



FieldPoint LabWindows/CVI Interface Help

April 2003 Edition, Part Number 370481B-01

This help file describes each function in the FieldPoint LabWindows/CVI interface.

To navigate this help file, use the **Contents**, **Index**, and **Search** tabs to the left of this window.

For more information about this help file, refer to the following topics:

[Conventions](#)—formatting and typographical conventions in this help file

[Important Information](#)



[Technical Support and Professional Services](#)

To comment on the documentation, email techpubs@ni.com

© 1998–2003 National Instruments Corporation. All rights reserved.

Conventions

The following conventions appear in this help file:

- [] Square brackets enclose optional items—for example, [response].
- » The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.
-  This icon denotes a note, which alerts you to important information.
-  This icon denotes a caution, which advises you of precautions to take to avoid injury, data loss, or a system crash.
- blue** Text in this color denotes a specific platform and indicates that the text following it applies only to that platform.
- bold** Bold text denotes items that you must select or click on in the software, such as menu items and dialog box options. Bold text also denotes parameter names, emphasis, or an introduction to a key concept.
- dark red** Text in this color denotes a caution.
- green Underlined text in this color denotes a link to a help topic, help file, or Web address.
- italic* Italic text denotes variables or cross references. This font also denotes text that is a placeholder for a word or value that you must supply.
- monospace Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames and extensions, and code excerpts.
- monospace bold** Bold text in this font denotes the messages and responses that the computer automatically prints to the screen. This font also emphasizes lines of code that are different from the other examples.
- monospace italic* Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

Using Functions from LabWindows/CVI

You must configure your device network in Measurement and Automation Explorer (MAX, at Program Files\National Instruments\MAX\NIMax.exe) before you use any of the functions from LabWindows/CVI. MAX stores all configuration information for the device network in a configuration (.iak) file. The FieldPoint LabWindows/CVI interface requires a valid .iak file to use the FieldPoint server from LabWindows/CVI.

[Configuring FieldPoint in MAX](#)

[FieldPoint LabWindows/CVI Function Reference](#)

[Error and Status Information](#)

Error and Status Information

Each function returns error codes and warnings through the return value of that function. A program must examine this value after each call to a FieldPoint LabWindows/CVI interface function to determine if an error occurred.

Possible error codes and their meanings appear with the corresponding function description.

FieldPoint LabWindows/CVI Function Reference

Class	Panel Name	Function Name
Server Management Server management functions manage the operations of a server. These functions create and destroy references to a particular server.	Open	FP_Open
	Close	FP_Close
I/O Point Management I/O point management functions manage the operations of I/O points. These functions create and destroy references to a particular I/O point.	Create Tag I/O Point	FP_CreateTagIOPoint
	Destroy I/O Point	FP_DestroyIOPoint
I/O Management I/O management functions manage reads and writes to I/O points. These functions perform synchronous and asynchronous read, write, and advise operations.	Read	FP_Read
	Read Cache	FP_ReadCache
	Advise	FP_Advise
	Change Advise State	FP_ChangeAdviseState
	Free PD Callback Buffer	FP_FreePDCallbackBuffer
	Stop	FP_Stop
	Write	FP_Write
	Error Message	FP_ErrorMsg

Data Types

LabWindows/CVI	FieldPoint
int	IAAttr
BOOLEAN	IABoolean
unsigned char	IAByte
unsigned long	IADeviceType
unsigned long	IAHandle
DWORD	IAParam
unsigned long	IAResult
char*	IAString
unsigned long	IATaskID
VARTYPE	IAType
double	IATimeStamp
unsigned long	IAVendor

Examples

FieldPoint examples for LabWindows/CVI are stored in the samples folder under your LabWindows/CVI directory.

FP_Advise

IAStatus FP_Advise (IAHandle serverHandle, IAHandle IOPointHandle, long adviseRate, IABoolean notifyOnChange, IAByte buffer[], unsigned long bufferSize, IABoolean callbackMethod, IAHandle callbackFunction, DWORD cParam, IATaskID *taskID);

Purpose

This function continuously reads time-stamped data at a specified rate from the I/O point into a buffer. Advise is an asynchronous operation. After FP_Advise initiates the operation, it immediately returns, and the client thread continues execution. The callback function executes according to the behavior you specify with **callbackMethod**. The advise operation will continue to monitor the I/O point at the rate you specify until you explicitly terminate it with a call to FP_Stop or FP_Close.

Use FP_ChangeAdviseState to suspend or resume user callbacks.

Use FP_ReadCache to read the last known value on the I/O point.

Use FP_Stop to terminate a single advise operation.



Note The recommended method of monitoring an I/O point is to schedule an advise on an I/O point, pass NULL for the function pointer, and use a timer on the UIR to periodically read the cache on the I/O point. When you use NULL for the **callbackFunction** pointer, your program ignores **callbackMethod**.

Parameter List

Name	Type	Description
serverHandle	IAHandle	Handle to a specific server session. Create the handle with FP_Open.
IOPointHandle	IAHandle	The handle that FP_CreateTagIOPoint returns. The handle identifies the I/O point described in this function.
adviseRate	long	Specifies the poll rate for the I/O point in milliseconds. An advise rate of 1,000 indicates that the I/O point is polled once per second.
notifyOnChange	IABoolean	Indicates when to execute the callback. When TRUE, the server invokes the callback only if the value changed from the last read. If it is FALSE, the server invokes the callback after every read. Valid Range: (0) FALSE - Notify Always (1) TRUE - Notify Only on Change
buffer	IAByte[]	Memory that you allocate. The server copies data into this buffer.
bufferSize	unsigned long	Indicates the number of bytes allocated in buffer .
callbackMethod	IABoolean	Specifies which type of callback mechanism the server uses for the specified Advise task. The FieldPoint LabWindows/CVI interface provides two different methods, as described below: (0) FALSE - PostDeferred Callbacks (1) TRUE - Asynchronous Callbacks
callbackFunction	IAHandle	A function pointer (as defined with the specification that follows) or a NULL value indicating no callback function. When this parameter is NULL, your program ignores callbackMethod .
cParam	DWORD	Parameter that you define to be passed to the callback. You can use this parameter to pass meaningful or useful data when processing the reply.
taskID	IATaskID	Created by FP_Advise and should be used when calling FP_ReadCache or FP_Stop.



Caution Do not use local data buffers, or the same data buffer for multiple advise operations, because it causes unpredictable results. It is a good idea to use distinct global data buffers for each advise task.

Parameter Discussion

PostDeferred Callbacks

This option is valid with LabWindows/CVI versions 5.0 and later. The server executes the callback in the main LabWindows/CVI thread each time LabWindows/CVI processes system events. A considerable delay could occur between the call to the function by the server and when LabWindows/CVI executes the function. Thus, for a large number of I/O points or fast advise rates, the PostDeferred queue may become full, causing the callbacks to be dropped. Call `FP_FreePDCallbackBuffer` in the callback function for post deferred callbacks to free the buffer allocated by the server manager.

Asynchronous Callbacks

The asynchronous call returns the data to the callback as soon as the server completes the operation. If you are using a version of LabWindows/CVI earlier than 5.5, observe the following guidelines for the callback function:

- The function must be short and must not call other I/O or UIR functions.
- Use only multithread-safe LabWindows/CVI libraries. For example, do not update UIR controls in an asynchronous callback.



Note The recommended method of monitoring an I/O point is to schedule an advise on an I/O point, pass NULL for the function pointer, and use a timer on the UIR to periodically read the cache on the I/O point. When you use NULL for the **callbackFunction** pointer, your program ignores **callbackMethod**.

callbackFunction

If you pass a callback function into this parameter and set the callback method to post deferred, LabWindows/CVI allocates memory for every callback. You must free this memory in the callback function using the `FP_FreePDCallbackBuffer` function. No memory is allocated for asynchronous callbacks, so you should not use the `FP_FreePDCallbackBuffer` function for asynchronous callbacks.

The function has the following definition:

```
void CVICALLBACK yourFunction (void* buffer);
```

The type of the function pointer is `FP_CallbackFuncPtr`:

```
FP_CallbackFuncPtr yourfuncPtr = yourFunction;
```

Enter '*yourfuncPtr*' in the **callbackFunction** control.

LabWindows/CVI returns the callback data as a pointer to void. To get back

meaningful data you must cast the void pointer to a struct of the following type:

```
FP_CallbackParamType;
```

The members of the struct are

```
IAHandle hIOpoint; //handle to the IOPoint
IAStatus status; //status
IABoolean CBMethod; //the mechanism for the callback
IAByte* buffer; //buffer containing the data
UInt32 buffersize; //size of databuffer
IATimeStamp timeStamp; //time stamp
DWORD cParam; //user specified parameter
```

For example:

```
FP_CallbackParamType g_param;
```

Use g_param to read back the data in your callback function:

```
void yourFunction (void* buffer)
{
    g_param = *((FP_CallbackParamType*) buffer); //cast data
    int bsize = g_param.buffersize;
    //bsize contains the buffer size
}
```

Return Value

The LabWindows/CVI manager or the server can return the following status codes. FP_ErrorMsg converts the status codes into descriptive strings.

LabWindows/CVI Manager-Level Error Codes	0x8480 IA_MGR_ERROR 0x8484 IA_MGR_SERVER_NOT_LOADED 0x8487 IA_MGR_INVALID_SERVER_HND
FieldPoint Server-Level Standard Error Codes	0x8000 to 0x83FF

Refer to the error section in your server help file for more information.

FP_ChangeAdviseState

IStatus FP_ChangeAdviseState (IAHandle serverHandle, IATaskID taskID, IABoolean callbackState);

Purpose

This function resumes or suspends user callbacks on all advise tasks or a specified advise task. The advise operation does not stop, and `FP_ReadCache` still returns the last known valid data on the I/O point.

Use this function to disable callbacks to your callback function without stopping the advise operation on an I/O point.

Parameter List

Name	Type	Description
serverHandle	IAHandle	Handle to a specific server session. Create the handle with FP_Open.
taskID	IATaskID	Task ID of the advise operation to enable or disable the callback. Enter zero to change the state of all current advise operations. FP_Advise returns this value when you initiate the advise operation. taskID remains valid after this function.
callbackState	IABoolean	Enables or disables user callbacks for advise operations you specified. Valid Range: (0) FALSE - Resume : Enable callbacks (1) TRUE - Suspend : Disable callbacks

Return Value

The LabWindows/CVI manager or the server can return the following status codes. FP_ErrorMsg converts the status codes into descriptive strings.

LabWindows/CVI Manager-Level Error Codes	0x8480 IA_MGR_ERROR 0x8484 IA_MGR_SERVER_NOT_LOADED 0x8487 IA_MGR_INVALID_SERVER_HND 0x8488 IA_MGR_INVALID_TASK_ID
FieldPoint Server-Level Standard Error Codes	0x8000 to 0x83FF

Refer to the error section in your server help file for more information.

FP_Close

IAStatus FP_Close (IAHandle serverHandle);

Purpose

This function closes a server session. `FP_Close` stops all running advise tasks on the server you specify. Destroy all I/O points using `FP_DestroyIOPoints` before you call `FP_Close`.

Parameter List

Name	Type	Description
serverHandle	IAHandle	Handle to a specific server session. Create the handle with FP_Open.

Return Value

The LabWindows/CVI manager or the server can return the following status codes. FP_ErrorMsg converts the status code to a descriptive string.

LabWindows/CVI Manager-Level Error Codes	0x8480 IA_MGR_ERROR 0x8484 IA_MGR_SERVER_NOT_LOADED 0x8485 IA_MGR_DLL_NOT_MAPPED 0x8487 IA_MGR_INVALID_SERVER_HND
FieldPoint Server-Level Standard Error Codes	0x8000 to 0x83FF

Refer to the error section in your server help file for more information.

FP_CreateTagIOPoint

IAStatus FP_CreateTagIOPoint (IAHandle serverHandle, IAString commName, IAString deviceName, IAString
itemName, IAHandle *IOPointHandle);

Purpose

This function creates an I/O point that represents a digital input, analog output, or other accessible data on a FieldPoint device using the configuration information of the named tag. You must have entered the tag information into the server configuration (.iak) file for this call to succeed. The server returns a handle to the new I/O point in hIOPoint.

Use FP_DestroyIOPoint to destroy the I/O point after all I/O operations have completed.

Parameter List

Name	Type	Description
serverHandle	IAHandle	Handle to a specific server session. Create the handle with FP_Open.
commName	IAString	Name of the communications resource configured in MAX.
deviceName	IAString	Name of the device configured in MAX.
itemName	IAString	Name of the item configured in MAX.
IOPointHandle	IAHandle (passed by reference)	Returned handle that represents the I/O point. Use the I/O point handle in read, read cache, write, and advise operations. Destroy the handle using FP_DestroyIOPoint.

Return Value

The LabWindows/CVI manager or the server can return the following status codes. FP_ErrorMsg converts the status codes into descriptive strings.

LabWindows/CVI Manager-Level Error Codes	0x8480 IA_MGR_ERROR 0x8484 IA_MGR_SERVER_NOT_LOADED 0x8487 IA_MGR_INVALID_SERVER_HND
FieldPoint Server-Level Standard Error Codes	0x8000 to 0x83FF

Refer to the error section in your server help file for more information.

FP_DestroyIOPoint

IAStatus FP_DestroyIOPoint (IAHandle serverHandle, IAHandle IOPointHandle);

Purpose

This function destroys an I/O point handle. Call this function on any handle you created with `FP_CreateTagIOPoint`. The I/O point handle is no longer valid after a call to this function.

Parameter List

Name	Type	Description
serverHandle	IAHandle	Handle to a specific server session. Create the handle with FP_Open.
IOPointHandle	IAHandle	Handle to an I/O point. FP_CreateTagIOPoint returns the handle.

Return Value

The LabWindows/CVI manager or the server can return the following status codes. FP_ErrorMsg converts the status codes into descriptive strings.

LabWindows/CVI Manager-Level Error Codes	0x8480 IA_MGR_ERROR 0x8484 IA_MGR_SERVER_NOT_LOADED 0x8487 IA_MGR_INVALID_SERVER_HND
FieldPoint Server-Level Standard Error Codes	0x8000 to 0x83FF

Refer to the error section in your server help file for more information.

FP_ErrorMsg

IAStatus FP_ErrorMsg (IAHandle serverHandle, IAStatus IAStatus, char errorMessage[]);

Purpose

This function copies a NULL-terminated ASCII message string that describes the corresponding provided status code in the buffer that you allocate.

Parameter List

Name	Type	Description
serverHandle	IAHandle	Handle to a specific server session. Create the handle with FP_Open.
IAStatus	IAStatus	Used to find an ASCII error string.
errorMessage	char []	Memory that you allocate. The server copies the error message into this buffer. Maximum error message size is 256 bytes.

Return Value

The LabWindows/CVI manager or the server can return the following status codes. FP_ErrorMsg converts the status codes into descriptive strings.

LabWindows/CVI Manager-Level Error Codes	0x8480 IA_MGR_ERROR 0x8484 IA_MGR_SERVER_NOT_LOADED 0x8487 IA_MGR_INVALID_SERVER_HND
FieldPoint Server-Level Standard Error Codes	0x8000 to 0x83FF

Refer to the error section in your server help file for more information.

FP_FreePDCallbackBuffer

IAStatus FP_FreePDCallbackBuffer (void *callbackBuffer);

Purpose

This function frees the post deferred callback buffer passed to the post deferred callback from the server manager.

Parameter List

Name	Type	Description
callbackBuffer	void *	The buffer passed to the post deferred callback from the server manager.

Return Value

The LabWindows/CVI manager or the server can return the following status codes. FP_ErrorMsg converts the status codes into descriptive strings.

LabWindows/CVI Manager-Level Error Codes	0x0 IA_SUCCESS 0x8480 IA_MGR_ERROR
FieldPoint Server-Level Standard Error Codes	0x8000 to 0x83FF

For details refer to the FieldPoint documentation or use the FP_ErrorMsg function.

FP_Open

IAStatus FP_Open (IAString configFilePath, IAHandle *serverHandle);

Purpose

This function opens a session with a server. Close the session with FP_Close.



Note Calling FP_Open more than once on the same server returns the same server handle.

Parameter List

Name	Type	Description
configFilePath	IAString	Path and name of the .iak configuration file created by MAX that contains the current FieldPoint configuration information. When this parameter is NULL, the server obtains its configuration information from the last configuration file used in MAX.
serverHandle	IAHandle (passed by reference)	Handle returned to identify a server. All other server functions use this handle.

Return Value

The LabWindows/CVI manager or the server can return the following status codes. FP_ErrorMsg converts the status codes into descriptive strings.

LabWindows/CVI Manager-Level Error Codes	0x8480 IA_MGR_ERROR 0x8481 IA_MGR_SERVER_DLL_NOT_FND 0x8482 IA_MGR_DLL_LOAD_FAILED 0x8483 IA_MGR_SERVER_LOAD_FAILED 0x8489 IA_MGR_FUNC_NOT_FND
FieldPoint Server-Level Standard Error Codes	0x8000 to 0x83FF

Refer to the error section in your server help file for more information.

FP_Read

IAStatus FP_Read (IAHandle serverHandle, IAHandle IOPointHandle, IABYTE buffer[], unsigned long bufferSize, SYSTEMTIME *timestamp);

Purpose

This function reads the value of the I/O point specified by the I/O point handle. This is a synchronous read.



Note The recommended method of monitoring an I/O point is to schedule an advise on an I/O point, pass NULL for the function pointer, and use a timer on the UIR to periodically read the cache on the I/O point. When you use NULL for the **callbackFunction** pointer, your program ignores **callbackMethod**.

Parameter List

Name	Type	Description
serverHandle	IAHandle	Handle to a specific server session. Create the handle with FP_Open.
IOPointHandle	IAHandle	Used to refer to an I/O point. You can create the handle with FP_CreateTagIOPoint.
buffer	IABYTE[]	Memory that you allocate. The server copies data into this buffer.
bufferSize	unsigned long	Size of the buffer that you created for the read operation. The buffer size should indicate the length (in bytes) of allocated memory pointed to by the buffer pointer.
timestamp	SYSTEMTIME (passed by reference)	Time when the FieldPoint server read the value. The timestamp is returned in the Windows SYSTEMTIME format.

Return Value

The LabWindows/CVI manager or the server can return the following status codes. FP_ErrorMsg converts the status codes into descriptive strings.

LabWindows/CVI Manager-Level Error Codes	0x8480 IA_MGR_ERROR 0x8484 IA_MGR_SERVER_NOT_LOADED 0x8487 IA_MGR_INVALID_SERVER_HND
FieldPoint Server-Level Standard Error Codes	0x8000 to 0x83FF

Refer to the error section in your server help file for more information.

FP_ReadCache

IAStatus FP_ReadCache (IAHandle serverHandle, IATaskID taskID, IABYTE buffer[], unsigned long bufferSize, SYSTEMTIME *timestamp);

Purpose

This function reads the last known value returned during an advise poll cycle. A valid task ID is necessary to invoke this function. You can call `FP_ReadCache` on a stopped advise operation or on one with suspended callbacks. For a stopped advise operation, this function returns the last valid value before the operation was stopped.

`FP_ReadCache` is useful for reading data and updating UIR controls when using the asynchronous callback mechanism.

Parameter List

Name	Type	Description
serverHandle	IAHandle	Handle to a specific server session. You create the handle with FP_Open.
taskID	IATaskID	Task ID of the advise operation for which to read the cache. FP_Advise returns the task ID.
buffer	IAByte[]	Memory that you allocate. The server copies data into this buffer.
bufferSize	unsigned long	Number of bytes allocated for the buffer. If the buffer size is too small, an error returns.
timestamp	SYSTEMTIME (passed by reference)	Time the data was retrieved from the physical device. The timestamp is returned in the Windows SYSTEMTIME format.



Caution Entering a constant or literal value may cause unpredictable results.

Return Value

The LabWindows/CVI manager or the server can return the following status codes. `FP_ErrorMsg` converts the status codes into descriptive strings.

LabWindows/CVI Manager-Level Error Codes	0x8480 IA_MGR_ERROR 0x8484 IA_MGR_SERVER_NOT_LOADED 0x8485 IA_MGR_DLL_NOT_MAPPED 0x8487 IA_MGR_INVALID_SERVER_HND 0x8488 IA_MGR_INVALID_TASK_ID 0x848A IA_MGR_BUFFER_TOO_SMALL
FieldPoint Server-Level Standard Error Codes	0x8000 to 0x83FF

Refer to the error section in your server help file for more information.

FP_Stop

IAStatus FP_Stop (IAHandle serverHandle, IATaskID taskID);

Purpose

This function stops an advise operation on the I/O point with the specified taskID. The taskID becomes invalid after a call to this function.

Parameter List

Name	Type	Description
serverHandle	IAHandle	Handle to a specific server session. Create the handle with FP_Open.
taskID	IATaskID	Task ID of the advise operation for which to read the cache. FP_Advise returns the task ID. The task ID becomes invalid after a call to FP_Stop.

Return Value

The LabWindows/CVI manager or the server can return the following status codes. `FP_ErrorMsg` converts the status codes into descriptive strings.

LabWindows/CVI Manager-Level Error Codes	0x8480 IA_MGR_ERROR 0x8484 IA_MGR_SERVER_NOT_LOADED 0x8487 IA_MGR_INVALID_SERVER_HND 0x8488 IA_MGR_INVALID_TASK_ID
FieldPoint Server-Level Standard Error Codes	0x8000 to 0x83FF

Refer to the error section in your server help file for more information.

FP_Write

IAStatus FP_Write (IAHandle serverHandle, IAHandle IOPointHandle, IABYTE buffer[], unsigned long bufferSize);

Purpose

This function writes data from the buffer you specify to an I/O point. The operation is synchronous and therefore blocks client execution until the write operation completes.

Parameter List

Name	Type	Description
serverHandle	IAHandle	Handle to a specific server session. Create the handle with FP_Open.
IOPointHandle	IAHandle	Used to reference an I/O point. You can create the handle with FP_CreateTagIOPoint.
buffer	IAByte[]	Buffer containing data you want to write to the server.
bufferSize	unsigned long	Size of the buffer that you created for the write operation. The buffer size should indicate the length (in bytes) of allocated memory pointed to by the buffer pointer.

Return Value

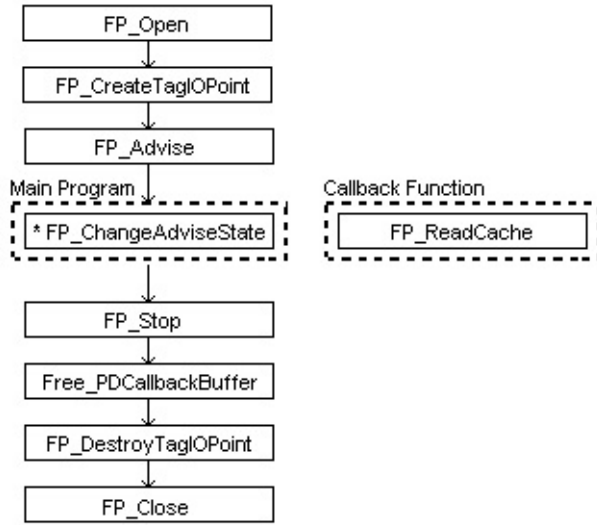
The LabWindows/CVI manager or the server can return the following status codes. FP_ErrorMsg converts the status codes into descriptive strings.

LabWindows/CVI Manager-Level Error Codes	0x8480 IA_MGR_ERROR 0x8484 IA_MGR_SERVER_NOT_LOADED 0x8487 IA_MGR_INVALID_SERVER_HND
FieldPoint Server-Level Standard Error Codes	0x8000 to 0x83FF

Refer to the error section in your server help file for more information.

Asynchronous Program Flow

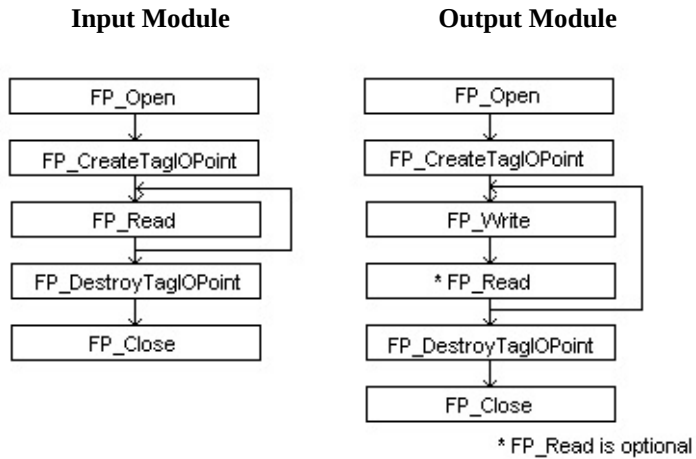
The following figure illustrates asynchronous LabWindows/CVI program flow.



* FP_ChangeAdviseState is optional

Synchronous Program Flow

The following figure illustrates synchronous LabWindows/CVI program flow for input and output modules.



Important Information

[Warranty](#)

[Copyright](#)

[Trademarks](#)

[Patents](#)

[Warning Regarding Use of NI Products](#)

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

Except as specified herein, National Instruments makes no warranties, express or implied, and specifically disclaims any warranty of merchantability or fitness for a particular purpose. Customer's right to recover damages caused by fault or negligence on the part of National Instruments shall be limited to the amount theretofore paid by the customer. National Instruments will not be liable for damages resulting from loss of data, profits, use of products, or incidental or consequential damages, even if advised of the possibility thereof. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's

failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

Trademarks

CVI™, FieldPoint™, National Instruments™, and ni.com™ are trademarks of National Instruments Corporation.

Product and company names mentioned herein are trademarks or trade names of their respective companies.

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the patents.txt file on your CD, or www.ni.com/patents

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) National Instruments products are not designed with components and testing for a level of reliability suitable for use in or in connection with surgical implants or as critical components in any life support systems whose failure to perform can reasonably be expected to cause significant injury to a human.

(2) In any application, including the above, reliability of operation of the software products can be impaired by adverse factors, including but not limited to fluctuations in electrical power supply, computer hardware malfunctions, computer operating system software fitness, fitness of compilers and development software used to develop an application, installation errors, software and hardware compatibility problems, malfunctions or failures of electronic monitoring or control devices, transient failures of electronic systems (hardware and/or software), unanticipated uses or misuses, or errors on the part of the user or applications designer (adverse factors such as these are hereafter collectively termed "system failures"). Any application where a system failure would create a risk of harm to property or persons (including the risk of bodily injury and death) should not be reliant solely upon one form of electronic system due to the risk of system failure. To avoid damage, injury, or death, the user or application designer must take reasonably prudent steps to protect against system failures, including but not limited to back-up or shut down mechanisms. Because each end-user system is customized and differs from National Instruments' testing platforms and because a user or application designer may use National Instruments products in combination with other products in a manner not evaluated or contemplated by National Instruments, the user or application designer is ultimately responsible for verifying and validating the suitability of National Instruments products whenever National Instruments products are incorporated in a system or application, including, without limitation, the appropriate design, process and safety level of such system or application.

Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at ni.com for technical support and professional services:

- **Support**—Online technical support resources include the following:
 - **Self-Help Resources**—For immediate answers and solutions, visit our extensive library of [technical support resources](#) available in English, Japanese, and Spanish at ni.com/support. These resources are available for most products at no cost to registered users and include software drivers and updates, a KnowledgeBase, product manuals, step-by-step troubleshooting wizards, conformity documentation, example code, tutorials and application notes, instrument drivers, discussion forums, a measurement glossary, and so on.
 - **Assisted Support Options**—[Contact NI engineers](#) and other measurement and automation professionals by visiting ni.com/support. Our online system helps you define your question and connects you to the experts by phone, discussion forum, or email.
- **Training**—Visit ni.com/custed for [self-paced tutorials, videos, and interactive CDs](#). You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, [NI Alliance Program](#) members can help. To learn more, call your local NI office or visit ni.com/alliance.

If you searched ni.com and could not find the answers you need, contact your [local office](#) or NI corporate headquarters. You can also visit the [Worldwide Offices](#) section of ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

Branch Offices

Office	Telephone Number
Australia	61 2 9672 8846
Austria	43 0 662 45 79 90 0
Belgium	32 0 2 757 00 20
Brazil	55 11 3262 3599
Canada (Calgary)	403 274 9391
Canada (Montreal)	514 288 5722
Canada (Ottawa)	613 233 5949
Canada (Québec)	514 694 8521
Canada (Toronto)	905 785 0085
Canada (Vancouver)	514 685 7530
China	86 21 6555 7838
Czech Republic	420 2 2423 5774
Denmark	45 45 76 26 00
Finland	385 0 9 725 725 11
France	33 0 1 48 14 24 24
Germany	49 0 89 741 31 30
Greece	30 2 10 42 96 427
Hong Kong	2645 3186
India	91 80 51190000
Israel	972 0 3 6393737
Italy	39 02 413091
Japan	81 3 5472 2970
Korea	82 02 3451 3400
Malaysia	603 9059 6711
Mexico	001 800 010 0793
Netherlands	31 0 348 433 466
New Zealand	64 09 914 0488
Norway	47 0 32 27 73 00
Poland	48 0 22 3390 150
Portugal	351 210 311 210
Russia	7 095 238 7139
Singapore	65 6 226 5886
Slovenia	386 3 425 4200
South Africa	27 0 11 805 8197
Spain	34 91 640 0085
Sweden	46 0 8 587 895 00

Switzerland	41 56 200 51 51
Taiwan	886 2 2528 7227
United Kingdom	44 0 1635 523545
United States (Corporate)	512 683 0100