IITM-DTT

Antidoc v2.0.5, Abinash S

Table of Contents

Chapter 1. Project description

No description found (add content in project description)

Chapter 2. My Computer

2.1. DQMH® modules

This section describes DQMH® module responsibilities and relationships.

2.1.1. Preamble

A DQMH module is the main component of an architecture based on DQMH® framework. A DQMH module is used to implement a section of the application that has one responsibility.

DQMH® framework defines two different type of DQMH module.

Singleton:

A Singleton DQMH module can have only one instance running at any given time.

Cloneable:

A Cloneable DQMH module can have one or multiple instances running in parallel.

DQMH® framework defines two different ways to carry data throughout the application and with both other DQMH modules and non-DQMH based code.

Request events:

A request is a code that fires an event requesting the DQMH module to do something. Multiple locations in the code can send events to the DQMH module.

Request events are many-to-one.

Requests are usually named using imperative tense.

Broadcast events:

A broadcast is a code that fires an event broadcasting that the DQMH module did something. Multiple Event Structures can register to handle the Broadcast Events.

Broadcast Events are one-to-many.

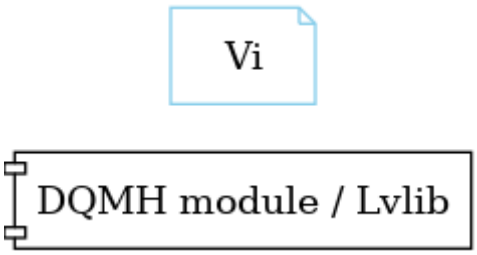
Broadcasts are usually named using past tense or passive voice.

NOTE	Refer to the DQMH® framework official documentation to find more details on how the framework works

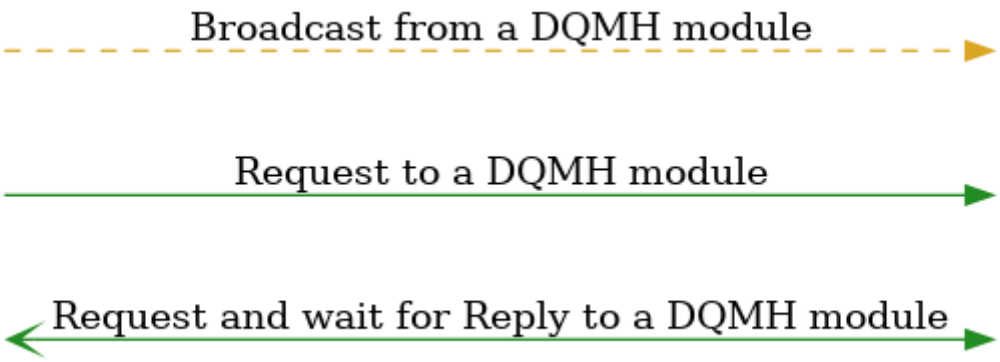
The following section gives you details on the project architecture relying on this framework. It gives you an overview of the modules' interaction and detailed information on each module.

Graphs used in this section have the following legend:

Components:

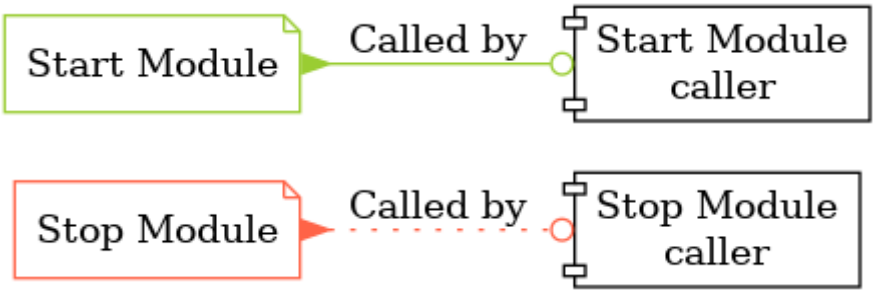


Events:



NOTE	One arrow can represent one or more events between two components
NOTE	Request and Request and wait for Reply are represented by only one arrow. If there is no Request and wait for Reply, Request representation is used. Otherwise Request and wait for Reply is used

Start and Stop module callers:



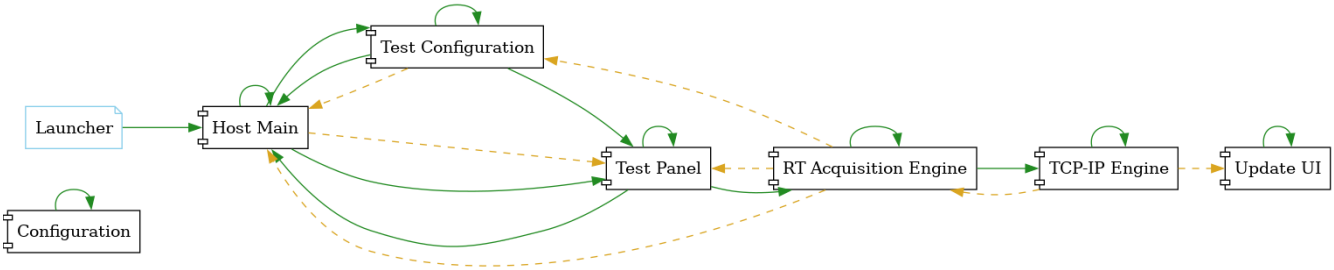
2.1.2. Modules overview

This project contains 5 singleton modules and 2 cloneable modules.

Table 1. Modules list

Singleton	Cloneable
Configuration.lvlib	RT Acquisition Engine.lvlib
Host Main.lvlib	TCP-IP Engine.lvlib
Test Configuration.lvlib	
Test Panel.lvlib	
Update UI.lvlib	

This graph represents the links between all DQMH modules.



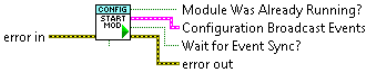
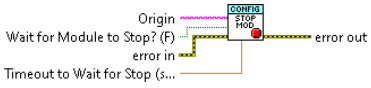
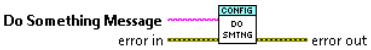
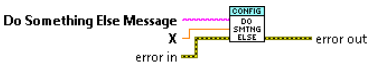
2.1.3. Configuration.lvlib

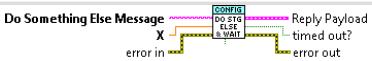
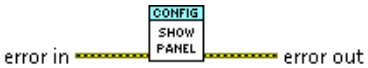
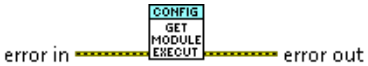
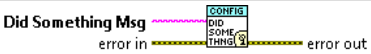
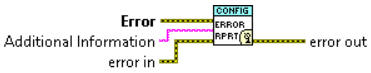
Type: Singleton

Responsibility: This Module's is used to configure all the channel's present the Realtime target.

Event list

Table 2. Events

Name	Type	Connector pane	Description	S.	R.	I.
Start Module			Launches the module Main VI. After calling this VI, you can optionally register for broadcast events from the module by wiring the broadcast events output of this VI to a Register For Events function. After the optional Register For Events function call, you should always call the Synchronize Module Events.vi for this module with the 'Wait for Event Sync?' output of this VI to the corresponding input of the Synchronize Module Events.vi. To see an example of the proper wiring pattern, see the "Start Module: Value Change" event frame in the API Tester VI for this module.			
Stop Module			Send the Stop request to the Module's Main.vi. If Wait for Module to Stop? is TRUE, this VI will wait until the module main VI stops, and will timeout at the Timeout to Wait for Stop value. This value defaults to "-1", which means the VI will not timeout, and will always wait until the module main VI stops before completing execution. Note: The Timeout to Wait for Stop value is ignored if 'Wait for Module to Stop?' is set to FALSE.			
Do Something			Send the Do something request to the Module's Main.vi.			
Do Something Else			Send the Do Something Else request to the Module's Main.vi.			

Name	Type	Connector pane	Description	S.	R.	I.
Do Something Else and Wait for Reply			Send the Do Something Else request to the Module's Main.vi.			
Show Panel			Send the Show Panel request to the Module's Main.vi.			
Hide Panel			Send the Hide Panel request to the Module's Main.vi.			
Get Module Execution Status			Fire the Get Module Execution Status request.			
Show Diagram			This VI tells the Module to show its block diagram to facilitate troubleshooting (add probes, breakpoints, highlight execution, etc).			
Module Did Init			Send the Module Did Init event to any VI registered to listen to this module's broadcast events.			
Did Something			Send the Did Something event to any VI registered to listen to this module's broadcast events.			
Status Updated			Send the Status Updated event to any VI registered to listen to events from the owning module.			
Error Reported			Send the Error Reported event to any VI registered to listen to events from the owning module.			
Module Did Stop			Send the Module Did Stop event to any VI registered to listen to this module's broadcast events.			
Update Module Execution Status			Broadcast event to specify whether or not the module is running.			

Type: → Request | → Request and Wait for Reply | → Broadcast

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

Module relationship

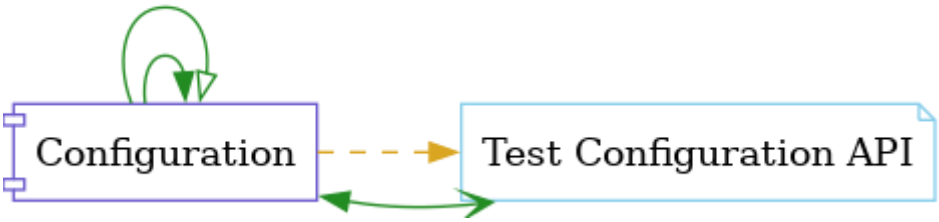


Table 3. Requests callers

Request Name	Callers
Do Something Else and Wait for Reply	Test Configuration API.vi
Do Something Else	Test Configuration API.vi
Do Something	Test Configuration API.vi
Get Module Execution Status	Configuration.lvlib:Obtain Broadcast Events for Registration.vi Configuration.lvlib:Start Module.vi
Hide Panel	Test Configuration API.vi
Show Diagram	Test Configuration API.vi
Show Panel	Test Configuration API.vi

Table 4. Broadcasts Listeners

Broadcast Name	Listeners
Did Something	Test Configuration API.vi
Error Reported	Test Configuration API.vi
Module Did Init	Test Configuration API.vi
Module Did Stop	Test Configuration API.vi
Status Updated	Test Configuration API.vi
Update Module Execution Status	Test Configuration API.vi

Table 5. Used requests

Module	Requests
Configuration.lvlib	Stop Module.vi

Table 6. Registered broadcast

Module	Broadcasts
—	—

Module Start/Stop calls

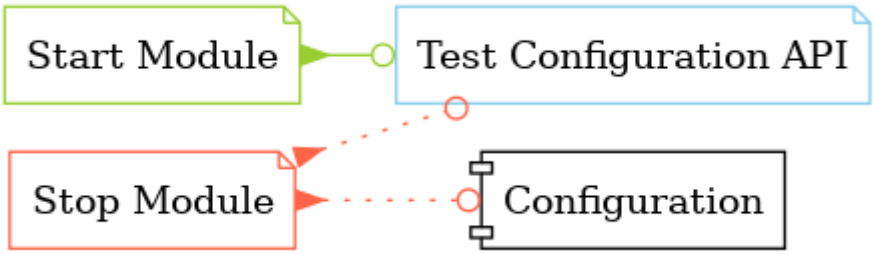


Table 7. Start and Stop module callers

Function	Callers
Start Module	Test Configuration API.vi
Stop Module	Configuration.lvlib:Handle Exit.vi Test Configuration API.vi

Module custom errors

TIP	Custom errors are added to the module via vi named `*--error.vi`.

Module Configuration.lvlib use the following custom errors:

Table 8. Custom errors

Name	Code	Description
Module Not Running	0	
Module Not Stopped	0	
Module Not Synced	0	
Request and Wait for Reply Timeout	0	

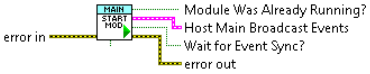
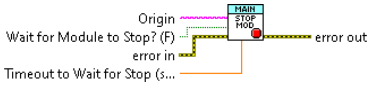
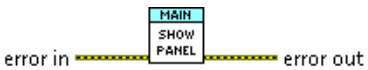
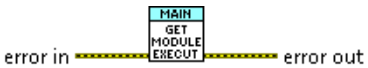
2.1.4. Host Main.lvlib

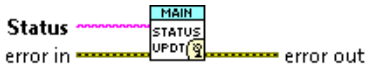
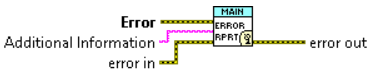
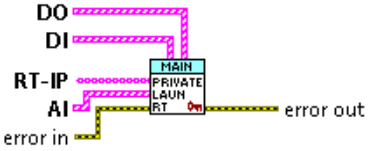
Type: Singleton

Responsibility: This is the Main GUI Module which is going to launch and control the realtime target's.

Event list

Table 9. Events

Name	Type	Connector pane	Description	S.	R.	I.
Start Module			Launches the module Main VI. After calling this VI, you can optionally register for broadcast events from the module by wiring the broadcast events output of this VI to a Register For Events function. After the optional Register For Events function call, you should always call the Synchronize Module Events.vi for this module with the 'Wait for Event Sync?' output of this VI to the corresponding input of the Synchronize Module Events.vi. To see an example of the proper wiring pattern, see the "Start Module: Value Change" event frame in the API Tester VI for this module.			
Stop Module			Send the Stop request to the Module's Main.vi. If Wait for Module to Stop? is TRUE, this VI will wait until the module main VI stops, and will timeout at the Timeout to Wait for Stop value. This value defaults to "-1", which means the VI will not timeout, and will always wait until the module main VI stops before completing execution. Note: The Timeout to Wait for Stop value is ignored if 'Wait for Module to Stop?' is set to FALSE.			
Show Panel			Send the Show Panel request to the Module's Main.vi.			
Hide Panel			Send the Hide Panel request to the Module's Main.vi.			
Get Module Execution Status			Fire the Get Module Execution Status request.			

Name	Type	Connector pane	Description	S.	R.	I.
Show Diagram			This VI tells the Module to show its block diagram to facilitate troubleshooting (add probes, breakpoints, highlight execution, etc).			
Module Did Init			Send the Module Did Init event to any VI registered to listen to this module's broadcast events.			
Status Updated			Send the Status Updated event to any VI registered to listen to events from the owning module.			
Error Reported			Send the Error Reported event to any VI registered to listen to events from the owning module.			
Module Did Stop			Send the Module Did Stop event to any VI registered to listen to this module's broadcast events.			
Update Module Execution Status			Broadcast event to specify whether or not the module is running.			
RT-Module Id's		[Host Main.lvlib:RT-Module Id's.vi]	This broadcast Event will give the Respective module if for the Respective Realtime Target.			
Launch RT Acq			This request event will launch the RT Acquisition module's based on the Given Input's.			

Type: → Request | → Request and Wait for Reply | → Broadcast

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

Module relationship

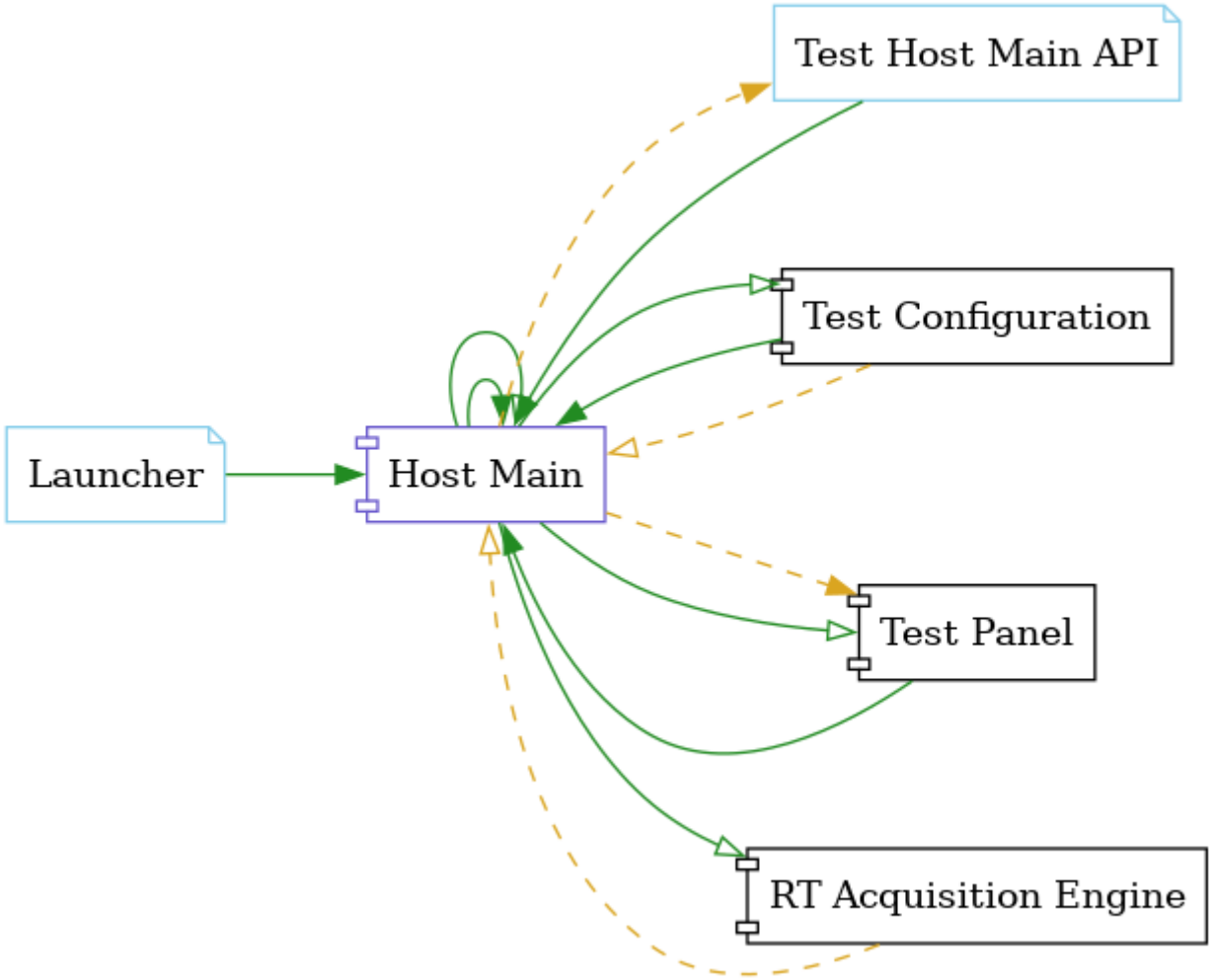


Table 10. Requests callers

Request Name	Callers
Get Module Execution Status	Host Main.lvlib:Obtain Broadcast Events for Registration.vi Host Main.lvlib:Start Module.vi
Hide Panel	Host Main.lvlib:Main.vi Test Host Main API.vi
Launch RT Acq	Host Main.lvlib:Main.vi
Show Diagram	Test Host Main API.vi
Show Panel	Launcher.vi Test Configuration.lvlib:Main.vi Test Host Main API.vi Test Panel.lvlib:Main.vi

Table 11. Broadcasts Listeners

Broadcast Name	Listeners
Error Reported	Test Host Main API.vi
Module Did Init	Test Host Main API.vi
Module Did Stop	Test Host Main API.vi

Broadcast Name	Listeners
RT-Module Id's	Test Host Main API.vi Test Panel.lvlib:Main.vi
Status Updated	Test Host Main API.vi
Update Module Execution Status	Test Host Main API.vi

Table 12. Used requests

Module	Requests
Host Main.lvlib	Hide Panel.vi Launch RT Acq.vi Stop Module.vi (2)
RT Acquisition Engine.lvlib	Stop Module.vi
Test Configuration.lvlib	Get Module Execution Status.vi (2) Show Panel.vi Stop Module.vi
Test Panel.lvlib	Get Module Execution Status.vi (2) Stop Module.vi

Table 13. Registered broadcast

Module	Broadcasts
RT Acquisition Engine.lvlib	RT Connection Status.vi
Test Configuration.lvlib	Error Reported.vi Module Did Init.vi Status Updated.vi

Module Start/Stop calls

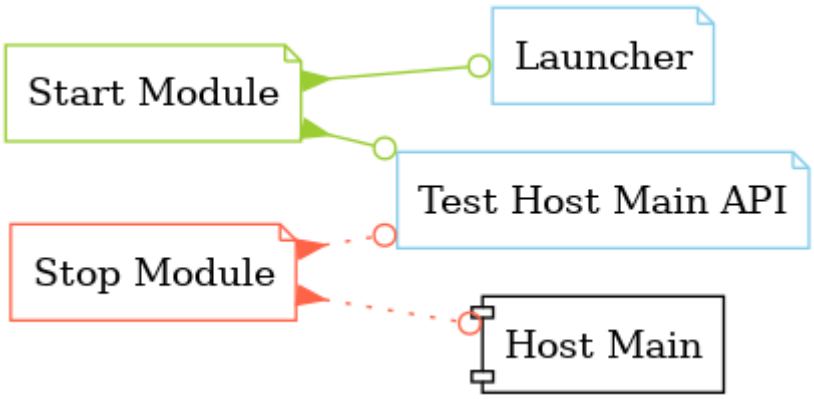


Table 14. Start and Stop module callers

Function	Callers
Start Module	Test Host Main API.vi Launcher.vi

Function	Callers
Stop Module	Host Main.lvlib:Main.vi Host Main.lvlib:Handle Exit.vi Test Host Main API.vi

Module custom errors

TIP	Custom errors are added to the module via vi named `*--error.vi`.

Module Host Main.lvlib use the following custom errors:

Table 15. Custom errors

Name	Code	Description
Module Not Running	0	
Module Not Stopped	0	
Module Not Synced	0	
Request and Wait for Reply Timeout	0	

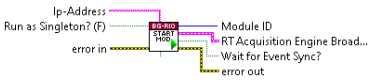
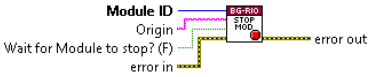
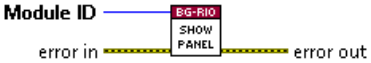
2.1.5. RT Acquisition Engine.lvlib

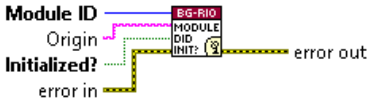
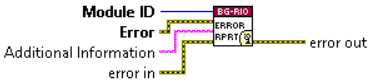
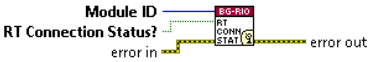
Type: Cloneable

Responsibility: This Clone Module will run for Each Target by taking it's respective IP address and open the TCP-IP Commnication between Host and the Respective Target.

Event list

Table 16. Events

Name	Type	Connector pane	Description	S.	R.	I.
Start Module			Launches an instance of the module Main VI. After calling this VI, you can optionally register for broadcast events from the module by wiring the broadcast events output of this VI to a Register For Events function. After the optional Register For Events function call, you should always call the Synchronize Module Events.vi for this module with the 'Wait for Event Sync?' output of this VI to the corresponding input of the Synchronize Module Events.vi. To see an example of the proper wiring pattern, see the "Run New Module: Value Change" event frame in the API Tester VI for this module.			
Stop Module			Send the Stop request to the Module's Main.vi. If Wait for Module to stop? is TRUE, then this VI will not complete execution until the Module Main VI has stopped running. Note: If the cloneable module is running as singleton, then the 'Wait for Module to stop?' input is ignored... this VI will always wait until a cloneable Main VI running as singleton has stopped running.			
Show Panel			Send the Show Panel request to the Module's Main.vi.			
Hide Panel			Send the Hide Panel request to the Module's Main.vi.			
Show Diagram			This VI tells the Module to show its block diagram to facilitate troubleshooting (add probes, breakpoints, highlight execution, etc).			
Write Data To the RT			This Request event will send the Data from the Host to Realtime Using TCP-IP Protocol.			

Name	Type	Connector pane	Description	S.	R.	I.
Module Did Init			Send the Module Did Init event to any VI registered to listen to this module's broadcast events.			
Status Updated			Send the Status Updated event to any VI registered to listen to events from the owning module.			
Error Reported			Send the Error Reported event to any VI registered to listen to events from the owning module.			
Module Did Stop			Send the Module Did Stop event to any VI registered to listen to this module's broadcast events.			
Update Module Execution Status			Fire the Get Module Execution Status request.			
RT Connection Status			This Broadcast Event will send wheter the RT is connected with the Host or Not.			

Type: → Request | → Request and Wait for Reply | → Broadcast

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

Module relationship

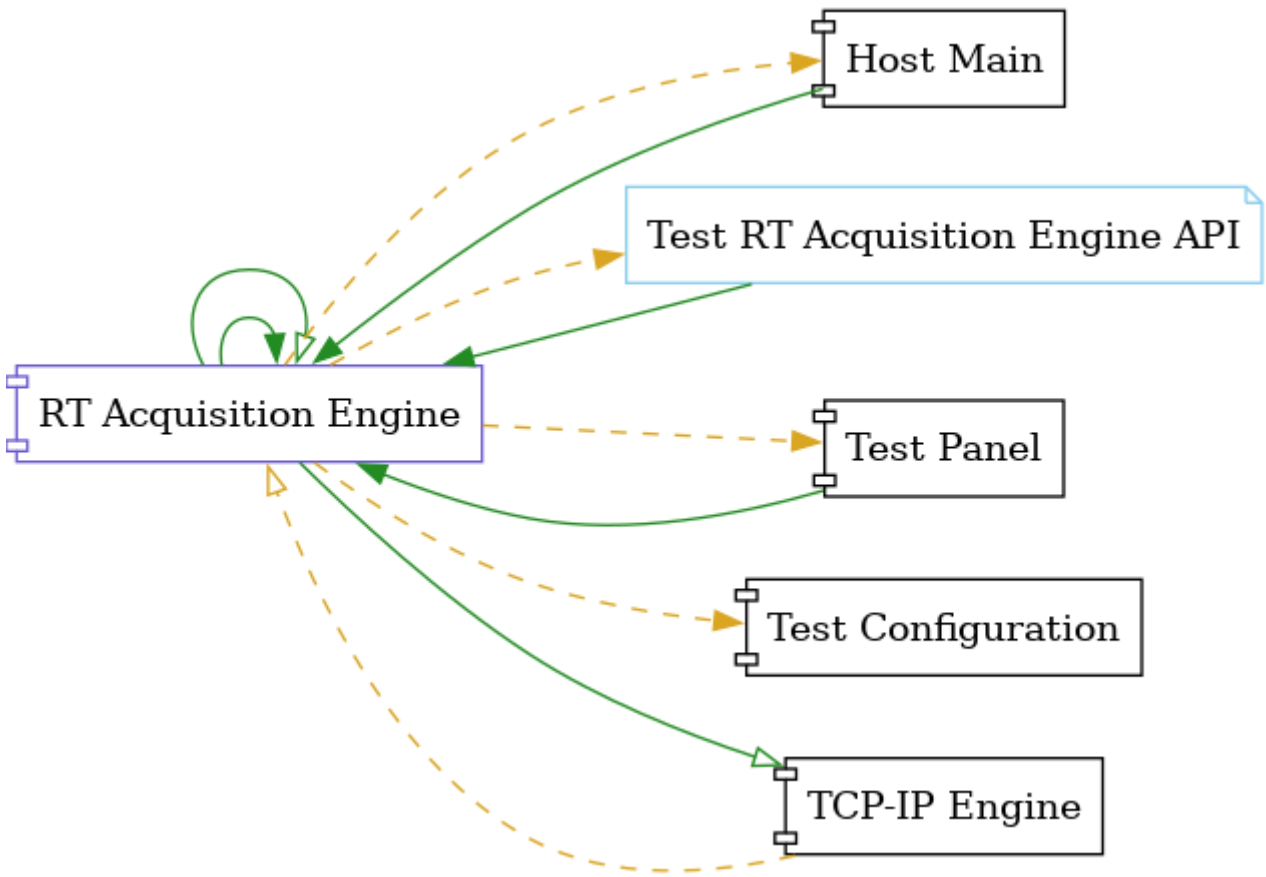


Table 17. Requests callers

Request Name	Callers
Hide Panel	Test RT Acquisition Engine API.vi
Show Diagram	Test RT Acquisition Engine API.vi
Show Panel	Test RT Acquisition Engine API.vi
Write Data To the RT	Test Panel.lvlib:Main.vi Test RT Acquisition Engine API.vi

Table 18. Broadcasts Listeners

Broadcast Name	Listeners
Error Reported	Test RT Acquisition Engine API.vi
Module Did Init	Test RT Acquisition Engine API.vi
Module Did Stop	Test RT Acquisition Engine API.vi
RT Connection Status	Host Main.lvlib:Main.vi Test RT Acquisition Engine API.vi Test Configuration.lvlib:Main.vi Test Panel.lvlib:Main.vi
Status Updated	Test RT Acquisition Engine API.vi
Update Module Execution Status	Test RT Acquisition Engine API.vi

Table 19. Used requests

Module	Requests
RT Acquisition Engine.lvlib	Stop Module.vi
TCP-IP Engine.lvlib	Read TCP Data.vi (2) Stop Module.vi TCP-Close Connection.vi TCP-Open Connection.vi (2) TCP-Write.vi (3)

Table 20. Registered broadcast

Module	Broadcasts
TCP-IP Engine.lvlib	Error Reported.vi Module Did Init.vi Module Did Stop.vi Open Connection Status.vi Status Updated.vi TCP-Read Data Out.vi Update Module Execution Status.vi

Module Start/Stop calls

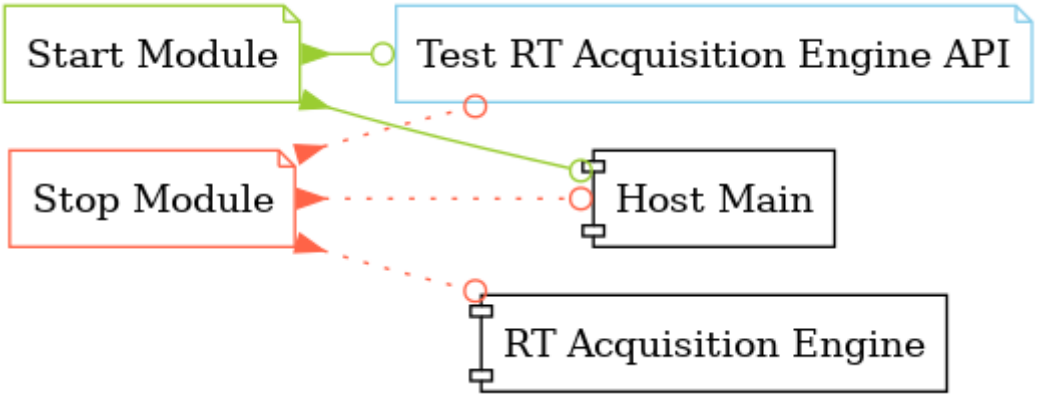


Table 21. Start and Stop module callers

Function	Callers
Start Module	Host Main.lvlib:Main.vi Test RT Acquisition Engine API.vi
Stop Module	Host Main.lvlib:Main.vi RT Acquisition Engine.lvlib:Handle Exit.vi Test RT Acquisition Engine API.vi

Module custom errors

TIP Custom errors are added to the module via vi named `*--error.vi`.

Module RT Acquisition Engine.lvlib use the following custom errors:

Table 22. Custom errors

Name	Code	Description
Master Reference Not Closed	0	
Module Not Running	0	
Module Not Stopped	0	
Module Not Synced	0	
Module Running as Cloneable	0	
Module Running as Singleton	0	
Request and Wait for Reply Timeout	0	

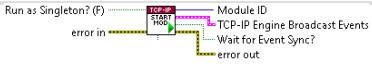
2.1.6. TCP-IP Engine.lvlib

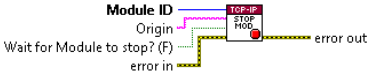
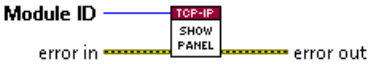
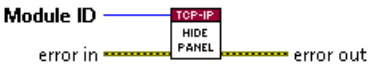
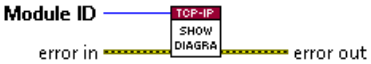
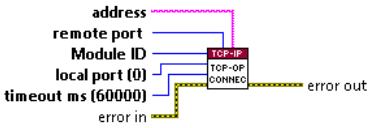
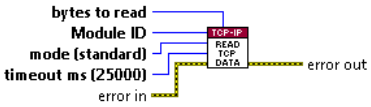
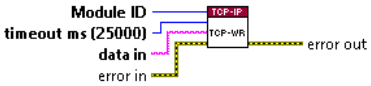
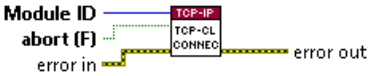
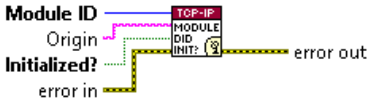
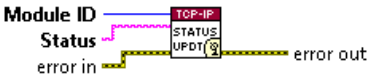
Type: Cloneable

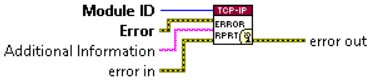
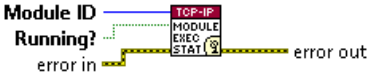
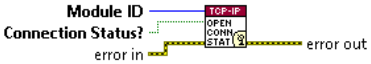
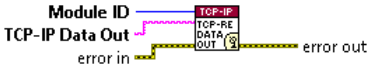
Responsibility: This clone module will transmit and receive comand's using TCP-IP Protocol.

Event list

Table 23. Events

Name	Type	Connector pane	Description	S.	R.	I.
Start Module			Launches an instance of the module Main VI. After calling this VI, you can optionally register for broadcast events from the module by wiring the broadcast events output of this VI to a Register For Events function. After the optional Register For Events function call, you should always call the Synchronize Module Events.vi for this module with the 'Wait for Event Sync?' output of this VI to the corresponding input of the Synchronize Module Events.vi. To see an example of the proper wiring pattern, see the "Run New Module: Value Change" event frame in the API Tester VI for this module.			

Name	Type	Connector pane	Description	S.	R.	I.
Stop Module			Send the Stop request to the Module's Main.vi. If Wait for Module to stop? is TRUE, then this VI will not complete execution until the Module Main VI has stopped running. Note: If the cloneable module is running as singleton, then the 'Wait for Module to stop?' input is ignored... this VI will always wait until a cloneable Main VI running as singleton has stopped running.			
Show Panel			Send the Show Panel request to the Module's Main.vi.			
Hide Panel			Send the Hide Panel request to the Module's Main.vi.			
Show Diagram			This VI tells the Module to show its block diagram to facilitate troubleshooting (add probes, breakpoints, highlight execution, etc).			
TCP-Open Connection			This Request event will request the TCP-Engine to open the Connection based on the input's wh have given.			
Read TCP Data			This Request Event will request the TCP-IP Module to read the data from the respective connection.			
TCP-Write			This Request event will request the TCP-IP Module to write the Data which is connected to the Data In terminal of the Input.			
TCP-Close Connection			This Request Event will request the TCP-IP module to close the respective opened connection.			
Module Did Init			Send the Module Did Init event to any VI registered to listen to this module's broadcast events.			
Status Updated			Send the Status Updated event to any VI registered to listen to events from the owning module.			

Name	Type	Connector pane	Description	S.	R.	I.
Error Reported			Send the Error Reported event to any VI registered to listen to events from the owning module.			
Module Did Stop			Send the Module Did Stop event to any VI registered to listen to this module's broadcast events.			
Update Module Execution Status			Fire the Get Module Execution Status request.			
Open Connection Status			This Broadcast Event will sent the Open Connection Status i.e whether TCP connection is established successfully between RT and Host.			
TCP-Read Data Out			This Broadcast Event will send the Data which has been read from the TCP-IP Conection.			

Type: → Request | → Request and Wait for Reply | → Broadcast

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

Module relationship

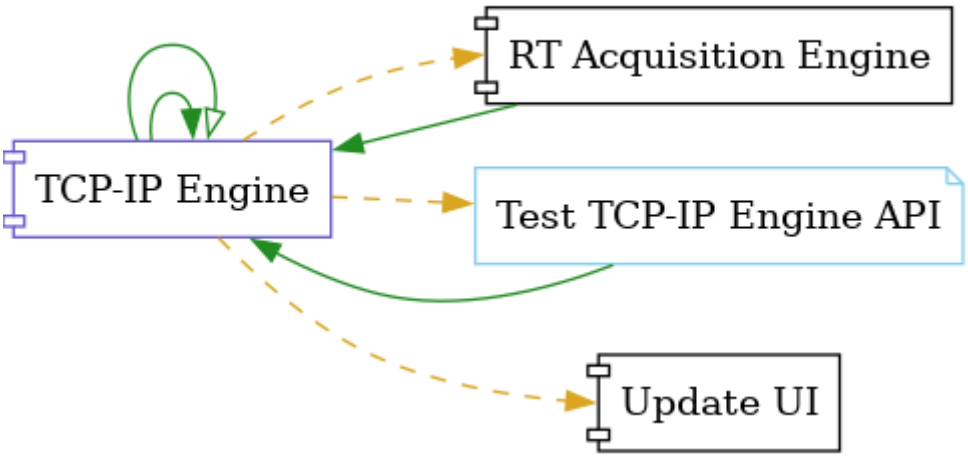


Table 24. Requests callers

Request Name	Callers
Hide Panel	Test TCP-IP Engine API.vi
Read TCP Data	RT Acquisition Engine.lvlib:Main.vi Test TCP-IP Engine API.vi
Show Diagram	Test TCP-IP Engine API.vi
Show Panel	Test TCP-IP Engine API.vi
TCP-Close Connection	Test TCP-IP Engine API.vi
TCP-Open Connection	RT Acquisition Engine.lvlib:Main.vi Test TCP-IP Engine API.vi
TCP-Write	RT Acquisition Engine.lvlib:Main.vi Test TCP-IP Engine API.vi

Table 25. Broadcasts Listeners

Broadcast Name	Listeners
Error Reported	RT Acquisition Engine.lvlib:Main.vi Test TCP-IP Engine API.vi
Module Did Init	RT Acquisition Engine.lvlib:Main.vi Test TCP-IP Engine API.vi
Module Did Stop	RT Acquisition Engine.lvlib:Main.vi Test TCP-IP Engine API.vi
Open Connection Status	RT Acquisition Engine.lvlib:Main.vi Test TCP-IP Engine API.vi
Status Updated	RT Acquisition Engine.lvlib:Main.vi Test TCP-IP Engine API.vi
TCP-Read Data Out	RT Acquisition Engine.lvlib:Main.vi Test TCP-IP Engine API.vi Update UI.lvlib:Main.vi
Update Module Execution Status	RT Acquisition Engine.lvlib:Main.vi Test TCP-IP Engine API.vi

Table 26. Used requests

Module	Requests
TCP-IP Engine.lvlib	Read TCP Data.vi Stop Module.vi TCP-Close Connection.vi TCP-Open Connection.vi TCP-Write.vi

Table 27. Registered broadcast

Module	Broadcasts
—	—

Module Start/Stop calls

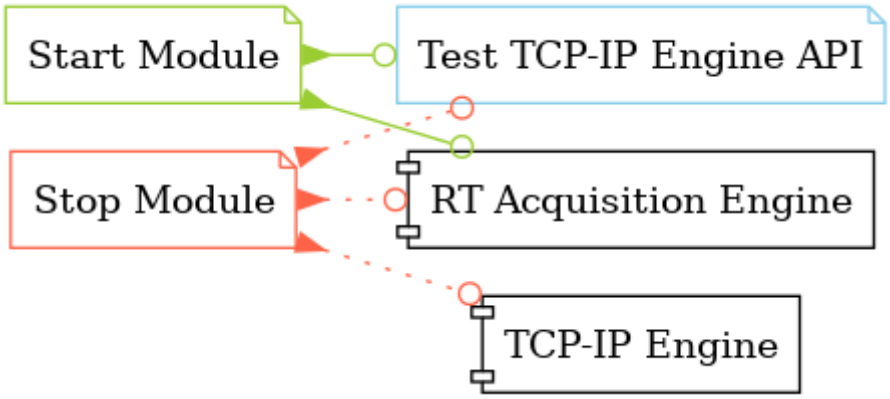


Table 28. Start and Stop module callers

Function	Callers
Start Module	RT Acquisition Engine.lvlib:Main.vi Test TCP-IP Engine API.vi
Stop Module	RT Acquisition Engine.lvlib:Main.vi TCP-IP Engine.lvlib:Handle Exit.vi Test TCP-IP Engine API.vi

Module custom errors

TIP Custom errors are added to the module via vi named `*--error.vi`.

Module TCP-IP Engine.lvlib use the following custom errors:

Table 29. Custom errors

Name	Code	Description
Master Reference Not Closed	0	
Module Not Running	0	
Module Not Stopped	0	
Module Not Synced	0	
Module Running as Cloneable	0	
Module Running as Singleton	0	
Request and Wait for Reply Timeout	0	

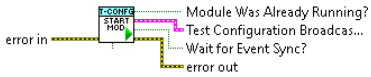
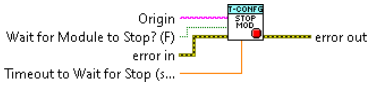
2.1.7. Test Configuration.lvlib

Type: Singleton

Responsibility: This module will give the configurating file's path which are required to do the test.

Event list

Table 30. Events

Name	Type	Connector pane	Description	S.	R.	I.
Start Module			Launches the module Main VI. After calling this VI, you can optionally register for broadcast events from the module by wiring the broadcast events output of this VI to a Register For Events function. After the optional Register For Events function call, you should always call the Synchronize Module Events.vi for this module with the 'Wait for Event Sync?' output of this VI to the corresponding input of the Synchronize Module Events.vi. To see an example of the proper wiring pattern, see the "Start Module: Value Change" event frame in the API Tester VI for this module.			
Stop Module			Send the Stop request to the Module's Main.vi. If Wait for Module to Stop? is TRUE, this VI will wait until the module main VI stops, and will timeout at the Timeout to Wait for Stop value. This value defaults to "-1", which means the VI will not timeout, and will always wait until the module main VI stops before completing execution. Note: The Timeout to Wait for Stop value is ignored if 'Wait for Module to Stop?' is set to FALSE.			

Name	Type	Connector pane	Description	S.	R.	I.
Do Something		Do Something Message, error in, error out	Send the Do something request to the Module's Main.vi.			
Do Something Else		Do Something Else Message, X, error in, error out	Send the Do Something Else request to the Module's Main.vi.			
Do Something Else and Wait for Reply		Do Something Else Message, X, error in, Reply Payload, timed out?, error out	Send the Do Something Else request to the Module's Main.vi.			
Show Panel		error in, error out	Send the Show Panel request to the Module's Main.vi.			
Hide Panel		error in, error out	Send the Hide Panel request to the Module's Main.vi.			
Get Module Execution Status		error in, error out	Fire the Get Module Execution Status request.			
Show Diagram		error in, error out	This VI tells the Module to show its block diagram to facilitate troubleshooting (add probes, breakpoints, highlight execution, etc).			
Module Did Init		Origin, Initialized?, error in, error out	Send the Module Did Init event to any VI registered to listen to this module's broadcast events.			
Did Something		Did Something Msg, error in, error out	Send the Did Something event to any VI registered to listen to this module's broadcast events.			
Status Updated		Status, error in, error out	Send the Status Updated event to any VI registered to listen to events from the owning module.			
Error Reported		Error, Additional Information, error in, error out	Send the Error Reported event to any VI registered to listen to events from the owning module.			
Module Did Stop		Origin, error in, error out	Send the Module Did Stop event to any VI registered to listen to this module's broadcast events.			

Name	Type	Connector pane	Description	S.	R.	I.
Update Module Execution Status		Running? error in T-CONFG MODULE EXEC STAT error out	Broadcast event to specify whether or not the module is running.			

Type: → Request | → Request and Wait for Reply | → Broadcast

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

Module relationship

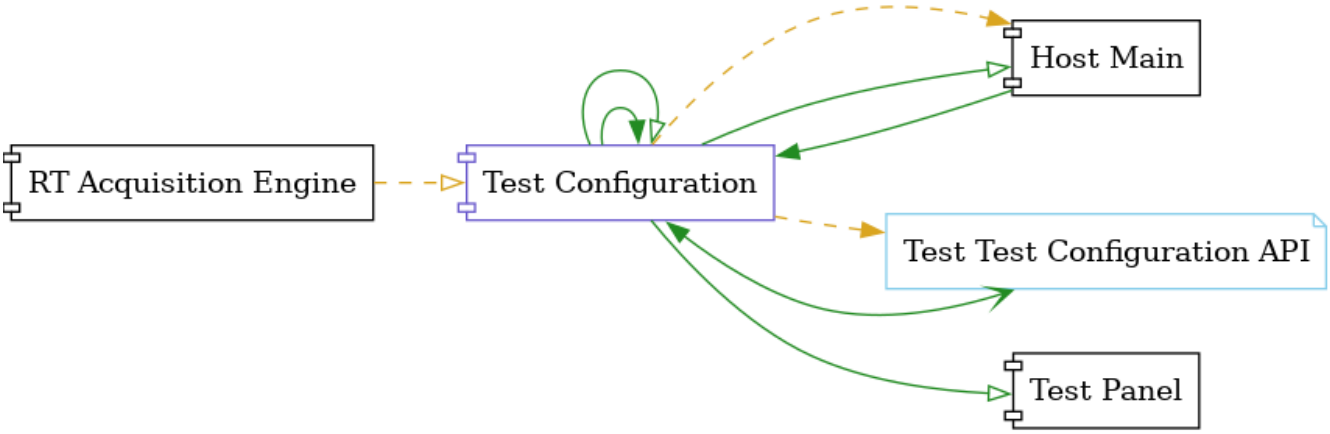


Table 31. Requests callers

Request Name	Callers
Do Something Else and Wait for Reply	Test Test Configuration API.vi
Do Something Else	Test Test Configuration API.vi
Do Something	Test Test Configuration API.vi
Get Module Execution Status	Test Configuration.lvlib:Obtain Broadcast Events for Registration.vi Test Configuration.lvlib:Start Module.vi
Hide Panel	Test Configuration.lvlib:Main.vi Test Test Configuration API.vi
Show Diagram	Test Test Configuration API.vi
Show Panel	Host Main.lvlib:Main.vi Test Test Configuration API.vi

Table 32. Broadcasts Listeners

Broadcast Name	Listeners
Did Something	Test Test Configuration API.vi
Error Reported	Host Main.lvlib:Main.vi Test Test Configuration API.vi
Module Did Init	Host Main.lvlib:Main.vi Test Test Configuration API.vi
Module Did Stop	Test Test Configuration API.vi
Status Updated	Host Main.lvlib:Main.vi Test Test Configuration API.vi
Update Module Execution Status	Test Test Configuration API.vi

Table 33. Used requests

Module	Requests
Host Main.lvlib	Show Panel.vi
Test Configuration.lvlib	Hide Panel.vi (2) Stop Module.vi
Test Panel.lvlib	Show Panel.vi

Table 34. Registered broadcast

Module	Broadcasts
RT Acquisition Engine.lvlib	RT Connection Status.vi

Module Start/Stop calls

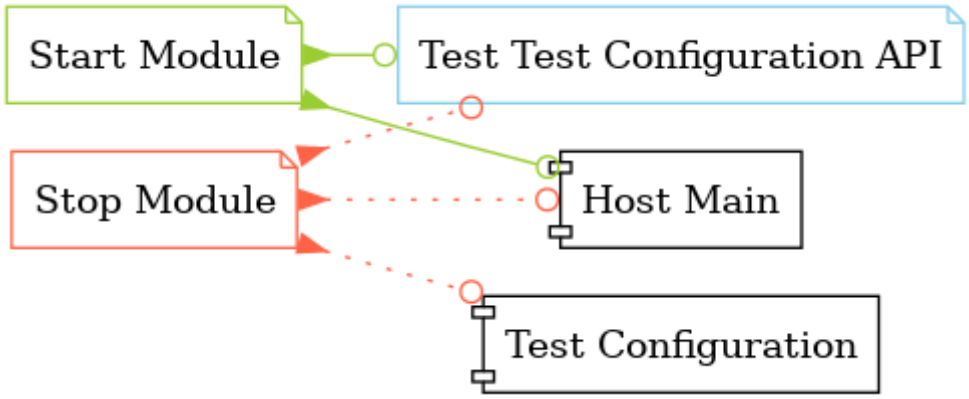


Table 35. Start and Stop module callers

Function	Callers
Start Module	Host Main.lvlib:Main.vi Test Test Configuration API.vi
Stop Module	Host Main.lvlib:Main.vi Test Configuration.lvlib:Handle Exit.vi Test Test Configuration API.vi

Module custom errors

TIP	Custom errors are added to the module via vi named `*--error.vi`.

Module Test Configuration.lvlib use the following custom errors:

Table 36. Custom errors

Name	Code	Description
Module Not Running	0	
Module Not Stopped	0	
Module Not Synced	0	
Request and Wait for Reply Timeout	0	

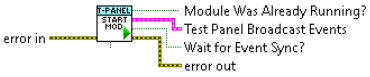
2.1.8. Test Panel.lvlib

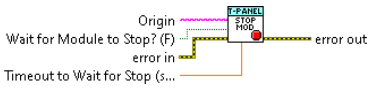
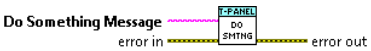
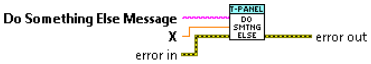
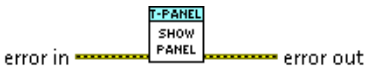
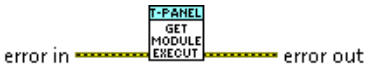
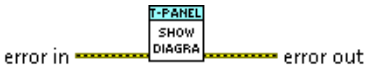
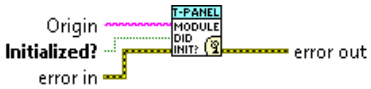
Type: Singleton

Responsibility: Using this test panel the controlling and monitoring of all the componente's for the Drop Test operation has been performed.

Event list

Table 37. Events

Name	Type	Connector pane	Description	S.	R.	I.
Start Module			Launches the module Main VI. After calling this VI, you can optionally register for broadcast events from the module by wiring the broadcast events output of this VI to a Register For Events function. After the optional Register For Events function call, you should always call the Synchronize Module Events.vi for this module with the 'Wait for Event Sync?' output of this VI to the corresponding input of the Synchronize Module Events.vi. To see an example of the proper wiring pattern, see the "Start Module: Value Change" event frame in the API Tester VI for this module.			

Name	Type	Connector pane	Description	S.	R.	I.
Stop Module			Send the Stop request to the Module's Main.vi. If Wait for Module to Stop? is TRUE, this VI will wait until the module main VI stops, and will timeout at the Timeout to Wait for Stop value. This value defaults to "-1", which means the VI will not timeout, and will always wait until the module main VI stops before completing execution. Note: The Timeout to Wait for Stop value is ignored if 'Wait for Module to Stop?' is set to FALSE.			
Do Something			Send the Do something request to the Module's Main.vi.			
Do Something Else			Send the Do Something Else request to the Module's Main.vi.			
Do Something Else and Wait for Reply			Send the Do Something Else request to the Module's Main.vi.			
Show Panel			Send the Show Panel request to the Module's Main.vi.			
Hide Panel			Send the Hide Panel request to the Module's Main.vi.			
Get Module Execution Status			Fire the Get Module Execution Status request.			
Show Diagram			This VI tells the Module to show its block diagram to facilitate troubleshooting (add probes, breakpoints, highlight execution, etc).			
Module Did Init			Send the Module Did Init event to any VI registered to listen to this module's broadcast events.			

Name	Type	Connector pane	Description	S.	R.	I.
Did Something		Did Something Msg, error in, error out	Send the Did Something event to any VI registered to listen to this module's broadcast events.			
Status Updated		Status, error in, error out	Send the Status Updated event to any VI registered to listen to events from the owning module.			
Error Reported		Error, Additional Information, error in, error out	Send the Error Reported event to any VI registered to listen to events from the owning module.			
Module Did Stop		Origin, error in, error out	Send the Module Did Stop event to any VI registered to listen to this module's broadcast events.			
Update Module Execution Status		Running?, error in, error out	Broadcast event to specify whether or not the module is running.			

Type: → Request | → Request and Wait for Reply | → Broadcast

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

Module relationship

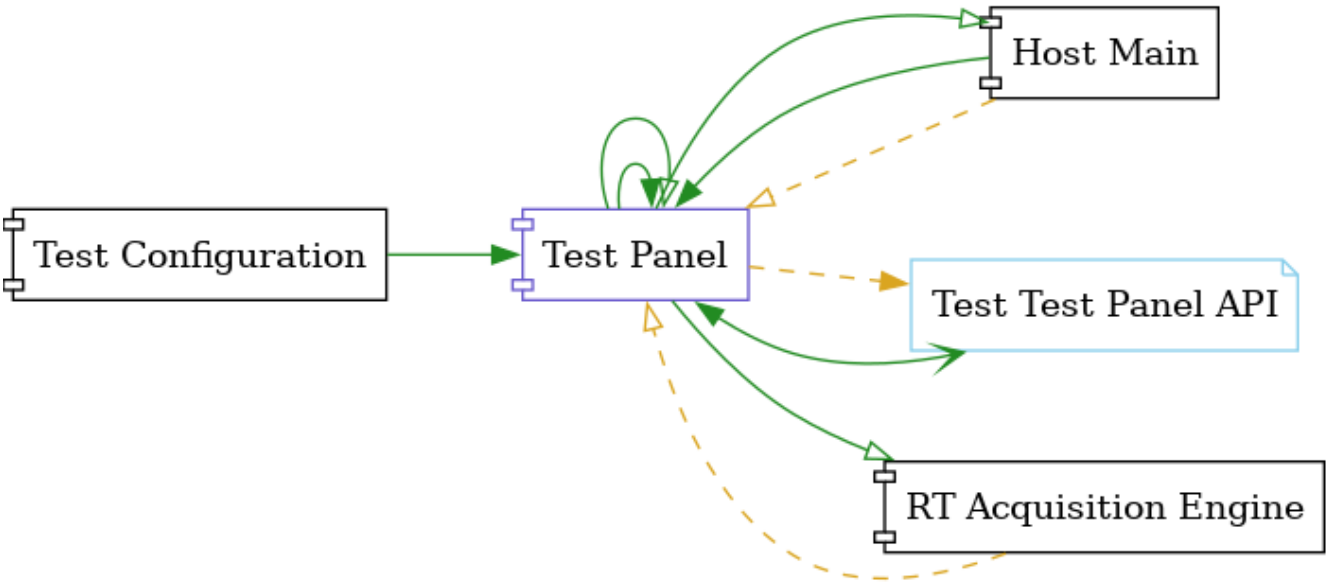


Table 38. Requests callers

Request Name	Callers
Do Something Else and Wait for Reply	Test Test Panel API.vi
Do Something Else	Test Test Panel API.vi
Do Something	Test Test Panel API.vi
Get Module Execution Status	Test Panel.lvlib:Obtain Broadcast Events for Registration.vi Test Panel.lvlib:Start Module.vi
Hide Panel	Test Panel.lvlib:Main.vi Test Test Panel API.vi
Show Diagram	Test Test Panel API.vi
Show Panel	Test Configuration.lvlib:Main.vi Test Test Panel API.vi

Table 39. Broadcasts Listeners

Broadcast Name	Listeners
Did Something	Test Test Panel API.vi
Error Reported	Test Test Panel API.vi
Module Did Init	Test Test Panel API.vi
Module Did Stop	Test Test Panel API.vi
Status Updated	Test Test Panel API.vi
Update Module Execution Status	Test Test Panel API.vi

Table 40. Used requests

Module	Requests
Host Main.lvlib	Get Module Execution Status.vi Show Panel.vi
RT Acquisition Engine.lvlib	Write Data To the RT.vi
Test Panel.lvlib	Hide Panel.vi Stop Module.vi

Table 41. Registered broadcast

Module	Broadcasts
Host Main.lvlib	RT-Module Id’s.vi
RT Acquisition Engine.lvlib	RT Connection Status.vi

Module Start/Stop calls

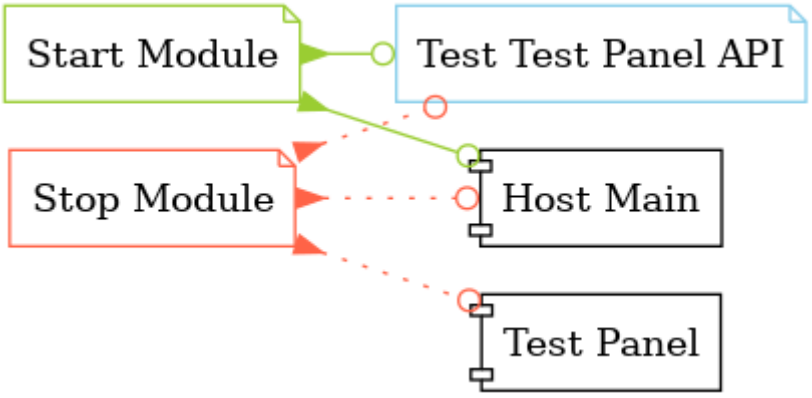


Table 42. Start and Stop module callers

Function	Callers
Start Module	Host Main.lvlib:Main.vi Test Test Panel API.vi
Stop Module	Host Main.lvlib:Main.vi Test Panel.lvlib:Handle Exit.vi Test Test Panel API.vi

Module custom errors

TIP Custom errors are added to the module via vi named `*--error.vi`.

Module Test Panel.lvlib use the following custom errors:

Table 43. Custom errors

Name	Code	Description
Module Not Running	0	
Module Not Stopped	0	
Module Not Synced	0	
Request and Wait for Reply Timeout	0	

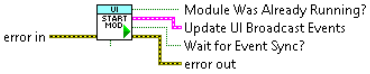
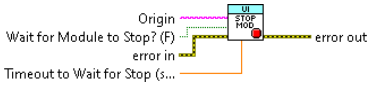
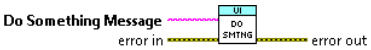
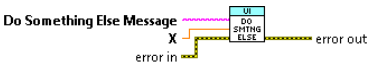
2.1.9. Update UI.lvlib

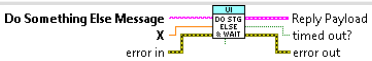
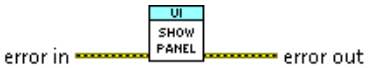
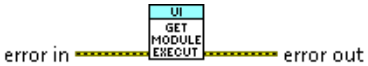
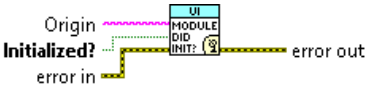
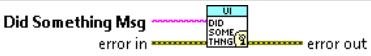
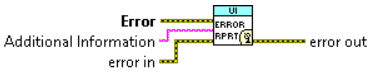
Type: Singleton

Responsibility: This Module will take all the Control and Indicator reference's from each Module and update their value's.

Event list

Table 44. Events

Name	Type	Connector pane	Description	S.	R.	I.
Start Module			Launches the module Main VI. After calling this VI, you can optionally register for broadcast events from the module by wiring the broadcast events output of this VI to a Register For Events function. After the optional Register For Events function call, you should always call the Synchronize Module Events.vi for this module with the 'Wait for Event Sync?' output of this VI to the corresponding input of the Synchronize Module Events.vi. To see an example of the proper wiring pattern, see the "Start Module: Value Change" event frame in the API Tester VI for this module.			
Stop Module			Send the Stop request to the Module's Main.vi. If Wait for Module to Stop? is TRUE, this VI will wait until the module main VI stops, and will timeout at the Timeout to Wait for Stop value. This value defaults to "-1", which means the VI will not timeout, and will always wait until the module main VI stops before completing execution. Note: The Timeout to Wait for Stop value is ignored if 'Wait for Module to Stop?' is set to FALSE.			
Do Something			Send the Do something request to the Module's Main.vi.			
Do Something Else			Send the Do Something Else request to the Module's Main.vi.			

Name	Type	Connector pane	Description	S.	R.	I.
Do Something Else and Wait for Reply			Send the Do Something Else request to the Module's Main.vi.			
Show Panel			Send the Show Panel request to the Module's Main.vi.			
Hide Panel			Send the Hide Panel request to the Module's Main.vi.			
Get Module Execution Status			Fire the Get Module Execution Status request.			
Show Diagram			This VI tells the Module to show its block diagram to facilitate troubleshooting (add probes, breakpoints, highlight execution, etc).			
Module Did Init			Send the Module Did Init event to any VI registered to listen to this module's broadcast events.			
Did Something			Send the Did Something event to any VI registered to listen to this module's broadcast events.			
Status Updated			Send the Status Updated event to any VI registered to listen to events from the owning module.			
Error Reported			Send the Error Reported event to any VI registered to listen to events from the owning module.			
Module Did Stop			Send the Module Did Stop event to any VI registered to listen to this module's broadcast events.			
Update Module Execution Status			Broadcast event to specify whether or not the module is running.			

Type: → Request | → Request and Wait for Reply | → Broadcast

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

Module relationship

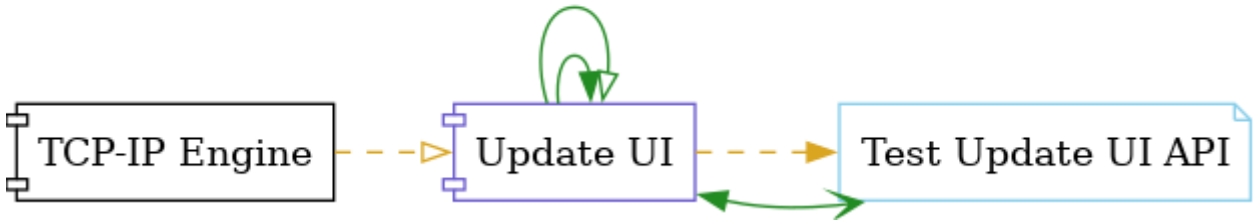


Table 45. Requests callers

Request Name	Callers
Do Something Else and Wait for Reply	Test Update UI API.vi
Do Something Else	Test Update UI API.vi
Do Something	Test Update UI API.vi
Get Module Execution Status	Update UI.lvlib:Obtain Broadcast Events for Registration.vi Update UI.lvlib:Start Module.vi
Hide Panel	Test Update UI API.vi
Show Diagram	Test Update UI API.vi
Show Panel	Test Update UI API.vi

Table 46. Broadcasts Listeners

Broadcast Name	Listeners
Did Something	Test Update UI API.vi
Error Reported	Test Update UI API.vi
Module Did Init	Test Update UI API.vi
Module Did Stop	Test Update UI API.vi
Status Updated	Test Update UI API.vi
Update Module Execution Status	Test Update UI API.vi

Table 47. Used requests

Module	Requests
Update UI.lvlib	Stop Module.vi

Table 48. Registered broadcast

Module	Broadcasts
TCP-IP Engine.lvlib	TCP-Read Data Out.vi

Module Start/Stop calls

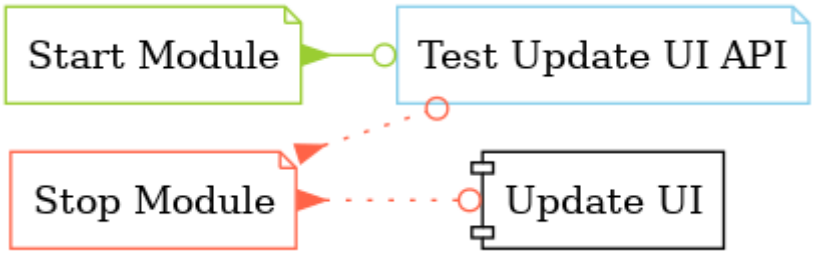


Table 49. Start and Stop module callers

Function	Callers
Start Module	Test Update UI API.vi
Stop Module	Test Update UI API.vi Update UI.lvlib:Handle Exit.vi

Module custom errors

> **TIP** Custom errors are added to the module via vi named `*--error.vi`.

Module Update UI.lvlib use the following custom errors:

Table 50. Custom errors

Name	Code	Description
Module Not Running	0	
Module Not Stopped	0	
Module Not Synced	0	
Request and Wait for Reply Timeout	0	

2.2. Libraries

This section describes the libraries contained in the project.

2.2.1. camera commands.lvlib

Responsibility: No description found (add content in lvlib description)

Version: 1.0.0.0

This library has no functions set to non private scope.

2.3. Custom errors

> **TIP** Custom errors are added via vi named `*--error.vi`.

Table 51. Custom errors

Name	Code	Description	Owned by
Master Reference Not Closed	0		RT Acquisition Engine.lvlib TCP-IP Engine.lvlib
Module Not Running	0		Configuration.lvlib Host Main.lvlib RT Acquisition Engine.lvlib TCP-IP Engine.lvlib Test Configuration.lvlib Test Panel.lvlib Update UI.lvlib
Module Not Stopped	0		Configuration.lvlib Host Main.lvlib RT Acquisition Engine.lvlib TCP-IP Engine.lvlib Test Configuration.lvlib Test Panel.lvlib Update UI.lvlib
Module Not Synced	0		Configuration.lvlib Host Main.lvlib RT Acquisition Engine.lvlib TCP-IP Engine.lvlib Test Configuration.lvlib Test Panel.lvlib Update UI.lvlib
Module Running as Cloneable	0		RT Acquisition Engine.lvlib TCP-IP Engine.lvlib
Module Running as Singleton	0		RT Acquisition Engine.lvlib TCP-IP Engine.lvlib
Request and Wait for Reply Timeout	0		Configuration.lvlib Host Main.lvlib RT Acquisition Engine.lvlib TCP-IP Engine.lvlib Test Configuration.lvlib Test Panel.lvlib Update UI.lvlib

Chapter 3. Capsule-3&4 (192.168.32.50)

3.1. Libraries

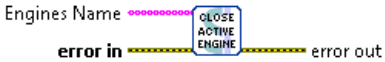
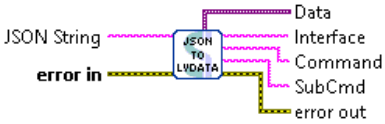
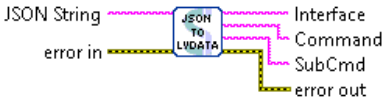
This section describes the libraries contained in the project.

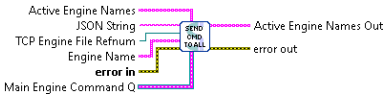
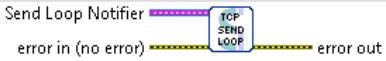
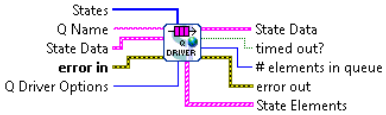
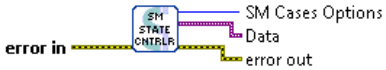
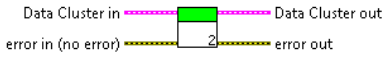
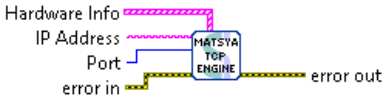
3.1.1. TCP Engine.lvlib

Responsibility: No description found (add content in lvlib description)

Version: 1.0.0.0

Table 52. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
TCP Engine AIO&DIO JSON Conversion	[TCP Engine.lvlib:TCP Engine AIO&DIO JSON Conversion.vi]	All Rights Reserved © 2017 SyncSols			
TCP ENGINE Async Data JSON Conversion		All Rights Reserved © 2017 SyncSols			
TCP Engine Close Active Engines		All Rights Reserved © 2017 SyncSols			
TCP Engine Device Name		This purpose of this VI is to get all engine names from JSON Commands. All Rights Reserved © 2017 SyncSols			
TCP Engine Hardware Info JSON Conversion		All Rights Reserved © 2017 SyncSols			
TCP Engine JSON to LabVIEW Data		All Rights Reserved © 2017 SyncSols			
TCP Engine JSON to LV Data		All Rights Reserved © 2017 SyncSols			

Name	Connector pane	Description	S.	R.	I.
TCP Engine Send Commands To all Modules		The purpose of this VI is to perform all functions of the hardware modules. Functions 1)Engine Launch 2)Configuration 3)Update 4)All Start 5)Engine start 6)Hardware Info. All Rights Reserved © 2017 SyncSols			
TCP Engine Send Loop		This VI will semd all engine response to HOST All Rights Reserved © 2017 SyncSols			
TCP Engine SM Q Driver		All Rights Reserved © 2017 SyncSols			
TCP Engine State Controller		All Rights Reserved © 2017 SyncSols			
TCP Engine Get Msg From Host		No description found (add content in vi description)			
TCP Engine		This VI will communicate with HOST through TCP/IP Communication, also it communicates with all engines through queue mechanism. All Rights Reserved © 2023 SyncSols			

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

3.2. JKI State Machines

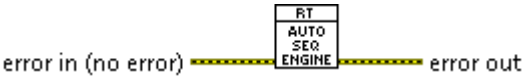
This section describes the JKI State Machines contained in the project.

3.2.1. Preamble

A JKI State Machine ™ is a State Machine built using the template provided by JKI.

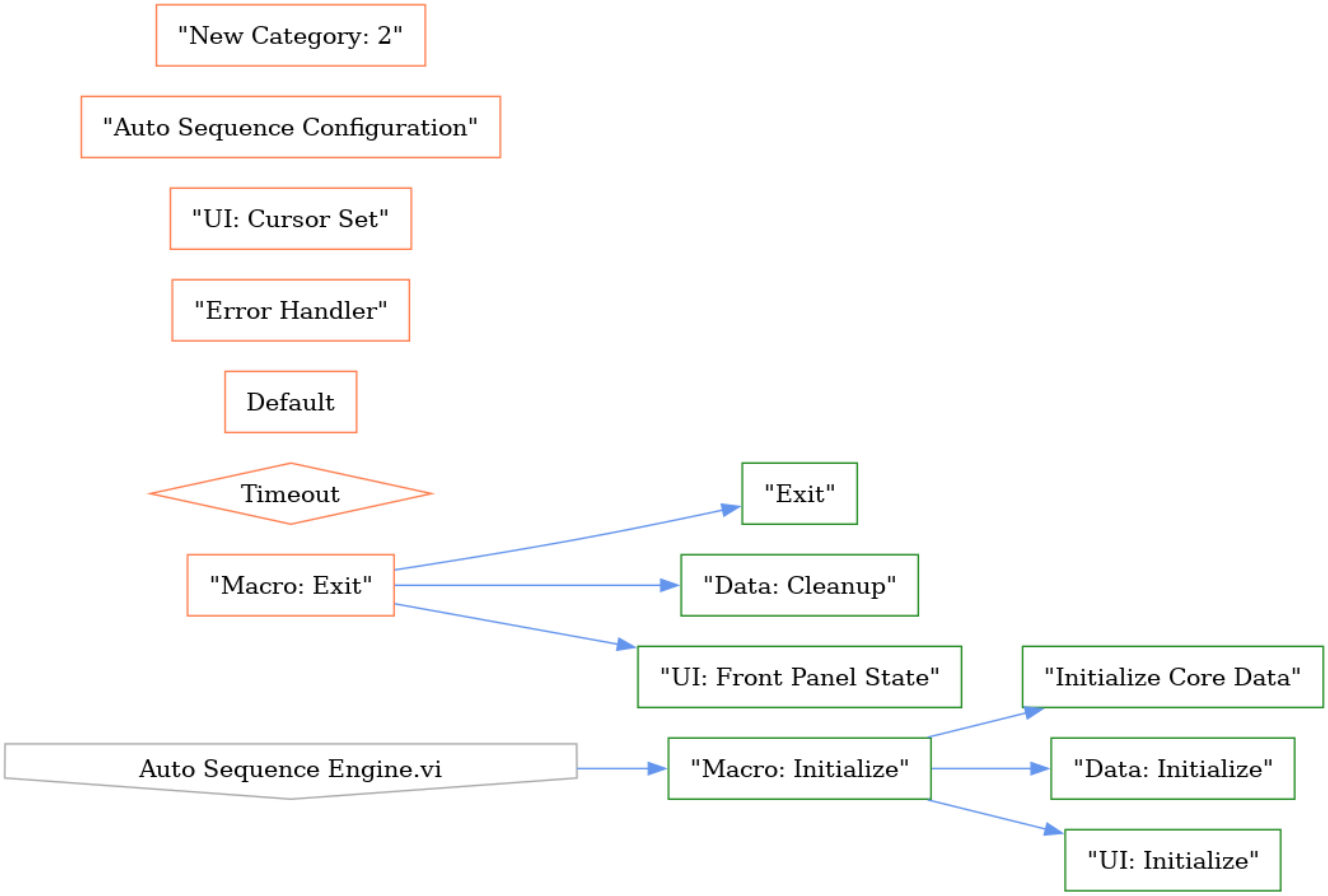
This section describes all JKI State Machine found in the project.

3.2.2. Auto Sequence Engine.vi



Description: No description found (add content in VI description)

States relathionship



State machine detailed info

State Name	State Documentation	State Callers	States Called
"", "Event Structure", "Idle"			
Timeout			
Core			
Default			
"Initialize Core Data"		"Macro: Initialize"	
"Error Handler"			
"Exit"		"Macro: Exit"	
Data			
"Data: Initialize"		"Macro: Initialize"	
"Data: Cleanup"		"Macro: Exit"	

State Name	State Documentation	State Callers	States Called
UI			
"UI: Initialize"		"Macro: Initialize"	
"UI: Cursor Set"			
"UI: Front Panel State"		"Macro: Exit"	
Macro			
"Macro: Initialize"	Initialization Macro (This is called once, when the VI starts)		"Initialize Core Data" "Data: Initialize" "UI: Initialize"
"Macro: Exit"			"Exit" "Data: Cleanup" "UI: Front Panel State"
Auto Sequence			
"Auto Sequence Configuration"			
"New Category: 2"			

Chapter 4. Legal Information

4.1. Document creation

This document has been generated using the following tools.

4.1.1. Antidoc

Project website: Antidoc

Maintainer website: Wovalab

BSD 3-Clause License

Copyright © 2019, Wovalab, All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.1.2. Asciidoc for LabVIEW™

Project website: Asciidoc toolkit

Maintainer website: Wovalab

BSD 3-Clause License

Copyright © 2019, Wovalab, All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.1.3. Graph Builder

Project website: Graph Builder

BSD 3-Clause License

Copyright © 2020, Cyril GAMBINI All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES

(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.2. Product used in the project

The documented project has been developed with the following products.

4.2.1. DQMH®

Copyright © 2021 DQMH® Consortium, LLC. All Rights Reserved.

Find more details on DQMH® Consortium website

4.2.2. JKI State Machine™

Copyright © 2018, JKI. All rights reserved.

Find more details on this page