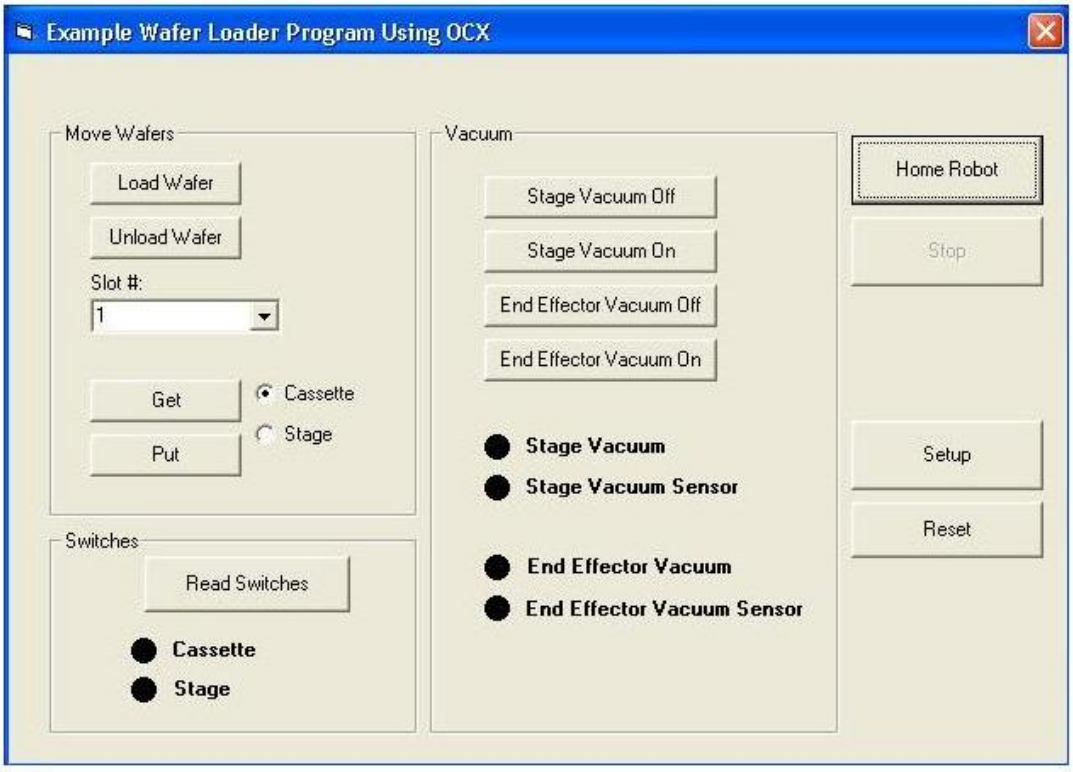
Franklin MCI
Franklin Mechanical & Control Inc

Description of the methods and properties of the Franklin ActiveX control

For those who need to integrate loader functionality within their own system software, this document is provided as a basic reference to the FranklinOCX ActiveX control and example VB6 source code provided by Franklin MCI. The example source code together with this document illustrates how to programmatically control the Franklin Loader from an external Windows application that incorporates the FranklinOCX ActiveX control.

Shown below is a screen shot of the supplied example software. This program invokes FranklinOCX methods to load and unload wafers, read the state of the cassette and stage switches, to control vacuum, and read vacuum sensors. The example program also allows access to the system setup panel.



This example program is written in Visual Basic 6. As such, all code examples within this document are given in VB6. Interfacing to the OCX, however, is possible with any development language that supports ActiveX controls.

Variable syntax in this manual follows VB6's short syntax for declaration of variables.

var1& is equivalent to '*var1 as Long*' (In VB6 Long is a 32 bit signed integer)
var1$ is equivalent to '*var1 as String*'

Procedures of the OCX control:

This is a partial list of the OCX procedure calls and properties grouped by relevance along with a brief description of each. A list of all methods and properties available are listed at the end of this document.

System initialization procedures.

Upon every power-up of the system, the following three procedures, ReadInitFile, InitCommPorts, and HomeLoader must be invoked before normal operation can commence. These must be performed in the order listed.

Procedure: ReadInitFile
Syntax: LoaderControl1.ReadInitFile(arg1$, arg2&, arg3$)
Use: Causes the OCX control to read the contents of the handler system initialization file.

Example: Call LoaderControl1.ReadInitFile(a$, er1&, error1$)

Where: a$, is set to the fully qualified path and filename of the initialization file. er1& returns "0" if no error, "1" if an error was encountered. error1$ returns an error description.

Procedure: InitCommPorts
Syntax: LoaderControl1.InitCommPorts(arg1&, arg2$)
Use: Causes the OCX control to open and initialize the serial communications ports between the computer and handler system. No arguments are passed to the OCX.
If the port(s) cannot be opened and/or initialized, an error is generated.

Example: Call LoaderControl1.InitCommPorts(er1&, error1$)

Where: er1& returns "0" if no error, "1" if an error was encountered.
error1$ returns an error description.

Procedure: HomeLoader
Syntax: LoaderControl1.HomeLoader(arg1&, arg2$)
Use: Causes the handler system to position all of the various motors to their "home" positions, making the system ready for use. At the completion of the homing sequence, the OCX enters an idle state awaiting commands from the controlling computer. If the system does not initialize properly, an error is generated.

Example: Call LoaderControl1.HomeLoader(er1&, error1$)

Where: er1& returns "0" if no error, "1" if an error was encountered.
error1$ returns an error description.

Optional initialization procedures

Procedure: SetLoaderInit
Syntax: SetLoaderInit(arg1&, arg2$)
Use: Provides a means of setting an initialization flag that remains set as long as the handler system remains powered up. This procedure proves most valuable when developing custom software for controlling the handler system. When used in conjunction with the GetLoaderInit property (shown below), developers can save them selves much time by not having to rerun the HomeLoader initialization procedure every time the application being developed is started or restarted. This is accomplished by simply reading the state of the initialization flag by using the GetLoaderInit property and if the property is returned as true, then the system has already gone through the homing procedure and the HomeLoader procedure can be skipped over or otherwise ignored.

Example: SetLoaderInit(er1&, error1$)

Where: er1& returns "0" if no error, "1" if an error was encountered.
error1$ returns an error description.

Property: GetLoaderInit
Syntax: arg1& = GetLoaderInit
Use: Provides a means of accessing the user settable initialization variable.

Where: arg1& = 1 if the initialization variable has been set. 0 if not, or the handler has lost power after the initialization flag has been previously set.

Note that SetLoaderInit and GetLoaderInit have no bearing on the operation of the handler system in any way and the use of which is completely optional to the application developer.

Fully automated sample exchange. The next two procedures allow for fully automatic operation of the handler, specifically in the operations of loading and unloading a sample to and from the inspection stage, respectfully.

Procedure: LoadWafer
Syntax: LoaderControl1.LoadWafer(arg1&, arg2&, arg3$, arg4&)
Use: Causes the handler system to fetch a sample from a specified slot of a loaded cassette and deliver it to the vacuum chuck located on the inspection stage. The system then enters an idle state. If the operation fails, an error is generated.

Example: Call LoaderControl1.LoadWafer(slot&, er1&, error1$, wafer_state&)

Fully automated sample exchange cont.

Where: slot& is set equal to the desired slot to fetch the sample from. The slot range is 1 to 25. Requesting a sample outside this range causes an error.
er1& returns “0” if no error, “1” if an error was encountered.
error1$ returns an error description.
wafer_state& returns an additional error code. This code has to do with vacuum detection errors and can be generated at various points of the loading operation. Values for these codes are listed elsewhere.

Procedure: UnloadWafer
Syntax: LoaderControl1.UnloadWafer(arg1&, arg2$, arg3&)
Use: Causes the handler system to retrieve a sample previously loaded to the inspection stage by using the “LoadWafer” command shown above. Note there is no provision to provide a slot number to return the sample to. This is because the OCX keeps track of the previously loaded sample location within the cassette it was removed from and thus, will be returned to the same slot.

Example: Call LoaderControl1.UnloadWafer(er1&, error1$, wafer_state&)

Where: er1& returns “0” if no error, “1” if an error was encountered.
error1$ returns an error description.
wafer_state& returns an additional error code. This code has to do with vacuum detection errors and can be generated at various points of the loading operation. Values for these codes are listed elsewhere.

Other automation procedures. The following three procedures offer partial automation of the handler system and are useful in the event of the necessity of removing a sample from the system or to set the system to its “change cassette” state.

Procedure: GetWafer
Syntax: LoaderControl1.GetWafer(arg1&, arg2&, arg3$, arg4&)
Use: Provides a means to fetch a sample from either the cassette or stage, bring it to an idle position (end effector retracted position) and enter an idle state. This is useful in an instance where a test, measurement or inspection has failed and the sample needs to be removed from the cassette or stage. Note that this operation does not relinquish the vacuum at the end effector. Vacuum may be released using the SetEndEffectorVacuum procedure if needed. If the operation fails, an error is generated.

Example: LoaderControl1.GetWafer(slot&, er1&, error1$, wafer_state&)

Other automation procedures cont.

Where: slot& is set either to “0” or a value of 1 to 25, which represents a slot of the loaded cassette to fetch a sample from. Setting the slot& argument to a value higher than 25 will cause an error. If slot& is set to “0”, then the operation is to fetch the sample from the stage otherwise, the sample is fetched from the loaded cassette.

Procedure: PutWafer
Syntax: LoaderControl1.PutWafer(arg1&, arg2&, arg3$, arg4&)
Use: Provides a means of returning a previously removed sample to either the cassette or stage.

Example: LoaderControl1.PutWafer(slot&, er1&, error1$, wafer_state&)

Where: slot& is set either to “0” or a value of 1 to 25, which represents a slot of the loaded cassette to return a sample to. Setting the slot& argument to a value higher than 25 will cause an error. If slot& is set to “0”, then the operation is to return the sample to the stage otherwise, the sample is returned to the loaded cassette.

Please note: It is not recommended to manually attempt to “re-insert” a sample to the system at any time by attempting to manually position it on the end effector. Because the samples need to be centered on the end effector with a relatively high amount of accuracy, doing so could cause the system to crash the sample when attempting to return it to the cassette causing possible damage to the sample or other items.

Procedure: SetReadyForLoad
Syntax: LoaderControl1.SetReadyForLoad(arg1&, arg2$)
Use: Causes the handler system to remove a sample from the inspection stage (if a sample has been previously loaded to the stage) and/or return the handler robot and elevator to the “exchange cassette” position. The “exchange cassette” position is such that the robot is turned towards the stage and the elevator is returned to its upper most position. This allows easy changing of the cassette.

Example: Call LoaderControl1.SetReadyForLoad(er1&, error1$)

Where: er1& returns “0” if no error, “1” if an error was encountered.
error1$ returns an error description.

End effector and stage chuck vacuum manipulation procedures.

The following two procedures provide a means of turning on or off the vacuum at the end effector and inspection stage chuck. While not normally needed during typical fully automatic operation, a need may arise when the setting or resetting of the vacuum at the end effector or stage may be necessary. Such a situation might be one of a need to remove a sample from the system.

Procedure: SetEndEffectorVacuum
Syntax: LoaderControl1.SetEndEffectorVacuum(arg1&, arg2&, arg3$)
Use: Provides a means to turn on/off the vacuum at the system end effector. If the operation fails, an error is generated. This procedure does not return the state of the vacuum sensor.

Example: Call LoaderControl1.SetEndEffectorVacuum(state&, er1&, error1$)

Where: state& is set to either “1” or “0” to turn on or off respectively the vacuum at the end effector. er1& returns “0” if no error, “1” if an error was encountered.
error1$ returns an error description.

Procedure: SetStageVacuum
Syntax: LoaderControl1.SetStageVacuum(arg1&, arg2&, arg3$)
Use: Provides a means to turn on/off the vacuum at the vacuum chuck mounted on the inspection stage. If the operation fails, an error is generated. This procedure does not return the state of the vacuum sensor.

Example: Call LoaderControl1.SetStageVacuum(state&, er1&, error1$)

Where: state& is set to either “1” or “0” to turn on or off respectively the vacuum at the vacuum chuck. er1& returns “0” if no error, “1” if an error was encountered.
error1$ returns an error description.

Reading vacuum sensors, cassette elevator and stage switches.

Procedure: ReadVacuumSensors
Syntax: LoaderControl1.ReadVacuumSensors(arg1&, arg2$, arg3&, arg4&, arg5&)
Use: Causes the system to read the state of the vacuum sensors and provides a means of determining the state of vacuum on the end effector and stage vacuum chuck. If the operation fails, an error is generated.

Example: LoaderControl1.ReadVacuumSensors(er1&, error1$, end_effector&, stage&, code&)

Where: end_effector& and stage& will be either “1” if vacuum is detected or “0” if not. code& returns a decimal value of “10” if the end effector has vacuum + “1” if the stage chuck has vacuum. Thus a value of “11”indicates both the end effector and stage vacuum have been detected. er1& returns “0” if no error, “1” if an error was encountered. error1$ returns an error description.

Procedure: ReadCassetteSwitch
Syntax: LoaderControl1.ReadCassetteSwitch(arg1&, arg2&, arg3$)
Use: Provides a means of detecting whether or not a cassette is loaded onto the cassette elevator.
Example: LoaderControl1.ReadCassetteSwitch(code&, er1&, error1$)

Misc. procedures

Procedure: ShowSetup
Syntax: LoaderControl1.ShowSetup
Use: Provides a means of accessing the system “Setup” panel. No arguments are passed when calling this procedure and there is no error generation provided. The setup panel is used during the system setup procedure and once the procedure is completed, generally there is no need to make any further adjustments. There may be instances however, when the setup panel might need to be accessed. *Because the nature of the setup panel is one where almost every aspect of the system is defined, it is recommended that some provision for a pass worded environment be provided. Unauthorized users tampering with the settings of the setup panel could prove to have disastrous results to the system and or product.*

Example: Call LoaderControl1.ShowSetup

Misc. procedures cont.

Procedure: StopLoader
Syntax: LoaderControl1.StopLoader
Use: Provides a means of halting the operation of the handler system

Example: Call LoaderControl1.StopLoader

Procedure: WaitTimer
Syntax: LoaderControl1.WaitTimer(arg1&)
Use: Provides a simple means of introducing a programmable delay within an application during runtime of the application.

Example: LoaderControl1.WaitTimer(delay_value&)

Where: delay_value& is set to a value of milliseconds the delay should occur.

Explanation of the wafer_state& error code variable.

1 No sample detected on pickup at cassette (empty slot)
2 End effector vacuum sensor shows sample already present
3 Stage vacuum sensor shows sample already present
4 End effector vacuum sensor shows active after release of vacuum
5 Unused
6 No stage vacuum detected after sample drop off

OCX Methods

CheckCommunicationElevator
CheckCommunicationMDrive
CloseCommPorts
GetWafer
HomeLoader
InitCommPorts
InitializeVacuum
LoadWafer
PutWafer
RaiseCode
ReadInitFile
ReadCassetteSwitch
ReadStageSwitch
ReadVacuumSensors
SetEndEffectorVacuum
SetLoaderInit
SetReadyForLoad
SetSolenoid
SetStageVacuum
ShowSetup
StopLoader
UnloadWafer
WaitTimer

OCX Properties

Get CassetteLoaded
Get EndEffectorVacuum
Get EndEffectorVacuumSensorState
Get GetLoaderInit
Get LoaderState
Get LoaderWaferSize
Get StageVacuum
Get StageVacuumSensorState
Get StageSwitch
Get SlotNumber

Let LoaderWaferSize
Let SlotNumber

Provided as preliminary documentation only.
Franklin MCI 01.28.10