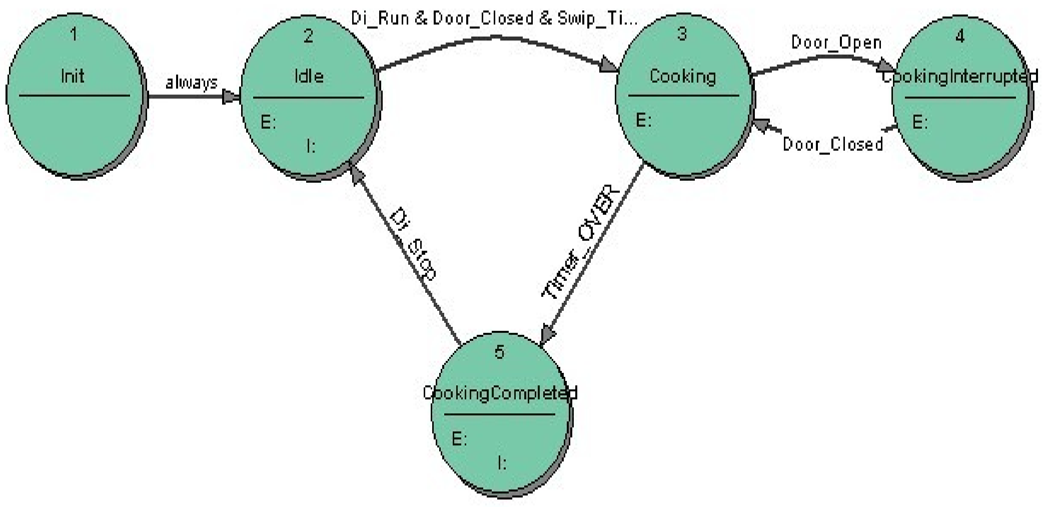
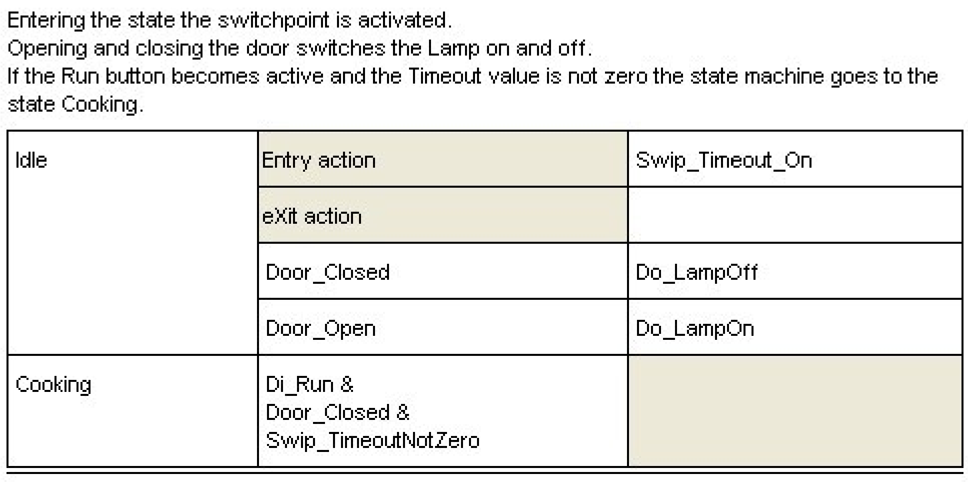


Figure 9 Mixed model

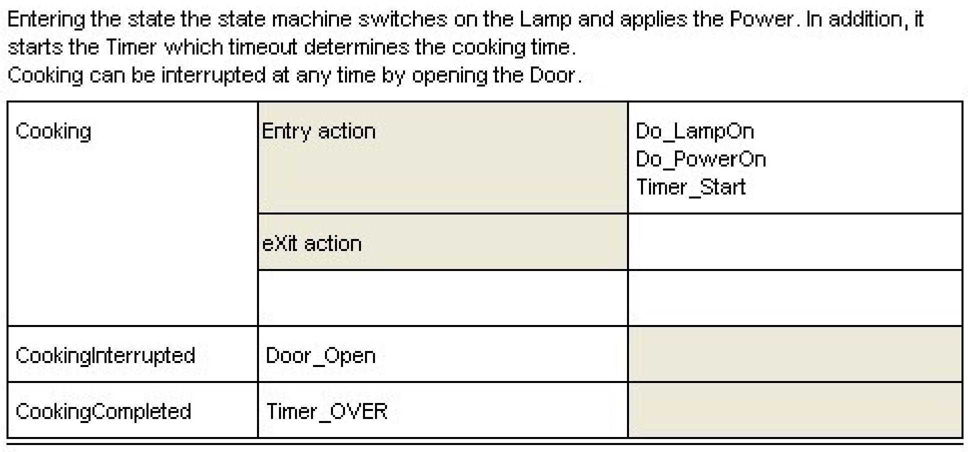
In the example, we use essentially a Moore model completed by Input Actions in the states: Idle and CookingCompleted. That small change in comparison with the original Moore model shown in Figure 1 gives amazing improvements: the reduction from 7 to 5 states. Considering the complexity both state machines: Moore and mixed model are similar, the mixed model having the advantage of fewer states.

On the other hand, though the Mealy model in Figure 5 and the mixed model have the same number of states, they are very different. Comparing the Mealy model with the mixed model we notice an essential simplification in favour of the mixed model.


Entering the state the switchpoint is activated.
Opening and closing the door switches the Lamp on and off.
If the Run button becomes active and the Timeout value is not zero the state machine goes to the state Cooking.

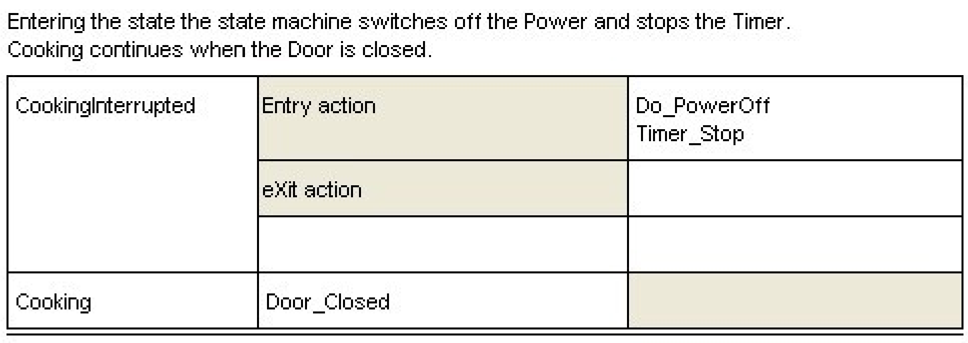
Idle	Entry action	Swip_Timeout_On
	eXit action	
	Door_Closed	Do_LampOff
	Door_Open	Do_LampOn
Cooking	Di_Run & Door_Closed & Swip_TimeoutNotZero	

Figure 10 Mixed model: the state transition table for the state Idle


Entering the state the state machine switches on the Lamp and applies the Power. In addition, it starts the Timer which timeout determines the cooking time.
Cooking can be interrupted at any time by opening the Door.

Cooking	Entry action	Do_LampOn Do_PowerOn Timer_Start
	eXit action	
CookingInterrupted	Door_Open	
CookingCompleted	Timer_OVER	

Figure 11Mixed model: the state transition table for the state Cooking


Entering the state the state machine switches off the Power and stops the Timer.
Cooking continues when the Door is closed.

CookingInterrupted	Entry action	Do_PowerOff Timer_Stop
	eXit action	
Cooking	Door_Closed	

Figure 12Mixed model: the state transition table for the state CookingInterrupted

Conclusions

By introducing the Exit Actions (which are not known in Mealy and Moore models) we still expand the possible specification means. Of course, we should not overemphasize the use of different action types: the comprehensibility may suffer. Therefore, the rule we suggest is: use a Moore model supported by Input Actions in obvious situations. The Exit actions should be treated as an auxiliary feature to be used sometimes in situations where they do not influence strongly the state machine behaviour. For instance, we may stop a timer as an Exit Action but we should never start a timer as an Exit Action.

The control of a Microwave oven is of course only a simple example. Unfortunately, it is rather difficult to define some suggestions based on theoretical considerations which would prove the advantages of one model against another one. Experience over several years tends to show that a mixture of both models produces the best results and the discussed Microwave oven control is a nice example supporting this thesis.

If the specified state machine is to be coded the model used has enormous influence on the program quality. A Moore model is very easy to code, the transition may be often implemented just by constants as initialized tables. The Mealy model opens the Pandora box: the program becomes so complex that we lose the state machine in the confusing code.