14.3.2 Write Data, Call Methods, Respond to Events

With the COM library, you can, in addition to reading values, change values in Plant Simulation, invoke methods and respond to events from Plant Simulation. The following scenario is intended to serve as an example: In Excel data are defined for the simulation. By a click on a button, these data are entered in Plant Simulation and the simulation begins. After the end of the simulation, the simulation results are entered into the Excel spreadsheet.

VBA class module—event handling

The event handler must be created in a VBA class module (as an internal method). This requires a special approach. First, create a class module in the VBA editor (Insert—Class module). Rename the class as "ps" (Properties Window—(name)). In the class, you must define an event handler. All events will be sent to this object. Therefore, write the following statement in the head of the class module:

```
Public WithEvents ps As eMPlantLib.remotecontrol
```

The object variable will be initialized via the constructor. To do this, add the following method to the class module:

```
Private Sub Class_Initialize()
  Set ps = New remotecontrol
End Sub
```

Create the table from Excel (Sheet1).

	A	B	C	D	E	F	G
1		results				process time (sec)	
2		waiting	working	blocked		M1	
3	M1	0%	0%	0%		M2	10
4	M2	0%	0%	0%			
5						start simulation	

Fig. 14.9 Example Excel table

You also need a frame in Plant Simulation like in Fig. 14.10.

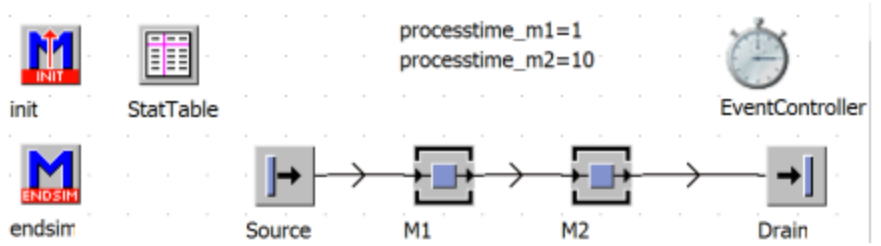

Fig. 14.10 Example frame

Machines M1 and M2 each have an availability of 95 per cent and an MTTR of one minute. Set the interval of the source to zero. Create the table StatTable according to Fig. 14.11.

	string 0	real 1	real 2	real 3
string		waiting	working	blocked
1	M1			
2	M2			

Fig. 14.11 Table StatTable

The following procedure is to be programmed:

1. By clicking on the button Start Simulation, Plant Simulation is started and the model is loaded; the processing times of M1 and M2 are transferred to Plant Simulation. The simulation is initialized (reset) and started.
2. The init method sets the process times of the machines.

3. EndSim writes the statistical data into the table StatTable and sends an event to Excel that the simulation is complete.
4. The event handler in Excel writes the results of the simulation into the Excel spreadsheet and terminates the simulation (and closes Plant Simulation).

The following methods of the COM interface are required in VBA for the next steps (Table 14.3):

Table 14.3 Methods of the class RemoteControl

Methods	Description
<rc>.LoadModel(path)	Opens a Plant Simulation model
<rc>.getValue(ps_path)	Reads from the open Plant Simulation Model values with the path ps_path
<rc>.setValue ps_path, value	Sets values in Plant Simulation
<rc>.ResetSimulation (eventController)	Resets the simulation
<rc>.startSimulation (eventController)	Starts the simulation
<rc>.quit	Closes Plant Simulation, changes are not saved

For 1.), assign a macro to the button in Excel: Insert the following programming (you will have a different name for the method; note the declaration of pss outside of the method).

```
Dim pss As ps
Sub Schaltfläche1_KlickenSieAuf()
Set pss = New ps
pss.ps.LoadModel ("D:\com_1_en.spp")
'set values
pss.ps.SetValue ".Models.Frame.processtime_m1",_
                  Range("G2").Value
pss.ps.SetValue ".Models.Frame.processtime_m2",_
                  Range("G3").Value
'start simulation
pss.ps.ResetSimulation(".Models.Frame.EventController")
pss.ps.StartSimulation
".Models.Frame.EventController")
End Sub
```

To 2.) Init method of the Plant Simulation model:

```
m1.procTime:=num_to_time(processtime_m1)
m2.procTime:=num_to_time(processtime_m2)
```

To 3.) Method endSim of the Plant Simulation model:

```
--write statistics
statTable[1,1]:=m1.statWaitingPortion
statTable[1,2]:=m2.statWaitingPortion
```

```
statTable[2,1]:=m1.statworkingPortion
statTable[2,2]:=m2.statWorkingPortion
statTable[3,1]:=m1.statBlockingPortion
statTable[3,2]:=m2.statBlockingPortion
fireSimTalkMessage("finish")
```

For 4.), the event handler must be programmed in the VBA class module. To do this, select the module sheet in the head with ps on the left and SimTalkMessage on the right. The method ps_SimTalkMessage takes as arguments the string that you pass via SimTalk with **fireSimTalkMessage**.

```
Private Sub ps_SimTalkMessage(ByVal text As String)
Dim x As Integer
Dim y As Integer
Dim i As Integer
Dim k As Integer
If text = "finish" Then
  x = ps.GetValue(".models.frame.statTable.xDim")
  y = ps.GetValue(".models.frame.statTable.yDim")
  For i = 1 To y
    For k = 1 To x
      Tabelle1.Cells(i + 2, k + 1).Value =
       ps.GetValue(".models.frame.statTable[" +
         CStr(k) + "," + CStr(i) + "]")
    Next
  Next
  ps.Quit
End If
End Sub
```