234GB

CV-X Series

ActiveX Control

Reference Manual

Read this manual before use.
Keep this manual in a safe place for future reference.

Introduction

This manual explains the basic usage and operation of CV-X Series ActiveX Control (CVX Control). Information is provided on the handling and application programming interface of the CVX Control. Read this manual thoroughly in order to understand how the CVX Control works and to maximize performance of the system.

Related manuals

Also, read the manual for the CV-X Series that will be connected when using the CVX Control.

Symbols

The following warning symbols are used to ensure safety and to prevent human injury and/or damage to property when using the system.

DANGER	It indicates a hazardous situation which, if not avoided, will result in death or serious injury.
WARNING	It indicates a hazardous situation which, if not avoided, could result in death or serious injury.
CAUTION	It indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.
NOTICE	It indicates a situation which, if not avoided, could result in product damage as well as property damage.
Important	It indicates cautions and limitations that must be followed during operation.
Point	It indicates additional information on proper operation.
Reference	It indicates tips for better understanding or useful information.

Notation

Unless otherwise noted, the examples in this manual illustrate the use of properties, methods, event notifications, and source code according to Visual Basic conventions.
Additionally, the following notation is also used where necessary.

Notation	Description
Private Sub, True, Long	Keywords unique to Visual Basic are in bold.
Connect	Strings that should be typed as is are in bold.
filename, varlist	Settings, arguments, and other items that should be replaced with the applicable item are in italics.
object	The object keyword refers to an object type of the CVX Control.
[expressionlist]	Optional items that can be omitted are surrounded by square brackets [ ].
`MyString = "Hello, world!"`	Source code examples are in this font.

General Cautions

DANGER	Do not use this product for the purpose of protecting the human body or any part thereof.
CAUTION	• Before starting or operating the system, check to make sure all system functions are working properly. • If any Keyence product fails, take all safety precautions to prevent damage before using the system again.
NOTICE	• If the system is operated beyond its published specifications or if the system is modified, its functions and performance cannot be guaranteed. • Please note that when the system is used in combination with other instruments, its functions and performance may be degraded.

Cautions

- This software is designed to be used by personnel familiar with the CV-X Series Controller, and ActiveX programming experience using Windows and its corresponding development language.
- For information on developing Windows applications that use ActiveX controls, refer to documentation from Microsoft Corporation (US) and commercially available literature.
- This software does not support calling an API from multiple threads.
  Please call the API from a single thread only.

Trademarks

- Windows Vista, Visual Basic, Visual C++ and Visual C# are registered trademarks of Microsoft Corporation (US).
- Intel and Intel Core are trademarks of Intel Corporation in the United States and other countries.
- ActiveX is the trademark of the Microsoft Corporation, U.S.A. in the United States and/or other countries.
- "SD Memory Card" is a registered trademark of the SD Association.
- "QR Code" is the registered trademark of DENSO WAVE INCORPORATED.

Other company names and product names noted in this document are registered trademarks or trademarks of their respective companies. The ™ mark and ® mark have been omitted in this manual.

Software License Agreement

NOTICE TO USER: PLEASE READ THIS SOFTWARE LICENSE AGREEMENT ("THIS AGREEMENT") CAREFULLY. BY USING ALL OR ANY PORTION OF THE CV-X Series ActiveX Control ("THIS SOFTWARE"), YOU ARE AGREEING TO BE BOUND BY ALL THE TERMS AND CONDITONS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT USE THIS SOFTWARE.

If you install, copy and or use all or a portion of this Software on a device or computer, you will be deemed to have agreed to all the terms of this Agreement, and this Agreement will come into effect.

1. Definitions

1.1 "Use" or "using" refers to the accessing, installing, downloading, copying, operating and or otherwise benefiting from the utilization of this Software.

1.2 "This Software" denotes the software and all associated documentation provided by KEYENCE.

1.3 "User" or "User's" infers the company or persons who have had the license to use this Software granted to them by KEYENCE.

2. Grant of License.

2.1 In compliance with all of the terms and conditions of this Agreement, KEYENCE grants the non-exclusive and non-transferable license to install and use this Software.

2.2 The granting of the license permits the single reproduction and or copying of this Software for the User's backing up or archiving purposes.

2.3. This Software maybe installed on multiple devices and computers for the User's benefit and use.

2.4. The transfer of this Software via devices and computers with this Software installed on them by the User to third parties is permitted. In such a case, the third party who receives this Software may continue to use this Software in the same manner as the previous User.

2.5. With the transfer of this Software to a third party, the original User may install this Software to additional devices and computers for further use as required. In such a case, the third party who receives this Software may continue to use this Software in the same manner as the previous User.

2.6. The User maintains and guarantees that any thirdparties (as detailed in the previous two sections) who receive and use this Software agree to this license Agreement and comply with all the terms and conditions.

3. Restrictions.

3.1. This Software may not be modified by the User in any form except from the installation of updates or new functions provided by KEYENCE.

3.2. The reverse engineering, decompiling or disassembling of this Software by any persons are not permitted.

3.3. Without the prior permission of KEYENCE, the User may not reproduce or copy this Software for selling or distributing to a third party

4. Intellectual Property Rights.

Except as stated herein, KEYENCE reserves all rights, titles and interests in this Software, along with all associated copyrights, trademarks, and other intellectual property rights.

5. Disclaimer.

This Software is licensed to the User "AS IS" and without any warranty of any kind. In no event does KEYENCE or its suppliers accept or assume any liability for any damages, claims, costs or profit loss as a result of operating this Software.

6. Termination.

6.1 Under this Agreement the User's license will automatically terminate if this Software and any associated copies of this Software are destroyed or voluntarily returned to KEYENCE.

6.2 Under this Agreement the User's license will terminate immediately and automatically without any notice from KEYENCE if there is any failure to comply with any of the terms and conditions of this Agreement. Upon termination, the using of this Software shall cease, and all copies (full or partial) of this Software should be destroyed or returned to KEYENCE.

6.3 Compensation will be sought by KEYENCE should any violation or breach of this Agreement result in any incurred costs or lost profit to KEYENCE.

7. Governing Law.

7.1 This Agreement will be governed by and construed in accordance with the substantive laws of Japan.

7.2 If any part of this Agreement is found void and unenforceable, the rest of this Agreement will remain intact, valid and enforceable according to the associated terms and conditions.

Contents

Introduction

Related manuals 2
Symbols 2
Notation 3
General Cautions 3
Cautions 3
Trademarks 3

Software License Agreement

Overview of CV-X Series ActiveX Control

1. Before Using This Product

1.1. Major Functions 12
1.2. System Configuration 13
1.3. File Structure 13
1.4. Operating Environment 14
1.5. Version Compatibility 15

2. Development Steps

2.1. Installing/Uninstalling the ActiveX Control 17
2.1.1. Installing the ActiveX Control 17
2.1.2. Uninstalling the ActiveX Control 17
2.2. Loading the CVX Control into the Development Environment 18
2.3. Place the Control on a form 18

3. API List

3.1. Properties List 20
3.2. Methods List 21
3.3. Events List 22

Preparations

4. Common Settings for Communications

4.1. Operating Procedures 25
4.2. Initializing the Communication Function 25
4.3. Connecting to the Controller 26
4.4. Disconnecting From the Controller 27
4.5. Connection Limitations 28
4.5.1. Connecting from a single CVX Control to a controller 28
4.5.2. Connecting CVX Controls to a single controller 29
4.5.3. Concurrent connections with the CV-X Series Simulation-Software and CV-X Series Terminal-Software 30
4.6. Communication Errors 31
4.6.1. Communication exceptions 31
4.6.2. Execution errors 32
4.7. API Reference 33
4.7.1. Address property 33
4.7.2. Port property 34
4.7.3. Connected property 35
4.7.4. Initialize method 36
4.7.5. Connect method 37
4.7.6. Disconnect method 38

Typical Applications

5. Using the Remote Desktop

5.1. Operating Modes 41
5.2. Operating in Auto Update Mode 42
5.3. Operating in Manual Update Mode 43
5.4. Stopping the Remote Desktop 45
5.5. Operating the Controller via the Mouse 46
5.6. Saving an Image of the Remote Desktop Screen to a File 47
5.7. Clearing the Remote Desktop Screen 48
5.8. API Reference 49
5.8.1. RemoteDesktopStarted property 49
5.8.2. EnableRemoteMouseOperation property 50
5.8.3. StartRemoteDesktop method 51
5.8.4. StopRemoteDesktop method 52
5.8.5. UpdateRemoteDesktop method 53
5.8.6. CaptureRemoteDesktop method 54
5.8.7. ClearRemoteDesktop method 55
5.8.8. OnRemoteDesktopUpdated event 56

6 Receiving Results Data Output

6.1. To Start Receiving Results Data Output 58
6.2. Retrieving Results Data Output 60
6.3. To Stop Receiving Results Data Output 63
6.4. API Reference 64
6.4.1. ResultLogStarted property 64
6.4.2. StartResultLog method 65
6.4.3. StopResultLog method 66
6.4.4. UpdateResultLogData method 67
6.4.5. GetResultLogData method 68
6.4.6. OnResultLogDataReceived event 69
6.4.7. OnResultLogStopped event 70

7. Receiving Image Data Output

7.1. To Start Receiving Image Output Data 72
7.2. To Stop Receiving Image Output Data 74
7.3. API Reference 75
7.3.1. ImageLogStarted property 75
7.3.2. StartImageLog method 76
7.3.3. StopImageLog method 77
7.3.4. OnImageLogDataReceived event 78
7.3.5. OnImageLogStopped event 79

8. Retrieving Archived Image Data

8.1. Retrieving History Lists 81
8.2. Retrieving Archived Image Data 83
8.3. API Reference 85
8.3.1. ArchivedImageTransferBusy property 85
8.3.2. GetArchivedImageList method 86
8.3.3. GetArchivedImageInfo method 87
8.3.4. GetArchivedImage method 88
8.3.5. OnArchivedImageTransferred event 90

9. Retrieving Information of Setting Files

9.1. Retrieve Information of Program Setting (Library Setting) Files 92
9.1.1. Retrieving Information Lists 92
9.1.2. GetInspectList method 94
9.1.3. GetInspectInfo method 95
9.1.4. GetDictionaryList method 96
9.1.5. GetDictionaryInfo method 97

10.Sending/Receiving/Deleting Files

10.1.Retrieving a File List 99
10.2.Sending/Receiving Files 101
10.3.API Reference 103
10.3.1. FileTransferBusy property 103
10.3.2. GetFileList method 104
10.3.3. GetFileInfo method 105
10.3.4. DownloadFile method 106
10.3.5. UploadFile method 107
10.3.6. OnFileTransferred event 109
10.3.7. DeleteFile method 110

11.Issuing Non-Procedural Commands

11.1.Executing Commands 112
11.1.1 Executing synchronous methods 112
11.1.2. Executing asynchronous methods 114
11.2.Common Errors 115
11.3.API Reference 116
11.3.1. CommandBusy property 116
11.3.2. ExecuteCommand method 117
11.3.3. ExecuteCommandAsync method 118
11.3.4. OnCommandFinished event 119

12.Non-communication Functions

12.1.Reading the Version Number of the CVX Control 121
12.2.Adjusting the Position and Size of the CVX Control 121
12.3.Hiding the Control 121
12.4.Mouse and Keyboard Event Driven Processing 122
12.4.1. Mouse events 122
12.4.2. Keyboard events 122
12.5.API Reference 123
12.5.1. Version property 123

Appendix

Appendix

Error Codes 126
Illegal Characters 127
1. Illegal characters for folder names 127
2. Illegal characters for file names 127
Limitations When Using Multiple CVX Controls Simultaneously 128
1. Remote desktop 128
2. Results logging, Image logging 129
3. Other Communication Functions (Archived Image Data Transfers, File Transfers, Non-procedural Commands) 130

Overview of CV-X Series ActiveX Control

▶ Before Using This Product

Development Steps

API List

1. Before Using This Product

1.1. Major Functions

The CVX Control is an ActiveX control that operates in conjunction with the CV-X Series Controller over an Ethernet (TCP/IP) connection. CVX Control provides these major functions.

(1) Remote desktop

This allows monitoring and operation of the controller from the PC. Users can retrieve and display the monitor screen of the controller.

(2) Results and image logging

Users can receive and collect data output from the controller. Logging allows users to:

- store result data and image data outputted from the controller to a file.
- display and control operations based on results retrieved using the results logging function.

(3) Archived image data acquisition

Users can retrieve archived image data stored in the controller.

(4) File transfers

Users can transfer any desired file including those of program setting, registered image, library, global settings, learning data to and from the controller.

(5) Non-procedural commands

Users can issue non-procedural commands to the controller.

1.2. System Configuration

Figure 1.1 shows a typical system configuration that uses the CVX Control.

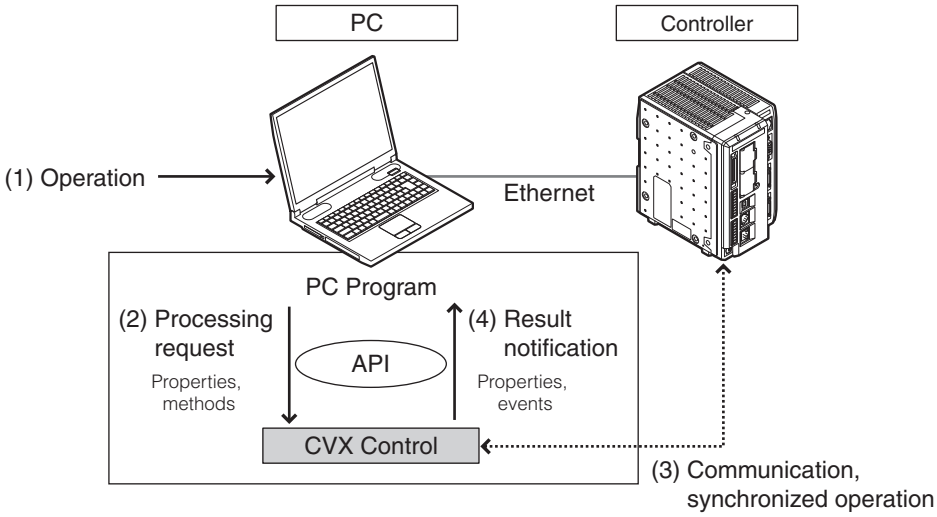


Figure 1.1 Typical System Configuration

(1) Users operate the PC Program through dialogs and the command line.

(2) The PC Program makes processing requests to the CVX Control through the API (application programming interface).

- The API for the CVX Control consists of properties, methods, and events.
- The PC program makes processing requests by setting properties and calling methods.
- Processing and status results are notified to the PC program through events and property values.

(3) The CVX Control works in conjunction with the CV-X Series Controller through a communication link.

(4) The CVX Control notifies processing results to the PC program using the API.

This manual explains the properties, methods, events, and operating specifications of the CVX Control.

1.3. File Structure

The CVX Control is provided as an OCX file.

CVXCTL.ocx

Before your program can use the CVX Control, the control must be registered in the Windows registry. See "2.1. Installing/Uninstalling the ActiveX Control" (Page 17) for details on registering the control to the registry.

1.4. Operating Environment

- PC programs that use the CVX Control are developed using a PC that runs the Windows operating system. Execute the created PC program as a Windows application on the PC.
- The CVX Control operates on applications developed in these environments.

Supported operating systems	Microsoft Windows Vista Home Basic, Home Premium, Business, Ultimate, Enterprise Microsoft Windows 7 Home Premium, Professional, Ultimate, Enterprise Microsoft Windows 10 Home, Pro, Enterprise For Microsoft Windows 7, Windows 10, the 64bit/32bit versions are supported. For all other operating systems, only the 32-bit version is supported. • Operating systems not listed are not supported. • English, Japanese, Traditional Chinese, Simplified Chinese, Korean, Thai, German, French, Italian, Mexican Spanish, Portuguese (Brazil), Vietnamese and Indonesian versions are supported. Other languages not listed here are not supported.
Programming languages	Microsoft Visual Basic 2008 (SP1) Microsoft Visual Basic 2010 Microsoft Visual Basic 2012 Microsoft Visual Basic 2013 Microsoft Visual C++ 2008 (SP1) Microsoft Visual C++ 2010 Microsoft Visual C++ 2012 Microsoft Visual C++ 2013 Microsoft Visual C# 2008 (SP1) Microsoft Visual C# 2010 Microsoft Visual C# 2012 Microsoft Visual C# 2013 • Changes may be made, without notice, to the environment listed here due to evaluation results.

- The CVX Control requires a personal computer that meets the following conditions.

Runtime environment	• CPU: Intel® Core™2 Duo 2 GHz or higher processor • RAM: 2 MB or higher • Available hard disk space: 100 MB or more (additional hard disk space required for saving received data) • Communication port: Ethernet interface • Users should apply Microsoft Visual C++ 2005 redistributable package (x86) SP1 vcredist_x86.exe.

1.5. Version Compatibility

- Functional enhancements may be offered in future releases of the CVX Control. Always check the version of the control when using.
- Newer versions are compatible with older versions. This allows PC programs developed for newer versions of the CVX Control to function with older versions.

Point

An error occurs if you try to add a new function only available in newer versions of the CVX Control, or place the control on a form for a project developed for an older version.
To resolve this issue, read the article at the URL shown below. (Article in English only. URL current as of October 2016.)
http://support.microsoft.com/kb/818220/en-us/

- Functions added to newer versions cannot be used in older versions. Therefore, PC programs that use functions added in newer versions of the CVX Control will not function with an older version of the CVX Control.

Before Using This Product

▶ **Development Steps**

APIList

2. Development Steps

The following are the steps for developing programs that use the CVX Control.

(1) Register the CVX Control in the registry.

(2) Launch the Visual Basic development environment and load the CVX Control to make it available in the project.

(3) Place the CVX control on a form.

(4) Program your application according to the application requirements.

This chapter explains steps (1), (2), and (3) above.

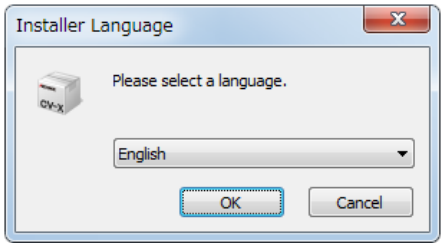
2.1. Installing/Uninstalling the ActiveX Control

2.1.1. Installing the ActiveX Control

(1) Download the latest file from the user support page.

User support page (user registration and login required)
http://www.keyence.com/special/download/cvx_activex.php

(2) Double-click the downloaded file.
The installer launches and extracts the installation files, and the language select screen appears.



(3) Select the language of the ActiveX Control to be installed, an click [OK].
The [Welcome] screen of the installation wizard appears.

Reference The version number may vary depending on the shipment time.

(4) Click [Next].
The [License Agreement] screen appears.

Reference See "Software License Agreement" (Page 4) for details on the user license agreement.

(5) Select [I accept the terms in the License Agreement], and then click [Next].
The screen in which the installation destination is selected appears.

(6) Proceed to the next operation according to the instructions on the screen.

(7) When the [Installation Complete] screen appears, click [Finish].
Installation of the ActiveX Control is now complete.

2.1.2. Uninstalling the ActiveX Control

Uninstall [CV-X Series ActiveX Control Ver.***] from [Add or Remove Programs] in [Control Panel] of the Windows.

When the control has been directly registered in the register without the installer

(1) Check the folder path including the CVXCTL.ocx file.

(2) Execute the following command from the command line (/u option).
`REGSVR32 /u` [folder path]`\CVXCTL.ocx`

Point The command must be executed by a user with System Administrator privileges.

2.2. Loading the CVX Control into the Development Environment

To develop an application using the CVX Control, you must load it into a development environment.

(1) Launch Visual Studio 2008/2010/2012/2013 and create a new project.

(2) From the [Tools] menu, select [Choose Toolbox Items].
The [Choose Toolbox Items] dialog box appears.

Reference: You can also display this dialog box by right-clicking on the Toolbox and selecting [Select Item] from the shortcut menu.

(3) Locate the [COM Components] tab and place a check in the [CVX Control], then click [OK].
The CVX Control icon is added to the Toolbox.

2.3. Place the Control on a form.

Select the CVX Control icon from the Toolbox and place it on the form. This will create an object for the CVX Control.

Point

- The control is shown as a black rectangle but during runtime, it changes to the controller's monitor screen when the remote desktop function is activated.
- You can reposition the control and change the control's size by using the mouse or property settings. You can also hide the control if the remote desktop function is not being used.
- Up to eight controls can be placed on a single application. Connect each control to a separate controller in order to communicate with multiple controllers simultaneously.

Reference

- See "12. Non-communication Functions" (Page 120) for more details on the CVX Control's position, size, and non-display settings.
- See "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when communicating from multiple CVX Controls.

3. API List

Before Using This Product

Development Steps

▶ API List

3.1. Properties List

The following list shows the major properties for the CVX Control.
Read the function explanation in each chapter for details on each property.

Table 3-1. Properties List

Function	Property	Controller version	Explanation
Connect/ Disconnect	Address	1.0	IP address of the controller at the connection destination (Page 33).
	Port	1.0	Port number of the controller at the connection destination (Page 34).
	Connected	1.0	Connection status with the controller (Page 35).
Remote desktop	RemoteDesktopStarted	1.0	Operating status of the remote desktop (Page 49).
	EnableRemoteMouseOperation	1.0*1	Enable mouse entry using the remote desktop function (Page 50).
Result logging	ResultLogStarted	1.0	Operating status of the result logging feature (Page 64).
Image logging	ImageLogStarted	1.0	Operating status of the image logging feature (Page 75).
Archived image data acquisition	ArchivedImageTransferBusy	1.0	Operating status of the archived image data acquisition feature (Page 85).
File transfer	FileTransferBusy	1.0	Operating status of file transfers (Page 103).
Non-procedural commands	CommandBusy	1.0	Operating status of non-procedural commands (Page 116).
Others	Version	1.0	Version of the CVX Control (Page 123).
Non-communication properties*	Top	1.0	Position of the control's upper edge.
	Bottom	1.0	Position of the control's bottom edge.
	Left	1.0	Position of the control's left edge.
	Right	1.0	Position of the control's right edge.
	Location	1.0	Position of the control's upper left corner.
	Width	1.0	Control width.
	Height	1.0	Control height.
	Size	1.0	Position of the control's width and height.
	Visible	1.0	Shows or hides the control.

*1 A later version of the controller is required for some functions. For details, see the explanation for each property.
Properties that have an asterisk or that are not included in the list are standard ActiveX controls.
For further details, read the technical literature from Microsoft Corporation.

3.2. Methods List

This list shows major methods of the CVX Control.

Read the function explanation in each chapter for more details on each method.

Table 3-2. Methods List

Function	Methods	Controller version	Explanation
Initialization	Initialize	1.0	Initializes the communication function (Page 36).
Connect/ Disconnect	Connect	1.0	Connects to the controller (Page 37).
	Disconnect	1.0	Disconnects from the controller (Page 38).
Remote desktop	StartRemoteDesktop	1.0	Starts the remote desktop (Page 51).
	StopRemoteDesktop	1.0	Stops the remote desktop (Page 52).
	UpdateRemoteDesktop	1.0	Updates the remote desktop (Page 53).
	CaptureRemoteDesktop	1.0	Saves an image of the remote desktop to a file (Page 54).
	ClearRemoteDesktop	1.0	Clears the remote desktop screen (Page 55).
Result logging	StartResultLog	2.0	Starts result logging (Page 65).
	StopResultLog	2.0	Stops result logging (Page 66).
	UpdateResultLogData	2.0	Specifies the target for acquiring result data output (Page 67).
	GetResultLogData	2.0	Retrieves result data output (Page 68).
Image logging	StartImageLog	1.0	Starts image logging (Page 76).
	StopImageLog	1.0	Stops image logging (Page 77).
Archived image data acquisition	GetArchivedImageList	1.0	Retrieves an information list of archived image data (Page 86).
	GetArchivedImageInfo	1.0	Retrieves information from the archived image data list (Page 87).
	GetArchivedImage	1.0*1	Retrieves archived image data (Page 88).
File information acquisition	GetInspectList	2.0	Retrieves an information list of inspection data (Page 94).
	GetInspectInfo	2.0	Retrieves information from the inspection data list (Page 95).
	GetDictionaryList	2.0	Retrieves an information list of library data (Page 96).
	GetDictionaryInfo	2.0	Retrieves information from the library list (Page 97).
File transfer/ File deletion	GetFileList	2.0	Retrieves the information list for files (Page 104).
	GetFileInfo	2.0	Retrieves information from the file list (Page 105).
	DownloadFile	2.0	Downloads a file (Page 106).
	UploadFile	2.0	Uploads a file (Page 107).
	DeleteFile	2.0	Deletes a file (Page 110).
Non-procedural commands	ExecuteCommand	2.0	Executes non-procedural commands. (Synchronous)
	ExecuteCommandAsync	2.0	Executes non-procedural commands. (Asynchronous)

*1 A later version of the controller is required for some functions. For details, see the explanation for each method.

Methods not included in the list are standard ActiveX control functions.

For further details, read the technical literature from Microsoft Corporation.

3.3. Events List

The following list shows major events for the CVX Control.

Read the function explanation in each chapter for more details on each event.

Table 3-3. Events List

Function	Events	Explanation
Remote desktop	OnRemoteDesktopUpdated	The remote desktop screen update (Page 56).
Result logging	OnResultLogDataReceived	Received results data output (Page 69).
	OnResultLogStopped	Stops result logging (Page 70).
Image logging	OnImageLogDataReceived	Received image output data (Page 78).
	OnImageLogStopped	Stops image logging (Page 79).
Archived image data acquisition	OnArchivedImageTransferred	Received transferred archived image data (Page 90).
File transfer	OnFileTransferred	File transfer completed (Page 109).
Non-procedural commands	OnCommandFinished	Executed a non-procedural command (Page 119).
Non-communication events *	ClickEvent	The mouse button was clicked (Page 122).
	DblClick	The mouse button was double-clicked (Page 122).
	MouseMoveEvent	The mouse was moved (Page 122).
	MouseDownEvent	The mouse button was depressed (Page 122).
	MouseUpEvent	The mouse button was released (Page 122).
	KeyDownEvent	A keyboard key was pressed (Page 122).
	KeyUpEvent	A keyboard key was released (Page 122).

Events that have an asterisk or that are not included in the list are standard ActiveX controls.

For further details, read the technical literature from Microsoft Corporation.

Reference

The events listed in this manual as "click", "mouse button (or keyboard) press", or "mouse button (or keyboard) release" are defined as follows.

- Click: Press followed by the release of a mouse button.
- Mouse button (or keyboard) press: This is the act of pressing (but not releasing) the mouse button (or keyboard key).
- Mouse button (or keyboard) release: This is the act of releasing the mouse button (or keyboard key).

Preparations

▶ **Common Settings for Communications**

4. Common Settings for Communications

This chapter explains the prerequisites for using the communication functions of the CVX Control, such as initialization, connection, and disconnection with the controller.
Make sure you read and understand the information in this chapter before using the functions explained in chapter 5 and beyond.

4.1. Operating Procedures

Your application should follow these operating procedures when using the CVX Control.
(Repeat steps (3) to (5) as necessary)

(1) Launch the application.
(2) Initialize communications.
(3) Connect to the controller.
(4) Execute the necessary communication functions.
(5) Disconnect from the controller.
(6) Exit the application.

This chapter explains steps (2), (3), and (5), which are the common procedures for using the communication functions.
The functions in Step (4) will be explained in chapter 5 and beyond.

4.2. Initializing the Communication Function

In order to use the communication functions provided by the CVX Control, you must initialize the communication function.

Related items

Initialize method (Page 36)

Procedures

(1) Call the Initialize method.

Sample source code

Initializing the communication function when the application starts.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As _
System.EventArgs) Handles MyBase.Load
     Dim state As Integer                    'for storing the return value
     state = CVX.Initialize()                'initializing communications

     If state = 0 Then                       'processing if this method succeeds
        Call MsgBox("Success")
     Else                                    'processing when an execution error
                                             occurs
        Call MsgBox("Execution error")       'application must be rebooted
     End If
End Sub
```

4.3. Connecting to the Controller

Establish the connection by specifying the target controller's IP address and port number.

Point Before you connect to the CV-X controller, make sure to set the IP address and port number (for connections with PC programs) in the controller's global settings, then turn the power on and connect it to the network.

Related items

Address property (Page 33), Port property (Page 34), Connect method (Page 37), Connected property (Page 35)

Procedures

(1) Set the Address property and Port property.

(2) Call the Connect method.

Sample source code

This code connects to a CV-X Series Controller when the Bn_Connect button is clicked.

```
Private Sub Bn_Connect_Click(sender As System.Object, e As _
System.EventArgs) Handles Bn_Connect.Click
     Dim state As Integer                    'for storing the return value
     With CVX
        .Address = "192.168.0.10"            'set the IP address
        .Port = 8502                         'set the port number
        state = .Connect()                   'connect to the controller
     End With

     If state = 0 Then                       'processing if this method succeeds
        Call MsgBox("Success")
     Else
     Call MsgBox("Execution error")          'processing when an execution error
                                             occurs
                                             'branch based on error code as
                                             necessary
     End If
End Sub
```

4.4. Disconnecting From the Controller

This procedure disconnects from the controller.

Related items

Disconnect method (Page 38), Connected property (Page 35)

Procedures

(1) Call the Disconnect method.

This method has no return value. The current state of the connection is reflected in the Connected property.

Sample source code

This code disconnects from the CV-X Series Controller when the Bn_Disconnect button is clicked.

```
Private Sub Bn_Disconnect_Click(sender As System.Object, e As _
System.EventArgs) Handles Bn_Disconnect.Click
    CVX.Disconnect()                        'disconnect from the controller
End Sub
```

4.5. Connection Limitations

This section explains the limitations that apply when using the Connect method to connect to a CV-X Series Controller.

4.5.1. Connecting from a single CVX Control to a controller

The CVX Control can connect to only one controller.
It does not support connections to more than one controller at the same time.

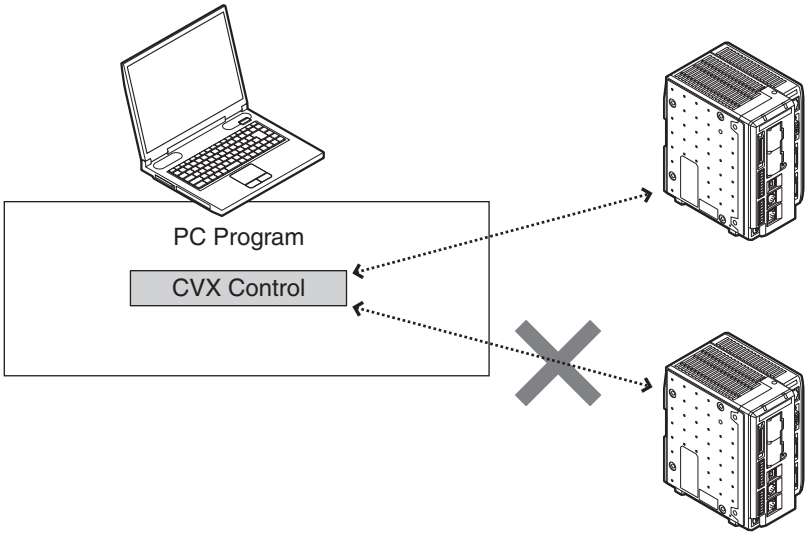


Figure 4-1. Simultaneous connections to multiple controllers not supported

If the Connect method is called to connect to another controller when the control is already connected to a controller, the call will fail (1003 illegal operation error). To connect to another controller, disconnect the controller that is currently connected first.

To connect to multiple controllers simultaneously, use several CVX Controls as described below and connect each control to a single controller.

- Place multiple CVX Controls on a single application, or
- Launch multiple instances of an application embedded with a single CVX Control.

4.5.2. Connecting CVX Controls to a single controller

A single controller can connect to up to four CVX Controls simultaneously. However, each CVX Control must be running on a separate PC.

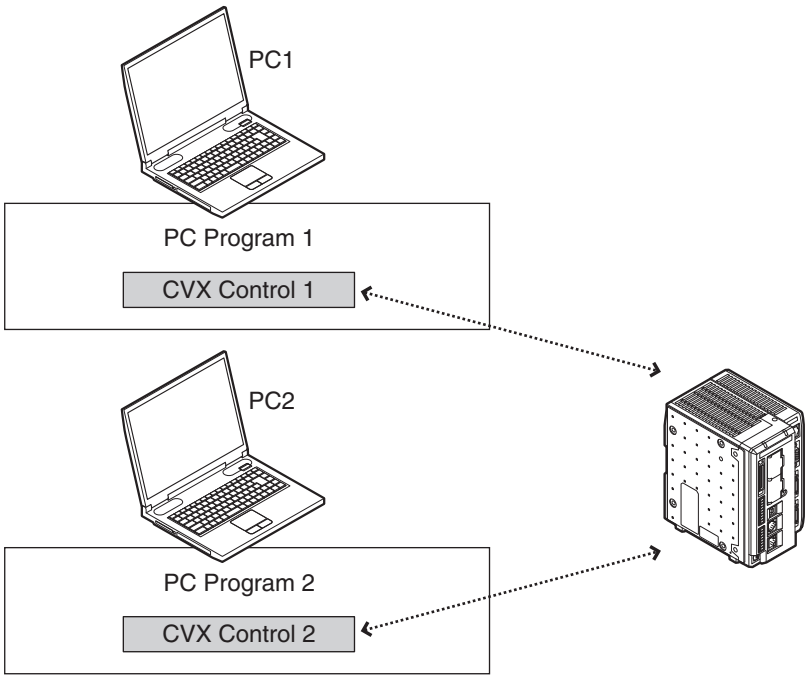


Figure 4-2. Multiple connections supported for separate PCs

If multiple instances of an application embedded with the CVX Control connect to the same controller (from the same PC), the most recent connection takes priority. All earlier connections are disconnected.

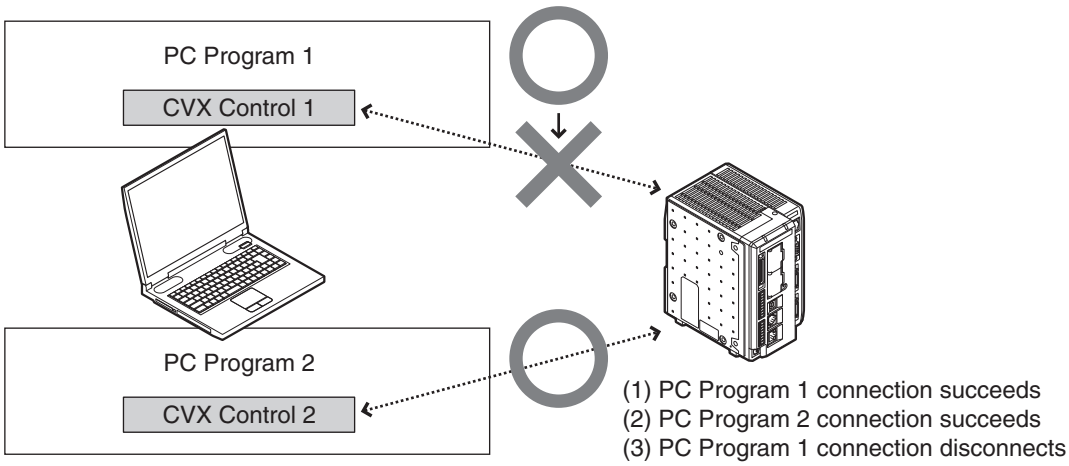


Figure 4-3. Connecting to the same controller from multiple instances on the same PC

If multiple CVX Controls on the same application connect to the same controller (from the same PC), the first connection has priority, and all later connections will fail.

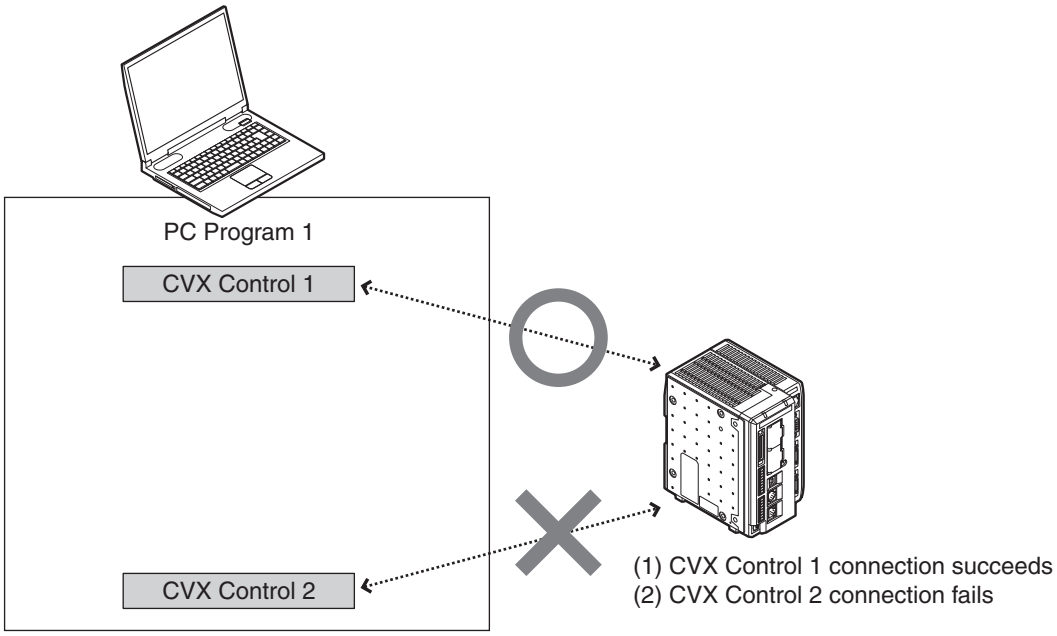


Figure 4-4. Connecting to the same controller from multiple controls on the same application

4.5.3. Concurrent connections with the CV-X Series Simulation-Software and CV-X Series Terminal-Software

An application that uses the CVX Control can connect to a single controller while that controller is connected to the CV-X Series Simulation-Software and CV-X Series Terminal-Software. A single controller can connect to up to four PC programs.

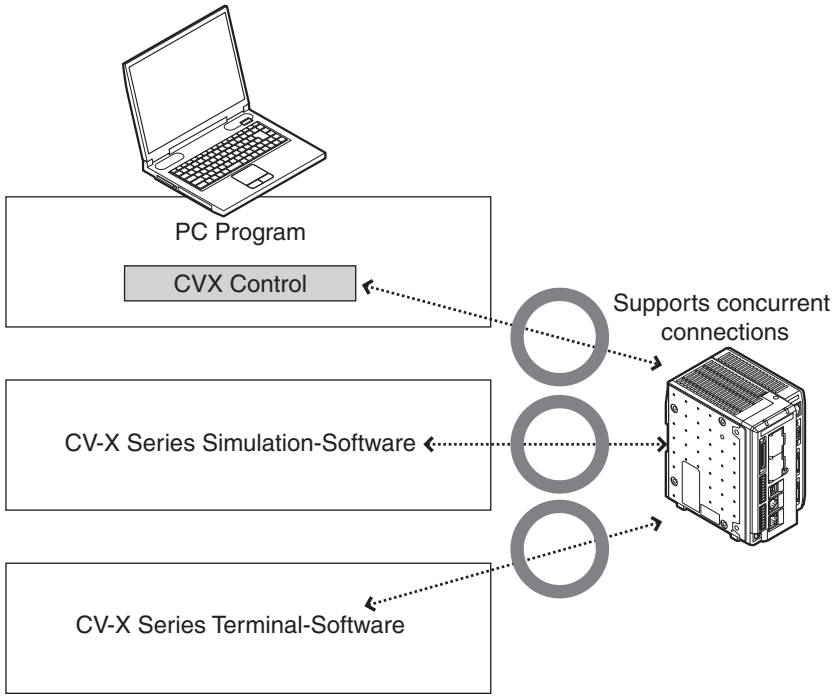


Figure 4-5. Concurrent Connections with the CV-X Series Simulation-Software and CV-X Series Terminal-Software

Point Although the controller will maintain concurrent connections, it may not be able to service logging and other communication functions simultaneously.
See "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when running these functions from multiple PCs.

4.6. Communication Errors

There are two types of communication errors that can occur when using the communication functions. The type of error determines how it should be handled.

(1) Communication exceptions

- These errors occur when communications with the controller are unavailable.
- The connection with the controller must be re-established to continue processing.

(2) Execution errors

- These errors occur when each function fails its processing task.
- Processing can continue because the connection is still maintained.
- The connection with the controller does not have to be re-established.

This section describes the cause of each error, how the API reacts, and troubleshooting.

Reference

- The cause of an error and the error code notified by each method and/or event will differ according to the function used. See the API reference on each function for more details on errors.
- See "Error Codes" (Page 126) for a complete list of error codes.

4.6.1. Communication exceptions

Communication exceptions are errors that indicate communications with the controller are unavailable.

- **Communication failures caused by a physical disconnection, such as a disconnected Ethernet cable**
  Indicates there is a physical disconnection in the communication path to the controller, such as a disconnected cable at the PC, controller, or a network hub or repeater.
- **Communication failures caused by the Ethernet communication environment, such as a communication overload**
  Indicates the connection cannot be maintained due to poor response, typically caused when there is an overload in data volume or a faulty cable.
- **Disconnected connections caused by connection limits or other problems not mentioned above**
  The connection with the controller is unconditionally disconnected due to problems outside the communication environment. This can occur when the controller IP address is changed, or when the connection is limited. (See "4.4. Disconnecting From the Controller" (Page 27) and "4.5. Connection Limitations" (Page 28).) Communication exceptions can also occur if the controller is under a heavy inspection load and cannot service communications.

When a communication exception occurs, the CVX Control performs the following actions depending on the timing of the error.

(1) If the communication exception occurs after calling a method but before the return value notification, the method's return value will be error code 1100 (communication exception).

(2) If the communication exception occurs while waiting for a process completed event, the event status code will be error code 1100 (communication exception).

(3) In any other instance (when communication processing is idle),

- the API will not notify any error codes.
- once the error occurs, events or other means of notification are unavailable.
- due to the connection being lost when the communication exception occurred, the return value for the next method called will be error code 1001 (process failed) because the connection does not exist.

Point
- When a communication exception occurs, the connection with the controller is disconnected.
- If an error notification from the API is detected as described above, reconnect to the controller using the Connect method.

4.6.2. Execution errors

An execution error is an error that does not qualify as a communication exception. See the API reference on each function for more details on error causes by functions.

These error causes are common for all functions.

- **If the communication function has not been initialized**
  the return value for any method called will be error code 1001 (processing failed).
- **If the connection to the controller has not been established**
  the return value for any method called immediately after a communication exception occurs or before establishing a connection with the Connect method, will be error code 1001 (processing failed).

When an execution error occurs, the CVX Control performs the following actions depending on the timing of the error. The execution error code notified will be a value other than 1100 (communication exception).

(1) If the execution error occurs after calling a method but before the return value notification,
the error code in the return value will identify the error.

(2) If the execution error occurs while waiting for a processing completed event,
the error code in the event status will identify the error.

Point
- When an execution error occurs, the method or event handler that caused the error is void.
- The API can still be used because the connection to the controller is maintained.

4.7. API Reference

4.7.1. Address property

Function

Sets the IP address of the target CV-X Series Controller as a text string.

- The value can also be read.
- This property is referenced when the Connect method (Page 37) is called.

Reference

See "4.3. Connecting to the Controller" (Page 26) for more details on the use of this property.

Syntax

*object.***Address** [= *string*]

Data type

String

Arguments

string	Specify the controller IP address as a text string delimited by periods. (Example: "192.168.0.10"). The maximum string length is 15 characters. IP addresses must include 3 periods and numbers must be from 0 to 255. If the string does not satisfy these conditions, the return value for the Connect method will be 1002 (illegal argument error).

Initial value

The initial state of this property is null. Be sure to set this property before connecting to the controller.

Related items

Connect method (Page 37)

4.7.2. Port property

Function

Sets the port number of the target CV-X Series Controller.

- The value can also be read.
- This property is referenced when the Connect method (Page 37) is called.

Reference See "4.3. Connecting to the Controller" (Page 26) for more details on the use of this property.

Syntax

object.**Port** [= *port*]

Data type

Integer

Arguments

port	Sets the port number of the controller. The setting range is 1 to 65533. If the value is outside the valid setting range, the return value for the Connect method will be 1002 (illegal argument error).

Initial value

8502

Related items

Connect method (Page 37)

4.7.3. Connected property

Function

Indicates the connection status with the controller.

- The value is read-only.
- This property is updated when the communication status changes due to calling the Connect/Disconnect methods, or when an error occurs.

Point

- If the Ethernet cable is unplugged when the connection status is True, the property will not immediately reflect False.
- The property will change to False after detecting the communication exception when a method is called, or an event is notified.
- Communication exceptions may occur even when this property is True.

Syntax

*object.***Connected**

Data type

Boolean

Arguments

True	Indicates the link is connected.
False	Indicates the link is disconnected.

Initial value

False

Related items

Connect method (Page 37), Disconnect method (Page 38)

4.7.4. Initialize method

Function

Initializes the communication function of the CVX Control.

- Call this method after launching the application, before using the communication functions.
- The method will fail if sufficient memory cannot be allocated.
- If this method fails, subsequent calls will also fail.

  The application must be restarted.

Reference See "4.2. Initializing the Communication Function" (Page 25) for more details on the use of this property.

Syntax

*object.***Initialize() As Integer**

Arguments

None

Return value

0	Processing succeeded	
1001	Processing failed	Failed to allocate sufficient memory.

Related items

None

4.7.5. Connect method

Function

Connects to the CV-X Series Controller.

- The Address property (Page 33) and Port property (Page 34) must be set beforehand.
- The connection status will be reflected in Connected property (Page 35).

Reference See "4.3. Connecting to the Controller" (Page 26) for more details on the use of this property.

Syntax

*object.***Connect() As Integer**

Arguments

None

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. Controller does not exist at the destination. Failed to connect to the specified port. Communication failure due to a fault in the communication path.
1002	Illegal argument	Incorrect format used for Address and/or Port properties.
1003	Illegal operation	A connection to the controller already exists.

Related items

Address property (Page 33), Port property (Page 34), Connected property (Page 35)

4.7.6. Disconnect method

Function

Disconnects from the currently connected controller.

- Be sure to disconnect the link before exiting the application.
- The connection status will be reflected in Connected property (Page 35).

Reference See "4.4. Disconnecting From the Controller" (Page 27) for more details on the use of this property.

Syntax

*object.***Disconnect ()**

Arguments

None

Return value

None

Related items

Connected property (Page 35)

Typical Applications

▶ Using the Remote Desktop

Receiving Results Data Output

Receiving Image Data Output

Retrieving Archived Image Data

Retrieving Information of Setting Files

Sending/Receiving/Deleting Files

Issuing Non-Procedural Commands

Non-communication Functions

5. Using the Remote Desktop

Remote desktop is a function used to display the monitor screen of the CV-X Series Controller on the CVX Control window via a communications link.

Users can save the content displayed in the window, and operate the controller by operating the mouse on the screen via communication.

- The communication link must be setup before you can use the functions in this chapter.
  Read "4. Common Settings for Communications" (Page 24) to initialize communication functions and connect to the controller.
- See "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

5.1. Operating Modes

The Remote Desktop function has two operating modes based on how the screen is updated: Auto and Manual.

Auto Update Mode

Updated screen data is sent to the CVX Control to update it whenever the controller screen updates. This mode reflects all changes on the controller screen and should be used when real-time control is desired.

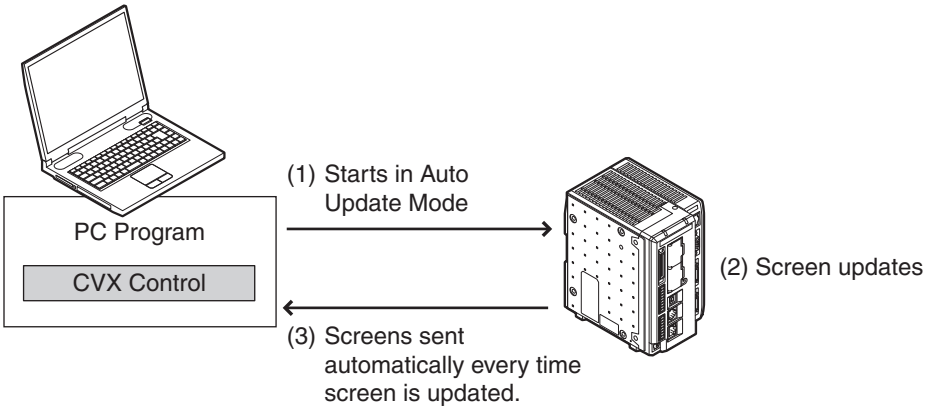


Figure 5-1. Auto Update Mode

Point
- If the CV-X controller screen updates very frequently, the communication load due to screen transfers will increase and may cause update delays.

Manual Update Mode

Updated screen data is sent to the CVX Control when calls are made to the UpdateRemoteDesktop method (Page 53). This mode should be used when you prefer to monitor the controller screen periodically.

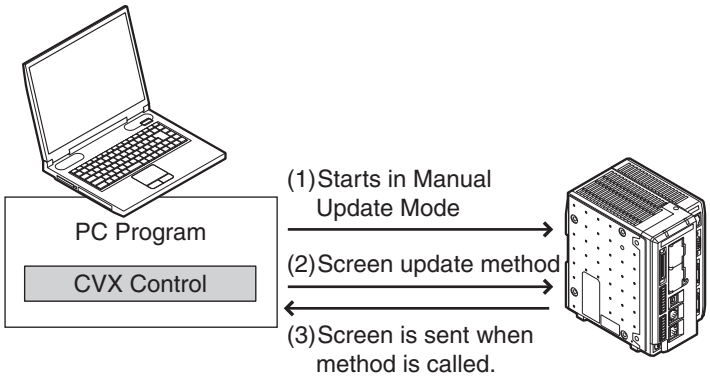


Figure 5-2. Manual Update Mode

Point
- Although this mode does not reflect all changes on the controller screen, it does allows you to manage the communication load by controlling the calling frequency of the UpdateRemoteDesktop method.
- By updating the Remote Desktop less often, you can lessen the impact on the updating speed of the controller screen.

5.2. Operating in Auto Update Mode

Related items

StartRemoteDesktop method (Page 51), OnRemoteDesktopUpdated event (Page 56)

Procedures

Connect to the controller. (See "4.3. Connecting to the Controller" (Page 26)).

(1) Specify the Auto Update Mode and call StartRemoteDesktop method (Page 51).
When the screen update completes, the OnRemoteDesktopUpdated event (Page 56) occurs.

Sample source code

The following code starts the Remote Desktop in Auto Update Mode when the Bn_Start button is clicked.
The OnRemoteDesktopUpdated event occurs each time the screen updates.

```
Private Sub Bn_Start_Click(sender As System.Object, e As _
System.EventArgs) Handles Bn_Start.Click
    'Establish the connection with the controller beforehand
    Dim state As Integer                    'for storing the return value
    state = CVX.StartRemoteDesktop(0)
                                            'start Remote Desktop (Auto Update
                                            Mode)
    If state = 0 Then                       'processing if this method succeeds
       Call MsgBox("Success")
    ElseIf state = 1100 Then                'processing when a communication
                                            exception occurs
       Call MsgBox("Communication exception")  'connection to the controller must be re-
                                            established
    Else
       Call MsgBox("Execution error")       'processing when an execution error
                                            occurs
       'branch based on error code as necessary
    End If
End Sub

Private Sub CVX_OnRemoteDesktopUpdated(sender As System.Object, e As _
AxCVXLib._DCVXEvents_OnRemoteDesktopUpdatedEvent) Handles _
CVX.OnRemoteDesktopUpdated
    'Screen update completed
    If e.state = 0 Then                     'processing if this method succeeds
       Call MsgBox("Success")
    Else                                    'processing when a communication
                                            exception occurs
       Call MsgBox("Communication exception")  'connection to the controller must be re-
                                            established
    End If
End Sub
```

5.3. Operating in Manual Update Mode

Related items

StartRemoteDesktop method (Page 51), UpdateRemoteDesktop method (Page 53), OnRemoteDesktopUpdated event (Page 56)

Procedures

Connect to the controller. (See "4.3. Connecting to the Controller" (Page 26)).

(1) Specify the Manual Update Mode and call StartRemoteDesktop method (Page 51).

(2) Call the UpdateRemoteDesktop method (Page 53).

The controller sends the monitor screen which updates the Remote Desktop.

When the screen update completes, the OnRemoteDesktopUpdated event (Page 56) occurs.

Sample source code

```
Private Sub Bn_Start_Click(sender As System.Object, e As _
System.EventArgs) Handles Bn_Start.Click
     'Establish the connection with the controller beforehand
     Dim state As Integer                          'for storing the return value
     state = CVX.StartRemoteDesktop(1)
                                                   'start Remote Desktop (Manual Update
                                                   Mode)
     If state = 0 Then                             'processing if this method succeeds
        Call MsgBox("Success")
     ElseIf state = 1100 Then                      'processing when a communication
                                                   exception occurs
        Call MsgBox("Communication exception")     'connection to the controller must be re-
                                                   established
     Else
        Call MsgBox("Execution error")             'processing when an execution error
                                                   occurs
        'branch based on error code as necessary
     End If
End Sub

Private Sub Bn_Update_Click(sender As System.Object, e As _
System.EventArgs) Handles Bn_Update.Click
     'Start the Remote Desktop (Manual Update Mode) beforehand
     Dim state As Integer                          'for storing the return value
     state = CVX.UpdateRemoteDesktop()
                                                   'update screen
     If state = 0 Then                             'processing if this method succeeds
        Call MsgBox("Success")
     ElseIf state = 1100 Then                      'processing when a communication
                                                   exception occurs
        Call MsgBox("Communication exception")     'connection to the controller must be re-
                                                   established
     Else
        Call MsgBox("Execution error")             'processing when an execution error
                                                   occurs
        'branch based on error code as necessary
     End If
End Sub
```

```
Private Sub CVX_OnRemoteDesktopUpdated(sender As System.Object, _
e As AxCVXLib._DCVXEvents_OnRemoteDesktopUpdatedEvent) Handles _
CVX.OnRemoteDesktopUpdated
    'Screen update completed
    If e.state = 0 Then                          'processing if this method succeeds
       Call MsgBox("Success")
    Else                                         'processing when a communication
                                                 exception occurs
       Call MsgBox("Communication exception")    'connection to the controller must be re-
                                                 established
    End If
End Sub
```

5.4. Stopping the Remote Desktop

Related items

StopRemoteDesktop method (Page 52)

Procedures

This procedure assumes the Remote Desktop has been started using the StartRemoteDesktop method (See "5.2. Operating in Auto Update Mode" (Page 42) and "5.3. Operating in Manual Update Mode" (Page 43)).

(1) Call the StopRemoteDesktop method.

If you call StopRemoteDesktop while waiting for the OnRemoteDesktopUpdated event (in the case the Remote Desktop was started in Manual Update mode and you called the UpdateRemoteDesktop method), the call will fail with a return value of 1003 illegal operation error. Instead, you should wait for the OnRemoteDesktopUpdated event to be notified before calling the StopRemoteDesktop method.

Sample source code

The following code stops the Remote Desktop when the Bn_Stop button is clicked.

```
Private Sub Bn_Stop_Click(sender As System.Object, e As _
System.EventArgs) Handles Bn_Stop.Click
    'This code assumes the Remote Desktop has been started
    Dim state As Integer                    'for storing the return value
    state = CVX.StopRemoteDesktop()         'stop Remote Desktop
    If state = 0 Then                       'processing if this method succeeds
       Call MsgBox("Success")
    ElseIf state = 1100 Then                'processing when a communication
                                            exception occurs
       Call MsgBox("Communication exception")  'connection to the controller must be re-
                                            established
    Else
       Call MsgBox("Execution error")       'processing when an execution error
                                            occurs
       'branch based on error code as necessary
    End If
End Sub
```

5.5. Operating the Controller via the Mouse

The remote desktop function can send mouse operation to the connected CV-X Series Controller and can operate it.

Related items

StartRemoteDesktop method (Page 51), EnableRemoteMouseOperation property (Page 50)

Procedures

(1) Enable remote mouse operation by setting the EnableRemoteMouseOperation property (Page 50).
Remote mouse operation is not dependent on the Remote Desktop operating mode (Auto or Manual).

(2) Start the Remote Desktop by calling the StartRemoteDesktop method (Page 51).
Screen operation with the mouse is enabled at this time.

Sample source code

This code enables remote mouse operation when the Bn_Start button is clicked, and starts the Remote Desktop.

```
Private Sub Bn_Start_Click(sender As System.Object, e As _
System.EventArgs) Handles Bn_Start.Click
    'Establish the connection with the controller beforehand
    Dim state As Integer                        'for storing the return value
    CVX.EnableRemoteMouseOperation = True       'enabling remote mouse operation
    state = CVX.StartRemoteDesktop(0)
                                                'start Remote Desktop (Auto Update
                                                Mode)
    Error processing (omitted)
End Sub
```

5.6. Saving an Image of the Remote Desktop Screen to a File

Related items

CaptureRemoteDesktop method (Page 54)

Procedures

(1) Call the CaptureRemoteDesktop method.

Sample source code

The following code captures the Remote Desktop screen to a file when the Bn_Capture button is clicked.

```
Private Sub Bn_Capture_Click(sender As System.Object, _
e As System.EventArgs) Handles Bn_Capture.Click
     Dim state As Integer                    'for storing the return value
     'saving the Remote Desktop
     state = CVX.CaptureRemoteDesktop("C:\Image\MyImage.bmp")

     If state = 0 Then                       'processing if this method succeeds
        Call MsgBox("Success")
     Else
        Call MsgBox("Execution error")       'processing when an execution error
                                             occurs
        'branch based on error code as necessary
     End If
End Sub
```

5.7. Clearing the Remote Desktop Screen

Related items

ClearRemoteDesktop method (Page 55)

Procedures

(1) Call the ClearRemoteDesktop method.
The screen shown on the CVX Control is initialized (to a black screen), regardless of the status of the connection with the controller, or Remote Desktop.

Sample source code

The following code clears the Remote Desktop screen when the Bn_Clear button is clicked.

```
Private Sub Bn_Clear_Click(sender As System.Object, e As _
System.EventArgs) Handles Bn_Clear.Click
     CVX.ClearRemoteDesktop()                    'initialize the Remote Desktop
End Sub
```

5.8. API Reference

5.8.1. RemoteDesktopStarted property

Function

Indicates the operating status of the Remote Desktop function (read-only).

This property is updated when the StartRemoteDesktop method (Page 51) and StopRemoteDesktop method (Page 52) are called to start and stop the Remote Desktop.

Syntax

*object.***RemoteDesktopStarted**

Data type

Boolean

Arguments

True	The Remote Desktop is running.
False	The Remote Desktop is stopped.

Initial value

False

Related items

StartRemoteDesktop method (Page 51), StopRemoteDesktop method (Page 52)

5.8.2. EnableRemoteMouseOperation property

Function

Toggles mouse operation between enable and disable. (The value can also be read.)

Specifying True enables screen operation via mouse.

Reference

- See "5.5. Operating the Controller via the Mouse" (Page 46) for more details on the use of this feature.

Syntax

object.**EnableRemoteMouseOperation** [= *enable*]

Data type

Boolean

Arguments

enable	Toggles mouse operation between enable and disable.	
	True	Enables remote mouse operation.
	False	Disables remote mouse operation.

Initial value

False

Related items

None

5.8.3. StartRemoteDesktop method

Function

Starts the Remote Desktop function.

The operating status will be reflected in RemoteDesktopStarted property (Page 49).

Point

- If Auto Update Mode has been specified, screen transfers will begin. The OnRemoteDesktopUpdated event (Page 56) is notified each time the screen update completes.
- No screen data is transferred if the Manual Update Mode is specified. Transfer the screen data by calling the UpdateRemoteDesktop method (Page 53).
- The remote desktop function cannot start if the VNC client is connected to the controller.

Reference

- See "5.2. Operating in Auto Update Mode" (Page 42) and "5.3. Operating in Manual Update Mode" (Page 43) for more details on the use of this feature.
- See "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

Syntax

object.**StartRemoteDesktop(ByVal** *mode* **As Short) As Integer**

Arguments

mode	Specify the operating mode for the Remote Desktop function. 0: Auto Update Mode 1: Manual Update Mode

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected.
1002	Illegal argument	Argument value is out of range.
1003	Illegal operation	The Remote Desktop is already running.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

OnRemoteDesktopUpdated event (Page 56), RemoteDesktopStarted property (Page 49)

5.8.4. StopRemoteDesktop method

Function

Stops the Remote Desktop function.

The operating status will be reflected in RemoteDesktopStarted property (Page 49).

Reference See "5.4. Stopping the Remote Desktop" (Page 45) for more details on the use of this property.

Syntax

*object.***StopRemoteDesktop() As Integer**

Arguments

None

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected.
1003	Illegal operation	The Remote Desktop is not running. A screen update triggered by the UpdateRemoteDesktop method is still in progress (waiting for the OnRemoteDesktopUpdated event).
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

RemoteDesktopStarted property (Page 49)

5.8.5. UpdateRemoteDesktop method

Function

Retrieves the monitor screen of the controller and updates the Remote Desktop screen.

The OnRemoteDesktopUpdated event (Page 56) is notified when the screen update completes.

Point This method functions only if Remote Desktop was started in Manual Update Mode by calling the StartRemoteDesktop method (Page 51).

Reference See "5.3. Operating in Manual Update Mode" (Page 43) for more details on the use of this feature.

Syntax

*object.***UpdateRemoteDesktop() As Integer**

Arguments

None

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected.
1003	Illegal operation	The Remote Desktop is not running. The Remote Desktop is running in Auto Update Mode. A screen update triggered by the UpdateRemoteDesktop method is still in progress (waiting for the OnRemoteDesktopUpdated event).
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

StartRemoteDesktop method (Page 51), OnRemoteDesktopUpdated event (Page 56)

5.8.6. CaptureRemoteDesktop method

Function

Saves the contents of the currently displayed Remote Desktop as a bitmap image file.
The file is saved as a 24-bit color bitmap file and the size is 1024 x 768 pixels.

Point

- This method can be executed regardless of the status of the connection with the controller, or the Remote Desktop.
- If a file already exists it will be overwritten (or an error will occur if the file is set to read-only).

Reference

See "5.6. Saving an Image of the Remote Desktop Screen to a File" (Page 47) for more details on the use of this property.

Syntax

object.**CaptureRemoteDesktop (ByVal** *dstFile* **As String) As Integer**

Arguments

dstFile	Specify the full path to the destination for the bitmap image file. Enter up to 255 characters. Specify "bmp" as the extension (the CVX Control does not check extensions). For characters that cannot be used in file names, see "Illegal Characters" (Page 127).

Return value

0	Processing succeeded	
1001	Processing failed	File save failed.
1002	Illegal argument	Illegal number of characters in path (NULL or exceeded maximum character count). Illegal file naming format.
1004	Out of space	Not enough disk space available.

Related items

None

5.8.7. ClearRemoteDesktop method

Function

Initializes the currently displayed Remote Desktop screen (to a black screen).

Point This method can be executed regardless of the status of the connection with the controller, or the Remote Desktop.

Reference See "5.7. Clearing the Remote Desktop Screen" (Page 48) for more details on the use of this property.

Syntax

*object.***ClearRemoteDesktop()**

Arguments

None

Return value

None

Related items

None

5.8.8. OnRemoteDesktopUpdated event

Function

This event occurs when the screen update of the Remote Desktop is completed.

Point

- In Auto Update Mode, this event is sent each time the controller screen is transferred after calling the StartRemoteDesktop method (Page 51).
- In Manual Update Mode, this event is sent when the screen transfer is completed as a result of calling the UpdateRemoteDesktop method (Page 53).

Reference

See "5.2. Operating in Auto Update Mode" (Page 42) and "5.3. Operating in Manual Update Mode" (Page 43) for more details on the use of this feature.

Syntax

Private Sub *object*_O**nRemoteDesktopUpdated(ByVal** *sender* **As Object, _**
ByVal *e* **As _**
AxCVXLib._DCVXEvents-OnRemoteDesktopUpdatedEvent) _
Handles *object.***OnRemoteDesktopUpdated**

Arguments: *e.state* **As Integer**

Arguments

state	Contains the processing result. See "Processing result" (next page) for details.

Processing result

0	Processing succeeded	
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

StartRemoteDesktop method (Page 51), UpdateRemoteDesktop method (Page 53)

Using the Remote Desktop

▶ **Receiving Results Data Output**

Receiving Image Data Output

Retrieving Archived Image Data

Retrieving Information of Setting Files

Sending/Receiving/Deleting Files

Issuing Non-Procedural Commands

Non-communication Functions

6 Receiving Results Data Output

This section explains how to receive data from the CV-X Series Controller over the communication link (results logging function).

- The logged data can be saved as a file on the PC.
- The results are retrieved as text and numerical data which can be used as feed back for display or control purposes.

Reference

- In order to use this function, "PC Program" in Output Settings must be set. For details on the Output Settings, see the related manuals.
- The communication link must be setup before you can use the functions in this chapter. Read "4. Common Settings for Communications" (Page 24) to initialize communication functions and connect to the controller.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

6.1. To Start Receiving Results Data Output

Related items

StartResultLog method (Page 65), OnResultLogDataReceived event (Page 69)

Procedures

Connect to the controller. (Refer to "4.3. Connecting to the Controller" (Page 26)).

(1) Notify the controller to start logging results.
Specify the operating mode (whether to save to a file) and the base folder these files should be saved in. Call the StartResultLog method to notify the controller to start logging results.

(2) From this point on, the controller outputs data as set in "PC Program" in the Output Settings.
The timing of the data outputs comes at the end of the program flow.

(3) The OnResultLogDataReceived event occurs each time data is received. The processing result and program setting number can be obtained from this event.

Point

- If the processing result is an error, results logging stops automatically.
- The file extension is set to "txt".
- If your objective is to obtain results data, you can specify not to save a file when specifying the operating mode in step (1).

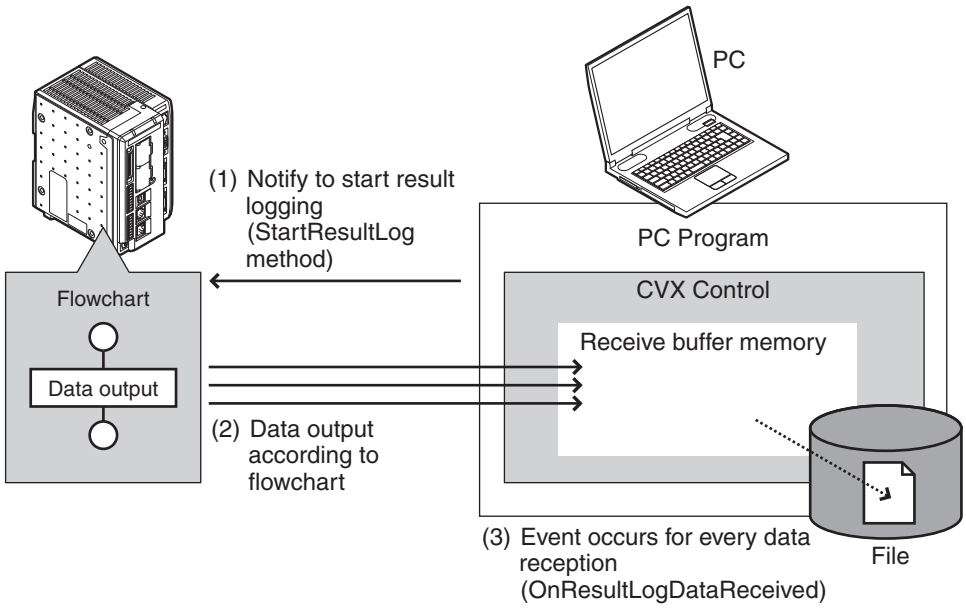


Figure 6-1. Results logging

Sample source code

The following code starts results logging when the Bn_StartResultLog button is clicked.
File saving is enabled and the base folder is set to "C:\work".
The program number and file name are displayed each time data is received.

```
Private Sub Bn_StartResultLog_Click(sender As System.Object, _
e As System.EventArgs) Handles Bn_StartResultLog.Click
  'Establish the connection with the controller beforehand
  Dim state As Integer                    'for storing the return value
  Dim baseFolder As String                'base path for storage file

  baseFolder = "C:\work"                  'start results logging
  state = CVX.StartResultLog(0, baseFolder)

  If state = 0 Then                       'processing if this method succeeds
       Call MsgBox("Success")
  ElseIf state = 1100 Then                'processing when a communication exception
                                          occurs
       Call MsgBox("Communication exception")
                                          'connection to the controller must be re-established
  Else
       Call MsgBox("Execution error")     'processing when an execution error occurs
       'branch based on error code as necessary
  End If
End Sub

Private Sub CVX_OnResultLogDataReceived(sender As System.Object, _
e As AxCVXLib._DCVXEvents_OnResultLogDataReceivedEvent) Handles _
CVX.OnResultLogDataReceived
  Dim strMsg As String                    'information on received data
  If e.state = 0 Then                     'processing if this method succeeds
       strMsg = " SD:" & e.driveNo & ", Program:"  & e.settingNo
       strMsg = strMsg & ", File:"  & e.dstFile
       Call MsgBox(strMsg)
  ElseIf e.state = 1100 Then              'processing when a communication exception
                                          occurs
       Call MsgBox("Communication exception")
                                          'connection to the controller must be re-established
  Else
       Call MsgBox("Execution error")     'processing when an execution error occurs
       'branch based on error code as necessary
  End If
End Sub
```

6.2. Retrieving Results Data Output

This section explains how to retrieve the most recently received results data.
Retrieving results data is a two-step process that involves specifying the target data and retrieving the output data items.

Related items

UpdateResultLogData method (Page 67), GetResultLogData method (Page 68)

Procedures

Start results logging. (Refer to "6.1. To Start Receiving Results Data Output" (Page 58)).

(1) Update data to be retrieved.
Calling the UpdateResultLogData method designates the most recent data, which was received by the results logging function, for retrieval.

(2) Retrieve the received data.
Specify the items in the Output Item Settings and call the GetResultLogData method to retrieve the corresponding items in the received data.

Point

- Data cannot be retrieved unless logging has been started.
- Repeated calls to the GetResultLogData method do not have the effect of updating data because the data to be retrieved must be set aside by calling the UpdateResultLogData method.
- To update retrieved data, the latest data must be set aside for retrieval by using the UpdateResultLogData method.

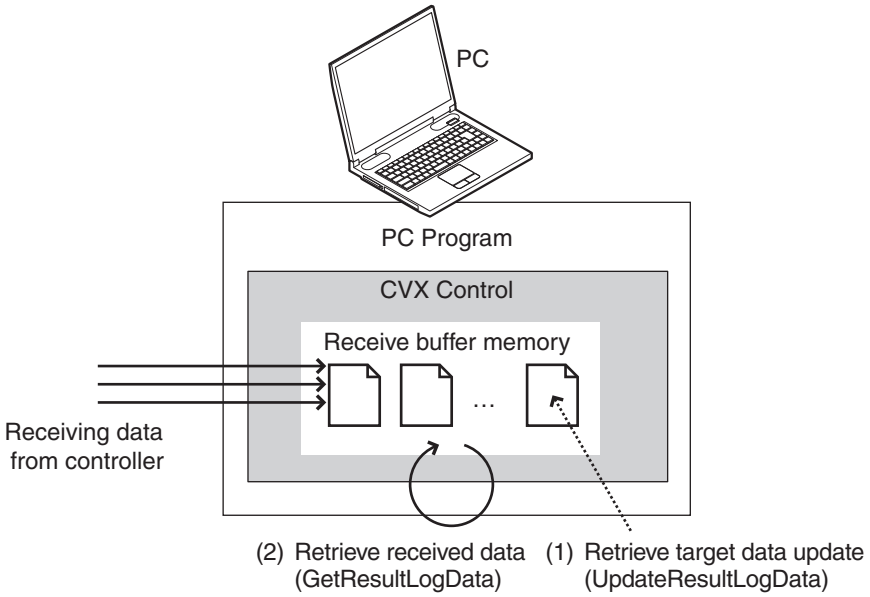
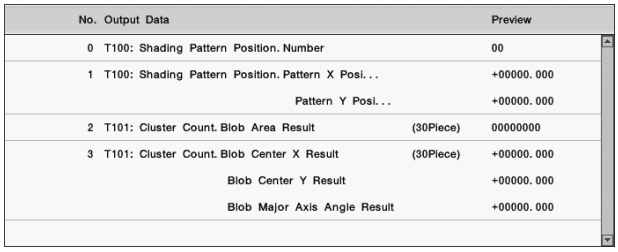


Figure 6-2. Retrieving results data output

The arguments for the GetResultLogData method, itemNo, arrayNo, and memberNo, must be set according to the output settings. This is illustrated below.

Assume that the output settings are set as follows.

Shading Pattern Position. Number
Shading Pattern Position. Pattern X Position/Y Position
Cluster Count. Blob Area Result
Cluster Count. Blob Center X Result/Center Y Result/Major Axis Angle Result



No.	Output Data		Preview
0	T100: Shading Pattern Position. Number		00
1	T100: Shading Pattern Position. Pattern X Posi. . .		+00000. 000
	Pattern Y Posi. . .		+00000. 000
2	T101: Cluster Count. Blob Area Result	(30Piece)	00000000
3	T101: Cluster Count. Blob Center X Result	(30Piece)	+00000. 000
	Blob Center Y Result		+00000. 000
	Blob Major Axis Angle Result		+00000. 000

Specify the arguments for the GetResultLogData method as shown below.

To retrieve the value of [Shading Pattern Position. Number], set

itemNo = 0	because it is the 0th output item;
arrayNo = 0	it is not an array;
memberNo = 0	and it is a scalar variable.

To retrieve the value of [Shading Pattern Position. Pattern Y Position], set

itemNo = 1	because it is the 1st output item;
arrayNo = 0	it is not an array;
memberNo = 1	and the desired member is the Y Position.

To retrieve the value of [Cluster Count. 5th Blob Area Result], set

itemNo = 2	because it is the 2nd output item;
arrayNo = 4	the desired element is the 5th element (start is 0);
memberNo = 0	and it is a scalar array variable.

To retrieve the value of [Cluster Count. 18th Blob Major Axis Angle Result], set

itemNo = 3	because it is the 3rd output item;
arrayNo = 17	the desired element is the 18th element (start is 0);
memberNo = 2	and the desired member is the Major Axis Angle.

Sample source code

The following code retrieves the latest data when the Bn_GetResultLogData button is clicked.
It is assumed the output settings has 4 items to output, each item being a circle array type variable (of 10 elements).

```
Private Sub Bn_GetResultLogData_Click(sender As System.Object, _
e As System.EventArgs) Handles Bn_GetResultLogData.Click
  'Start results logging beforehand
  Dim state As Integer                    'for storing the return value

  state = CVX.UpdateResultLogData()       'specify the data to retrieve
  If state <> 0 Then
       Call MsgBox("Data specification failed")
       Exit Sub                           'exit on failure
  End If

  'Retrieve the data
  Dim strError As String
  Dim index, arrayNo, memberNo As Integer
  Dim text(3,9,2) As String               'for data storage (text data)
  Dim data(3,9,2) As Double               'for data storage (numerical data)

  For index = 0 To 3                      'four output items
       For arrayNo = 0 To 9               '10 elements in each array
            For memberNo = 0 To 2
                                          'circles have 3 elements (CX, CY, CR)
                 state =CVX.GetResultLogData(index, arrayNo, _
                 memberNo,text(index,arrayNo,memberNo), _
                 data(index,arrayNo,memberNo))
                 If state <> 0 Then
                      strError = index & "," & arrayNo & "," & _
                      memberNo
                      Call MsgBox("Data retrieval failed:" & strError)
                 Exit Sub
                 End If
            Next memberNo
            Next arrayNo
       Next index
       Call MsgBox("Data retrieval succeeded")
End Sub
```

6.3. To Stop Receiving Results Data Output

Related items

StopResultLog method (Page 66), OnResultLogStopped event (Page 70)

Procedures

Start results logging. (Refer to "6.1. To Start Receiving Results Data Output" (Page 58)).

(1) Call the StopResultLog method.

(2) When results logging stops, the OnResultLogStopped event occurs.

Point If data is being received when the StopResultLog method is called, results logging will stop after one measurement count's worth of data is received.

Sample source code

The following code stops results logging when the Bn_StopResultLog button is clicked.

```
Private Sub Bn_StopResultLog_Click(sender As System.Object, _
e As System.EventArgs) Handles Bn_StopResultLog.Click
  'This code assumes the remote desktop has already been started
  Dim state As Integer                    'for storing the return value
        state = CVX.StopResultLog()       'stop results logging

  If state = 0 Then                       'processing if this method succeeds
        Call MsgBox("Success")
  ElseIf state = 1100 Then                'processing when a communication exception
                                          occurs
        Call MsgBox("Communication exception")
                                          'connection to the controller must be re-
                                          established
  Else
        Call MsgBox("Execution error")    'processing when an execution error occurs
        'branch based on error code as necessary
  End If
End Sub

Private Sub CVX_OnResultLogStopped(sender As System.Object, _
e As AxCVXLib._DCVXEvents_OnResultLogStoppedEvent) Handles _
CVX.OnResultLogStopped
  Dim strStop As String
        strStop = "Results logging stopped: "
  If e.state = 0 Then                     'processing if this method succeeds
        Call MsgBox(strStop & "Success")
  Else                                    'processing when a communication exception
                                          occurs
        Call MsgBox(strStop & "Communication exception")
                                          'connection to the controller must be re-
                                          established
  End If
End Sub
```

6.4. API Reference

6.4.1. ResultLogStarted property

Function

Indicates the operating status of the results logging function (read-only).

This property is updated when the StartResultLog method (Page 65) and OnResultLogDataReceived event (Page 69) are called to start and stop result logging.

Point Even if results logging stops automatically due to a processing error in the OnResultLogDataReceived event, this property will be updated.

Syntax

object.**ResultLogStarted**

Data type

Boolean

Arguments

True	Results logging has started.
False	Results logging has stopped.

Initial value

False

Related items

StartResultLog method (Page 65), OnResultLogDataReceived event (Page 69)

6.4.2. StartResultLog method

Function

Starts the result logging function.

The operating status will be reflected in ResultLogStarted property (Page 64).

The OnResultLogDataReceived event (Page 69) will notify you when results data output has been received.

Point The received data can be saved to a file depending on how the data is to be used.

Reference
- Refer to "6.1. To Start Receiving Results Data Output" (Page 58) for more details on the use of this feature.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

Syntax

object.**StartResultLog (ByVal** *mode* **As Short, ByVal** *folder* **As String) As Integer**

Arguments

mode	Specify whether to save the received data to a file. 0: save received data to a file 1: do not save received data to a file
folder	Specify the full path to the destination for the received data file. *This is only referenced* if *mode* argument is set to save the data as a file. Folder SD2 is created under this base folder, and the file is created according to the file naming rule. Enter up to 255 characters. For those characters that cannot be used in folder names, see "Illegal Characters" (Page 127).

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected. The controller is already logging results to another PC Program.
1002	Illegal argument	Illegal number of characters in folder name (NULL or exceeded maximum character count). Illegal folder naming format.
1003	Illegal operation	Results logging is already in progress.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

OnResultLogDataReceived event (Page 69), ResultLogStarted property (Page 64)

6.4.3. StopResultLog method

Function

Stops the result logging function.

This method only posts a request. The OnResultLogStopped event (Page 70) notifies when result logging has stopped.

Reference Refer to "6.3. To Stop Receiving Results Data Output" (Page 63) for more details on the use of this feature.

Syntax

*object.***StopResultLog() As Integer**

Arguments

None

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected.
1003	Illegal operation	Results logging has not been started.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

When results logging stops, the OnResultLogStopped event (Page 70) occurs.

6.4.4. UpdateResultLogData method

Function

Sets aside the most recent data that will be the target when GetResultLogData method calls for data retrieval.

Reference Refer to "6.2. Retrieving Results Data Output" (Page 60) for more details on the use of this feature.

Syntax

*object.***UpdateResultLogData As Integer**

Arguments

None

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized.
1003	Illegal operation	Logging has not been started.
1009	No target data	Target data not found.

Related items

GetResultLogData method (Page 68)

6.4.5. GetResultLogData method

Function

Retrieves the most recently received results data.

This method retrieves data as text strings and numerical data specified in the Output Item Settings by specifying four indexes.

Point The target data for retrieving must be designated beforehand with the UpdateResultLogData method.

Reference Refer to "6.2. Retrieving Results Data Output" (Page 60) for more details on the use of this feature.

Syntax

*object.***GetResultLogData(ByVal** *itemNo* **As Integer, _**
ByVal *arrayNo* **As Integer, ByVal** *memberNo* **As Integer, _**
ByRef *text* **As String, ByRef** *data* **As Double) As Integer**

Arguments

itemNo	Specify the output item number specified in output settings (0 to 255).
arrayNo	Specify the array number in the output items. (Specify the position from the first array starting at 0.) Specify 0 if no arrays were set in the output items.
memberNo	Specify the member number of the output item (0 to 5). Specify 0 when there is no member.

	Ouput item					
Member number	**Date**	**Position**	**Position/ Angle**	**Plane type information**	**Robot coordinates**	
					XYZRz	**XYZRxRyRz**
0	Year	X	X	X inclination	X	X
1	Month	Y	Y	Y inclination	Y	Y
2	Day	-	Angle	Z slices	Z	Z
3	Hour	-	-	-	Rz	Rx
4	Minute	-	-	-	-	Ry
5	Second	-	-	-	-	Rz

text	Contains a string of up to 255 characters formatted as specified in the output settings. Contains null (empty) if the method fails.
data	Contains the numeric data in the output. Set with -1 if the method fails or if the output contains only text string data.

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized.
1002	Illegal argument	Number of output item, array, or member is out of range.
1003	Illegal operation	Logging has not been started. The target data for retrieval has not been specified. Specify it with the UpdateResultLogData.
1009	No target data	No target data

Related items

UpdateResultLogData method (Page 67)

6.4.6. OnResultLogDataReceived event

Function

Occurs each time the control receives data from the controller.

The event notifies the program setting number and destination file name of the received data.

Point If the processing result in this event is an error, results logging stops automatically.

Reference Refer to "6.1. To Start Receiving Results Data Output" (Page 58) for more details on the use of this feature.

Syntax

Private Sub *object*_**OnResultLogDataReceived (ByVal** *sender* **As Object, _**
ByVal *e* **As AxCVXLib._DCVXEvents_OnResultLogDataReceivedEvent) _**
Handles *object.***OnResultLogDataReceived**

Arguments: *e.state* **As Integer,** *e.driveNo* **As Integer,** *e.settingNo* **As Integer,** *e.dstFile* **As String**

Arguments

state	Contains the processing result. See "Processing result" for details.
driveNo	Contains the SD card number. Set with -1 if retrieval fails. 1: SD1 2: SD2
settingNo	Contains the current program setting number (0 to 999). Set with -1 if retrieval fails.
dstFile	Contains the full path to the destination file in up to 255 characters. Contains null (empty) if the method fails.

Processing result

0	Processing succeeded	
1001	Processing failed	Access to the destination file failed.
1004	Out of space	Not enough memory space at destination.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

StartResultLog method (Page 65)

6.4.7. OnResultLogStopped event

Function

Occurs when the results logging stops as a result of the StopResultLog method.

Reference Refer to "6.3. To Stop Receiving Results Data Output" (Page 63) for more details on the use of this feature.

Syntax

Private Sub *object*_**OnResultLogStopped (ByVal** *sender* **As Object, _**
ByVal *e* **As AxCVXLib._DCVXEvents_OnResultLogStoppedEvent) _**
Handles *object*.**OnResultLogStopped**

Arguments: *e.state* **As Integer**

Arguments

state	Contains the processing result.	See "Processing result" for details.

Processing result

0	Processing succeeded	
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

StopResultLog method (Page 66), ResultLogStarted property (Page 64)

Using the Remote Desktop

Receiving Results Data Output

▶ **Receiving Image Data Output**

Retrieving Archived Image Data

Retrieving Information of Setting Files

Sending/Receiving/Deleting Files

Issuing Non-Procedural Commands

Non-communication Functions

7. Receiving Image Data Output

This section explains how to receive data from the CV-X Series Controller over the communication link (image logging function). The logged image data can be saved as a file on the PC.

Point

- In order to use this function, the output device setting for the Image output settings must be set to "PC Program". For details on the Image output settings, see the related manuals.
- The communication link must be setup before you can use the functions in this chapter. Read "4. Common Settings for Communications" (Page 24) to initialize communication functions and connect to the controller.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

7.1. To Start Receiving Image Output Data

Related items

StartImageLog method (Page 76), OnImageLogDataReceived event (Page 78)

Procedures

Connect to the controller. (Refer to "4.3. Connecting to the Controller" (Page 26)).

(1) Notify the controller to start logging images.
Specify the base folder where these files should be saved in. Call the StartImageLog method to notify the controller to start logging images.

(2) From this point on, the controller outputs image data.
The timing of the data outputs is at the end of the flow.

(3) The OnImageLogDataReceived event occurs each time data is received from the controller. The processing result and program setting number can be obtained from this event.

Point

- If the processing result is an error, image logging stops automatically.
- The received image data will be saved in a file according to the format specified in the Image output settings.
- The file name is determined by the file naming rule defined in the Image output settings.
- The file extension is determined by the image file saving format defined in the Image output settings.

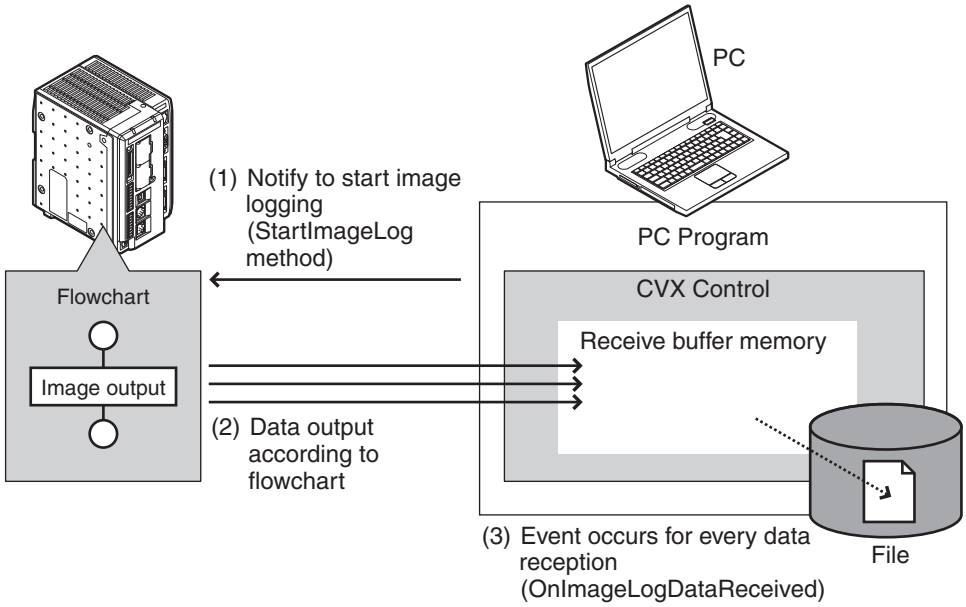


Figure 7-1. Image logging

Sample source code

The following code starts image logging when the Bn_StartImageLog button is clicked.
The base folder is set to "C:\work".
The program number and file name are displayed each time data is received.

```
Private Sub Bn_StartImageLog_Click(sender As System.Object, _
e As System.EventArgs) Handles Bn_StartImageLog.Click
  'Establish the connection with the controller beforehand
  Dim state As Integer                    'for storing the return value
  Dim baseFolder As String                'base path for storage file

  baseFolder = "C:\work"                  'start image logging
  state = CVX.StartImageLog(baseFolder)

  If state = 0 Then                       'processing if this method succeeds
       Call MsgBox("Success")
  ElseIf state = 1100 Then                'processing when a communication exception
                                          occurs
       Call MsgBox("Communication exception")
                                          'connection to the controller must be re-
                                          established
  Else
       Call MsgBox("Execution error")     'processing when an execution error occurs
       'branch based on error code as necessary
  End If
End Sub

Private Sub CVX_OnImageLogDataReceived(sender As System.Object, _
e As AxCVXLib._DCVXEvents_OnImageLogDataReceivedEvent) Handles _
CVX.OnImageLogDataReceived
  Dim strMsg As String                    'information on received data
  If e.state = 0 Then                     'processing if this method succeeds
       strMsg = " SD:" & e.driveNo & ", Program:" & e.settingNo
       strMsg = strMsg & ", Archive Condition:" & e.conditionType & _
       ", Total Count:" & e.prcCount
       Call MsgBox(strMsg)
  ElseIf e.state = 1100 Then              'processing when a communication exception
                                          occurs
       Call MsgBox("Communication exception")
                                          'connection to the controller must be re-
                                          established
  Else
       Call MsgBox("Execution error")     'processing when an execution error occurs
       'branch based on error code as necessary
  End If
End Sub
```

7.2. To Stop Receiving Image Output Data

Related items

StopImageLog method (Page 77), OnImageLogStopped event (Page 79)

Procedures

Start image logging. (Refer toe "7.1. To Start Receiving Image Output Data" (Page 72)).

(1) Call the StopImageLog method.

(2) When image logging stops, the OnImageLogStopped event occurs.

Point If data is being received when the StopImageLog method is called, image logging will stop after one measurement count's worth of data is received.

Sample source code

The following code stops image logging when the Bn_StopImageLog button is clicked.

```
Private Sub Bn_StopImageLog_Click(sender As System.Object, _
e As System.EventArgs) Handles Bn_StopImageLog.Click
  'This code assumes image logging is already in progress
  Dim state As Integer                    'for storing the return value
  state = CVX.StopImageLog()              'stop image logging

  If state = 0 Then                       'processing if this method succeeds
       Call MsgBox("Success")
  ElseIf state = 1100 Then                'processing when a communication exception
                                          occurs
       Call MsgBox("Communication exception")
                                          'connection to the controller must be re-
                                          established
  Else
       Call MsgBox("Execution error")     'processing when an execution error occurs
       'branch based on error code as necessary
  End If
End Sub

Private Sub CVX_OnImageLogStopped(sender As System.Object, _
e As AxCVXLib._DCVXEvents_OnImageLogStoppedEvent) Handles _
CVX.OnImageLogStopped
  Dim strStop As String
  strStop = "Image logging stopped: "
  If e.state = 0 Then                     'processing if this method succeeds
       Call MsgBox(strStop & "Success")
  Else                                    'processing when a communication exception
                                          occurs
       Call MsgBox(strStop & "Communication exception")
                                          'connection to the controller must be re-
                                          established
  End If
End Sub
```

7.3. API Reference

7.3.1. ImageLogStarted property

Function

Indicates the operating status of the image logging function (read-only).

This property is updated when the StartImageLog method (Page 76) and OnImageLogStopped event (Page 79) are called to start and stop image logging.

Point Even if image logging stops automatically due to a processing error in the OnImageLogDataReceived event, this property will be updated.

Syntax

*object.***ImageLogStarted**

Data type

Boolean

Arguments

True	Image logging has started.
False	Image logging has stopped.

Initial value

False

Related items

StartImageLog method (Page 76), OnImageLogStopped event (Page 79)

7.3.2. StartImageLog method

Function

Starts the image logging function.

The operating status will be reflected in ImageLogStarted property (Page 75).

The OnImageLogDataReceived event (Page 78) will notify you when image data output has been received.

Reference

- Refer to "7.1. To Start Receiving Image Output Data" (Page 72) for more details on the use of this feature.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

Syntax

object.**StartImageLog (ByVal** *folder* **As String) As Integer**

Arguments

folder	Specify the full path of the destination for the received data file. Folder SD2 is created under this base folder, and the file is created according to the file naming rule. Enter up to 255 characters. For those characters that cannot be used in folder names, see "Illegal Characters" (Page 127).

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected. The controller is already logging images to another PC Program.
1002	Illegal argument	Illegal number of characters in file name (NULL or exceeded maximum character count). Illegal folder naming format.
1003	Illegal operation	Image logging is already in progress.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

OnImageLogDataReceived event (Page 78), ImageLogStarted property (Page 75)

7.3.3. StopImageLog method

Function

Stops the image logging function.

This method only posts a request. The OnImageLogStopped event (Page 79) notifies when image logging has stopped.

Reference Refer to "7.2. To Stop Receiving Image Output Data" (Page 74) for more details on the use of this feature.

Syntax

*object.***StopImageLog() As Integer**

Arguments

None

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected.
1003	Illegal operation	Image logging has not been started.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

When image logging stops, the OnImageLogStopped event (Page 79) occurs.

7.3.4. OnImageLogDataReceived event

Function

Occurs each time the control receives image output data from the controller.

The event notifies the program setting number and destination file name of the received data.

Point If the processing result in this event is an error, image logging stops automatically.

Reference Refer to "7.1. To Start Receiving Image Output Data" (Page 72) for more details on the use of this feature.

Syntax

Private Sub *object*_**OnImageLogDataReceived (ByVal** *sender* **As Object, _**
ByVal *e* **As AxCVXLib._DCVXEvents_OnImageLogDataReceivedEvent) _**
Handles *object*.**OnImageLogDataReceived**

Arguments: *e.state* **As Integer,** *e.driveNo* **As Integer,** *e.settingNo* **As Integer,** *e.conditionType* **As Integer**, *e.prcCount* **As Integer**

Arguments

state	Contains the processing result. See "Processing result" for details.
driveNo	Contains the SD card number. Set with -1 if retrieval fails. 1:SD1 2: SD2
settingNo	Contains the current program setting number (0 to 999). Set with -1 if retrieval fails.
conditionType	Specify the Archive condition number 0 or 1. 0: Latest Archive 1: Total NG Archive
prcCount	Contains the measurement count that was output. Set with -1 if retrieval fails.

Processing result

0	Processing succeeded	
1001	Processing failed	Access to the destination file failed.
1004	Out of space	Not enough memory space at destination.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

StartImageLog method (Page 76)

7.3.5. OnImageLogStopped event

Function

Occurs when image logging stops as a result of the StopImageLog method (Page 77).

Reference Refer to "7.2. To Stop Receiving Image Output Data" (Page 74) for more details on the use of this feature.

Syntax

Private Sub *object*_**OnImageLogStopped (ByVal** sender **As Object, _**
ByVal *e* **As AxCVXLib._DCVXEvents_OnImageLogStoppedEvent) _**
Handles *object*.**OnImageLogStopped**

Arguments: *e*.**state As Integer**

Arguments

state	Contains the processing result. See "Processing result" for details.

Processing result

0	Processing succeeded	
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

StopImageLog method (Page 77), ImageLogStarted property (Page 75)

Using the Remote Desktop

Receiving Results Data Output

Receiving Image Data Output

▶ **Retrieving Archived Image Data**

Retrieving Information of Setting Files

Sending/Receiving/Deleting Files

Issuing Non-Procedural Commands

Non-communication Functions

8. Retrieving Archived Image Data

Users can retrieve archived image data stored in the CV-X Series controller (archived image data retrieval function).

- This section explains how to retrieve a list of information on archived image data stored in the controller.
- Users can retrieve stored image data and save them to a file.
- This function allows developers to create an application that retrieves and displays the information list of archived image data currently stored in the controller. Users can select the desired measurement count and retrieve the corresponding data.

Point

- In order to use this function, users must understand the archive function.
  For details on the archive function, see the related manuals.
- The communication link must be setup before you can use the functions in this chapter.
  Read "4. Common Settings for Communications" (Page 24) to initialize communication functions and connect to the controller.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

8.1. Retrieving History Lists

This section explains how to retrieve a list of information on archived image data stored in the controller. Each item in the information list can be used to obtain information on the total count of inspections archived, archive timestamp, status of outputs to the SD card, and output status over a communication link.

Retrieving history information is a two-step process that involves transferring an information list and retrieving data for each item in the list.

Related items

GetArchivedImageList method (Page 86), GetArchivedImageInfo method (Page 87)

Procedures

Connect to the controller. (Refer to "4.3. Connecting to the Controller" (Page 26)).

(1) Transfer the information list for archived image data.
Calling the GetArchivedImageList method transfers the information list for the archived image data stored in the controller's internal memory and obtains the number of items in the list.

(2) Call the GetArchivedImageInfo method to retrieve data for each item in the list transferred in (1).
Specify the index for the list items and call the number of list items in a loop.

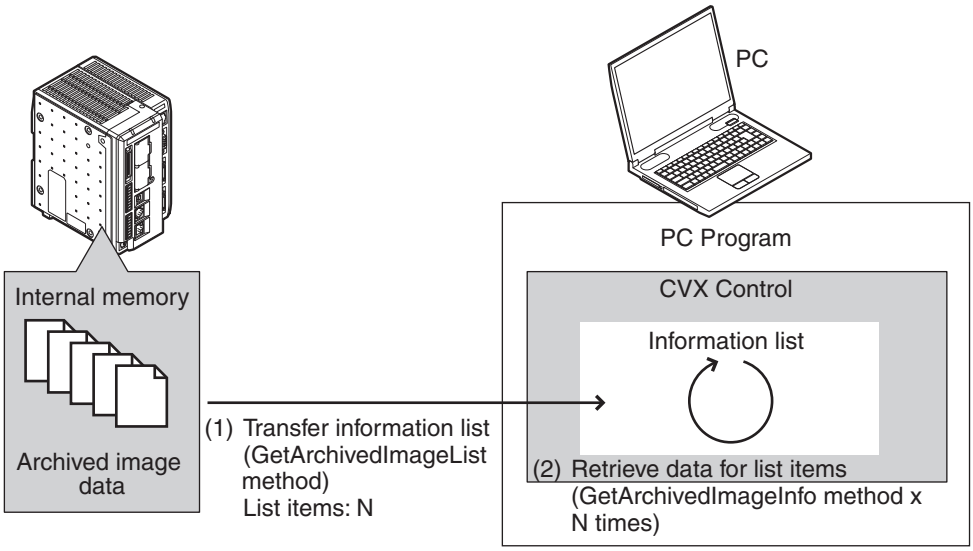


Figure 9-1. Retrieving the information list for archived image data

Sample source code

The following code retrieves the information list of archived image data when the Bn_GetList button click event occurs. The process is aborted if an error occurs.

```
Private Sub Bn_GetList_Click(sender As System.Object, _
e As System.EventArgs) Handles Bn_GetList.Click
  'Establish the connection with the controller beforehand
  Dim state As Integer                    'for storing the return value
  Dim listNum As Integer                  'for storing the number of lists

  'Transfer the information list
  listNum = 0
  state = CVX.GetArchivedImageList(0, listNum)

  If state = 0 Then                       'processing if this method succeeds
       Call MsgBox("Success")
  ElseIf state = 1100 Then                'processing when a communication exception
                                          occurs
       Call MsgBox("Communication exception")
                                          'connection to the controller must be re-
                                          established
       Exit Sub
  Else
       Call MsgBox("Execution error")     'processing when an execution error occurs
       Exit Sub
       'branch based on error code as necessary
  End If

  If listNum < 1 Then                     'exit if there are no list items
       Exit Sub
  End If

  'Retrieve each item in the transferred list
  Dim index As Integer                    'index for list items
  Dim count As Integer                    'archived image data info: measurement count
  Dim dateTime As Date                    'archived image data info: date and time data
                                          was stored
  Dim isCard As Integer                   'archived image data info: output status to SD
                                          card
  Dim isComm As Integer                   'archived image data info: output status to
                                          communication port

  For index = 0 To listNum - 1
      state = CVX.GetArchivedImageInfo(index, count, dateTime, _
      isCard, isComm)
      If state = 0 Then
             'processing if successful
             'i.e.: save retrieved data, reflect controls for display, etc.
      Else
             'processing if execution error occurs
             'branch based on error code as necessary
      End If
  Next index
End Sub
```

8.2. Retrieving Archived Image Data

This section explains how to transfer the list of information on archived image data stored in the controller and save it to a file.
Data transfer starts when the method is called, and an event is notified when the transfer is completed.

Related items

GetArchivedImage method (Page 88), OnArchivedImageTransferred event (Page 90)

Procedures

Connect to the controller. (Refer to "4.3. Connecting to the Controller" (Page 26)).

(1) Start transferring image data.
Call the GetArchivedImage method. The data transfer will start.

(2) When the data transfer is complete, the OnArchivedImageTransferred event occurs.
The processing results and type of data transferred is notified through an event.

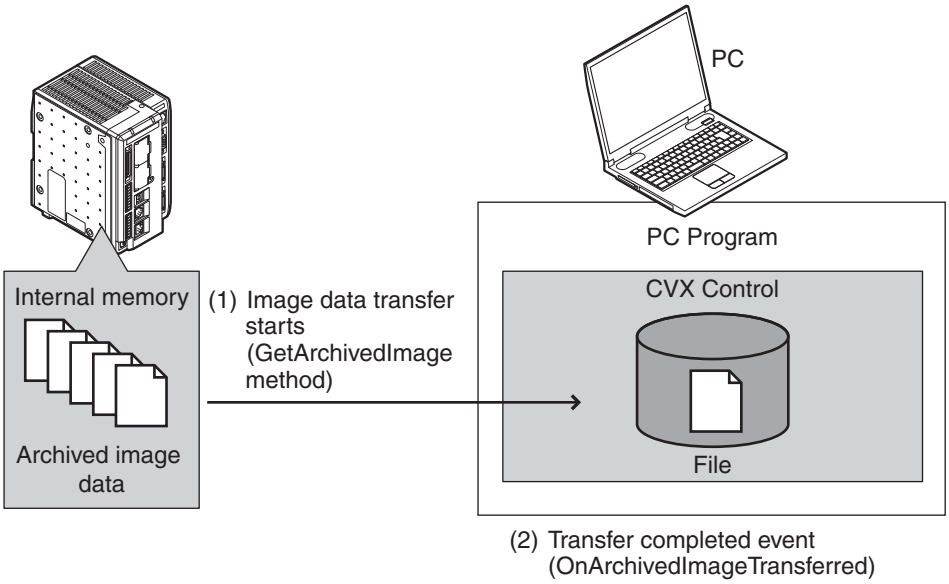


Figure 9-2. Transferring archived image data (image and results data)

Sample source code

The following code starts transferring archived image data when the Bn_GetHistImage button is clicked.
The image data is transferred for archive condition 1, measurement count 50, camera 1 image (in bitmap form) at actual size.
The file name at the destination (PC) is "history_image.bmp".

When the data transfer is complete, a message box appears showing the results of the transfer.

```
Private Sub Bn_GetHistImage_Click(sender As System.Object, e As _
System.EventArgs) Handles Bn_GetHistImage.Click
  'Establish the connection with the controller beforehand
  Dim state As Integer                    'for storing the return value
  Dim dstFile As String                   destination file name

                                          'start transferring archived image data
  dstFile = "C:\cv-x\history_image.bmp"
  state = CVX.GetArchivedImage(1, 50, 1, 0, 0, dstFile, 1)

  If state = 0 Then                       'processing if this method succeeds
       Call MsgBox("Success")
  ElseIf state = 1100 Then                'processing when a communication exception
                                          occurs
       Call MsgBox("Communication exception")
                                          'connection to the controller must be re-
                                          established
  Else
       Call MsgBox("Execution error")     'processing when an execution error occurs
       'branch based on error code as necessary
  End If
End Sub

Private Sub CVX_OnArchivedImageTransferred(sender As System.Object, _
e As AxCVXLib._DCVXEvents_OnArchivedImageTransferredEvent) Handles _
CVX.OnArchivedImageTransferred
  If e.state = 0 Then                     'processing if this method succeeds
       Call MsgBox("Success")
  ElseIf e.state = 1100 Then              'communication exception processing
       Call MsgBox("Communication exception")
                                          'connection to the controller must be re-
                                          established
  Else
       Call MsgBox("Execution error")
                                          'processing when an execution error occurs
       'branch based on error code as necessary
  End If
End Sub
```

8.3. API Reference

8.3.1. ArchivedImageTransferBusy property

Function

Indicates the operating status of the archived image data retrieval function (GetArchivedImage method (Page 88)) (read-only).
This property is updated when the GetArchivedImage method is called, and when the OnArchivedImageTransferred event (Page 90) is notified.

Syntax

*object.***ArchivedImageTransferBusy**

Data type

Boolean

Arguments

True	archived image data transfer in progress.
False	archived image data transfer not in progress.

Initial value

False

Related items

GetArchivedImage method (Page 88), OnArchivedImageTransferred event (Page 90)

8.3.2. GetArchivedImageList method

Function

Retrieves the number of information lists available for archived image data stored in the controller for the specified archive conditions.

Point

- Use GetArchivedImageInfo method (Page 87) to retrieve information about each list item.
- If the controller is not in online Run mode, or the controller or another PC Program is accessing the archived image data, an error will occur.

Reference

- Refer to "8.1. Retrieving History Lists" (Page 81) for more details on the use of this feature.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

Syntax

*object.***GetArchivedImageList(ByVal** *conditionNo* **As Integer, ByRef** *num* **As Integer) As Integer**

Arguments

conditionNo	Specify the Archive condition number 0 or 1. 0: Latest Archive 1: Total NG Archive
num	Contains the number of archived image data information lists that were retrieved. Set with -1 if retrieval fails.

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected. Another PC program (CVX Control) is accessing the archived image data.
1003	Illegal operation	The controller is not in Online Run mode. Retrieving archived image data (GetArchivedImage).
1006	Processing in progress	The controller is accessing archived image data. (Image Archive dialog, Statistics dialog, Non-procedural command)
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

GetArchivedImageInfo method (Page 87)

8.3.3. GetArchivedImageInfo method

Function

Retrieves the data for each list item (measurement count) from the archived image data information list obtained with the GetArchivedImageList method (Page 86).

Reference Refer to "8.1. Retrieving History Lists" (Page 81) for more details on the use of this feature.

Syntax

*object.***GetArchivedImageInfo(ByVal** *index* **As Integer, ByRef** *prcCount* **As Integer, _**
ByRef *date* **As Date, ByRef** *isSavedDrive* **As Integer, _**
ByRef *isSavedComm* **As Integer) As Integer**

Arguments

index	Specify the list number (0 to (lists retrieved - 1)). "Lists retrieved" is the value returned by the previous call to the GetArchivedImageList method.
prcCount	Contains the measurement count. Set with -1 if retrieval fails.
date	Contains the date and time the data was stored. Set with 01-01-1980 00:00:00 if retrieval fails.
isSavedDrive	Contains output state to the controller's SD card or USB HDD. This indicates whether the data has been saved by the Archived Image Data Output function, Image Archive viewer, the HS non-procedural command, or the StoreHistory method. Set with -1 if retrieval fails. 0: Not output 1: Output completed
isSavedComm	Contains the state of output over the communication link. This indicates whether the data has been saved by the Archived Image Data Output function, GetArchivedImage method. Set with -1 if retrieval fails. 0: Not output 1: Output completed

Return value

0	Processing succeeded	
1002	Illegal argument	List number is out of range.
1003	Illegal operation	All GetArchivedImageList methods have failed.

Related items

GetArchivedImageList method (Page 86)

8.3.4. GetArchivedImage method

Function

Retrieves image data stored in the controller and saves it to a file in the user-selected file format of JPEG or bitmap (with compression ratio).
This method only posts a request. An event is notified when the operation is completed.
The operating status will be reflected in ArchivedImageTransferBusy property (Page 85).

Point

- If the file already exists at the destination it will be overwritten.
- If the controller is not in Online Run mode, or the controller or another PC Program is accessing the archived image data, an error will occur.

Reference

- Refer to "8.2. Retrieving Archived Image Data" (Page 83) for more details on the use of this feature.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.
- The LumiTrax mode is supported on controller firmware versions 3.1.0000 or later.
- The multi-spectral mode is supported on controller firmware versions 4.1.0000 or later.

Syntax

object.**GetArchivedImage(ByVal** *conditionNo* **As Integer, ByVal** *prcCount* **As Integer, ByVal** *cameraNo* **As Integer, ByVal** *captureNo* **As Integer, ByVal** *lightingNo* **As Integer, ByVal** *dstFile* **As String, ByVal** *format* **As Integer) As Integer**

Arguments

conditionNo	Specify the Archive condition number 0 or 1. 0: Latest Archive 1: Total NG Archive
prcCount	Specify the measurement count (1 or more).
cameraNo	Specify the camera No. (1 to 4).
captureNo	Specify the capture No. (1 to 8) when the Multi-Capture is enabled. Specify 0 when the Multi-Capture is disabled.
lightingNo	Specify the lighting number(1 or 2) when the Program Setting is Connector type. 1: Lighting A 2: Lighting B Specifies the LumiTrax image type (10 to 180) when the LumiTrax mode is in use. 10: Normal Image 20: Shape Image 1 30: Shape Image 2 40: Shape Image 3 50: Texture Image 60: Gradient X Image 70: Gradient Y Image 80: ↓Image 90: ←Image 100: ↑Image 110: →Image 120: ↙Image 130 ↖Image 140: ↗Image 150: ↘Image 160: Tracking Image 1 170: Tracking Image 2 180: Ambient Light Image Specifies the image type (190 to 200) when the camera model is LJ-V. 190: Image Height 200: Gray Image Specifies the multi-spectral image type (210 to 310) when the multi-spectral mode is in use. 210: Average Gray Image 220: Color Image

	230: Color-Difference Image 240: UV Image 250: B Image 260: G Image 270: AM Image 280: R Image 290: FR Image 300: IR Image 310: W Image Specify 0 when the Program Setting is not Connector type, when the camera model is not LJ-V, or when the LumiTrax mode or the multi-spectral mode is not in use.
dstFile	Specify the full path of the file name at the destination (on the PC). The file is created with the specified name regardless of the file format. Add the "bmp" or "jpg" extension based on the file format specified in image archive settings (the CVX Control does not check extensions). Enter up to 255 characters. For characters that cannot be used in file names, refer to "Illegal Characters" (Page 127).
format	Specify the format of the image file. The compression ratio is the multiplier applied to the vertical and horizontal length. 0: JPEG format 1: BMP format (compression ratio: actual size, file size 1x) 2: BMP format (compression ratio: 1/2 times, file size 1/4x) 3: BMP format (compression ratio: 1/4 times, file size 1/16x)

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected. Another PC program (CVX Control) is accessing the archived image data. The destination drive is read-only.
1002	Illegal argument	Archive condition number or measurement count or camera number is out of range. Illegal number of characters in file name (NULL or exceeded maximum character count). Illegal file naming format.
1003	Illegal operation	The controller is not in Online Run mode. Retrieving archived image data (includes other CVX Controls) (GetArchivedImage).
1006	Processing in progress	The controller is accessing archived image data. (Image Archive dialog, Statistics dialog, Non-procedural command)
1009	No target data	No archived image data found.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

When processing is completed, the OnArchivedImageTransferred event (Page 90) occurs.

Event results

0	Processing succeeded	
1001	Processing failed	Access to the destination file failed. Not enough memory space at destination.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

8.3.5. OnArchivedImageTransferred event

Function

Occurs when archived image data transfer {GetArchivedImage method (Page 88)} is completed.

Point

This event is common for multiple methods. To distinguish which method the event refers to, check the arguments.

Reference

See "8.2. Retrieving Archived Image Data" (Page 83) for more details on the use of this feature.

Syntax

Private Sub *object*_**OnArchivedImageTransferred(ByVal** *sender* **As Object, _**
ByVal *e* **As AxCVXLib._DCVXEvents_OnArchivedImageTransferredEvent) _**
Handles *object*.**OnArchivedImageTransferred**

Arguments: *e.state* **As Integer**

Arguments

state	Contains the processing result. Refer to "Processing result" for details.

Processing result

0	Processing succeeded

For details on errors, see the [Event results] for each method.

Related items

GetArchivedImage method (Page 88), ArchivedImageTransferBusy property (Page 85)

Using the Remote Desktop

Receiving Results Data Output

Receiving Image Data Output

Retrieving Archived Image Data

▶ **Retrieving Information of Setting Files**

Sending/Receiving/Deleting Files

Issuing Non-Procedural Commands

Non-communication Functions

9. Retrieving Information of Setting Files

This chapter explains how to retrieve information (name, etc.) of the settings files including program settings and library settings. (retrieval of setting file information)

Point

- In order to use the functions in this chapter, users must understand exactly what settings files are.
  For details on the various settings files, refer to the related manuals.
- The communication link must be setup before you can use the functions in this chapter.
  Read "4. Common Settings for Communications" (Page 24) to initialize communication functions and connect to the controller.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.
- When uploading files, the firmware version of the receiving controller must be compatible with the version of the settings data file, otherwise the transfer will fail.

9.1. Retrieve Information of Program Setting (Library Setting) Files

This section explains how to retrieve information of program setting (library setting) files from the controller's SD card via communication link.

9.1.1. Retrieving Information Lists

This section explains how to retrieve an information list of the program setting files on the SD card in the controller.

- The list provides information on the SD card number, program setting number, and program setting name.
- The data is obtained through a two-step process, transfer of the information list and retrieval of data for each item in the list.

Related items

GetInspectList method (Page 94), GetInspectInfo method (Page 95), GetDictionaryList method (Page 96) and GetDictionaryInfo method (Page 97).

Procedures

Connect to the controller. (Refer to "4.3. Connecting to the Controller" (Page 26)).

(1) Transfer the information list for program settings.
Calling the GetInspectList method transfers the list for the program setting files stored on the SD card in the controller and allows retrieval of the number of items in the list.

(2) Call the GetInspectInfo method to retrieve the data for each item in the list transferred in (1).
Specify the index of the list item and call this method in a loop for the number of items in the list.

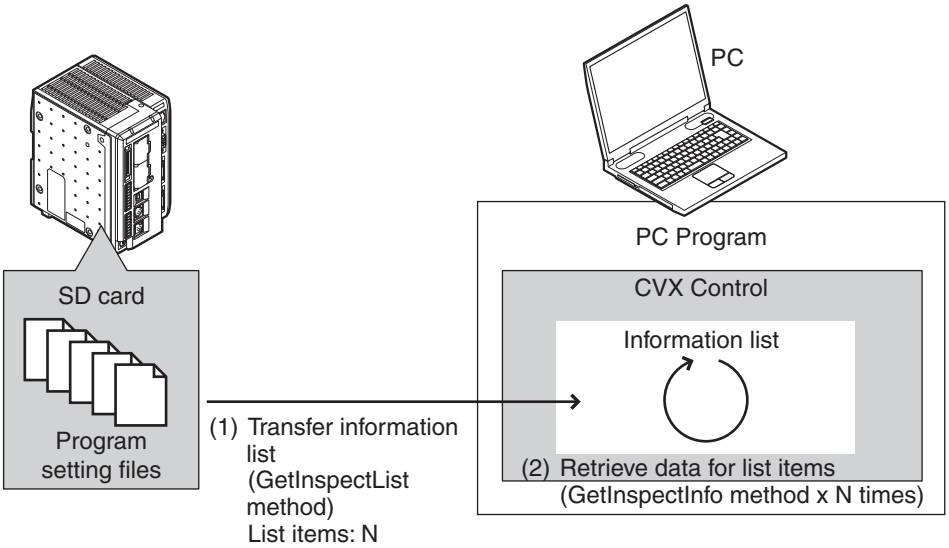


Figure 10-1. Retrieving the list for inspection setting files

Sample source code

Retrieves the list of program setting files when the Bn_GetInspectList button is clicked.
The process is aborted if an error occurs.

```
Private Sub Bn_GetInspectList_Click(sender As System.Object, _
e As System.EventArgs) Handles Bn_GetInspectList.Click
  'Establish the connection with the controller beforehand
  Dim state As Integer                    'for storing the return value
  Dim listNum As Integer                  'for storing the number of lists

  'Transfer the information list
  listNum = 0
  state = CVX.GetInspectList(listNum)

  If state = 0 Then                       'processing if this method succeeds
       Call MsgBox("Success")
  ElseIf state = 1100 Then                'processing when a communication exception
                                          occurs
       Call MsgBox("Communication exception")
                                          'connection to the controller must be
                                          re-established
       Exit Sub
  Else
       Call MsgBox("Execution error")     'processing when an execution error occurs
       Exit Sub
       'branch based on error code as necessary
  End If

  If listNum < 1 Then                     'exit if there are no list items
       Exit Sub
  End If

  'Retrieve each item in the transferred list
  Dim index As Integer                    'index for list items
  Dim languageID As Integer               'setting info: language information
  Dim driveNo As Integer                  'setting info: SD card No.
  Dim settingNo As Integer                'setting info: program setting No.
  Dim name As String = ""                 'setting info: program setting name
  languageID = 1                          'specifies the controller language

  For index = 0 To listNum - 1
       state = CVX.GetInspectInfo(index, languageId, driveNo, _
       settingNo, name)
       If state = 0 Then
            'processing if successful
            'Ex.: save retrieved data, reflect in control for display purposes, etc.
       Else
            'processing if execution error occurs
            'branch based on error code as necessary
       End If
  Next index
End Sub
```

9.1.2. GetInspectList method

Function

Retrieves a list of information on program setting data from the controller.

To retrieve information for each item in the list, use the GetInspectInfo method (Page 95).

Reference

- Refer to "9.1.1. Retrieving Information Lists" (Page 92) for more details on the use of this feature.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

Syntax

object.**GetInspectList(ByRef** *num* **As Integer) As Integer**

Arguments

num	Contains the number of information lists that were retrieved. Set to -1 if retrieval fails.

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected.
1003	Illegal operation	Settings file upload/download in progress (including other ActiveX controls) (download/upload of any type).
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

GetInspectInfo method (Page 95)

9.1.3. GetInspectInfo method

Function

Retrieves data from the program settings information list retrieved with the GetInspectList method (Page 94).

Reference Refer to "9.1.1. Retrieving Information Lists" (Page 92) for more details on the use of this feature.

Syntax

*object.***GetInspectInfo(ByVal** *index* **As Integer,**
ByVal *languageId* **As Integer, ByRef** *driveNo* **As Integer, _**
ByRef *settingNo* **As Integer ByRef** *name* **As String) As Integer**

Arguments

index	Specify the list number (0 to (lists retrieved - 1)). "Lists retrieved" is the value returned by calling the GetInspectList method (Page 94).
languageId	Specify the language of the string which will be stored in "name" by No. 0 OS's language 1 Controller's language 2 English 3 Japanese 4 Simplified Chinese 5 Traditional Chinese 6 Korean 7 Thai 8 German 9 French 10 Italian 11 Mexican Spanish 12 Indonesian 13 Brazilian Portuguese 14 Vietnamese
driveNo	Contains the SD card number. Set to -1 if retrieval fails. 1: SD1 2: SD2
settingNo	Contains the program setting number (0 to 999). Set to -1 if retrieval fails.
name	Contains the program name using up to 60 characters. Contains null (empty) if the method fails.

Return value

0	Processing succeeded	
1002	Illegal argument	List number is out of range.
1003	Illegal operation	All calls to the GetInspectList method (Page 94) have failed.

Related items

GetInspectList method (Page 94)

9.1.4. GetDictionaryList method

Function

Retrieves a numbered list of library files from the controller.

To retrieve information in the list, use the GetDictionaryInfo method (Page 97).

Reference Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

Syntax

*object.***GetDictionaryList(ByRef** *num* **As Integer) As Integer**

Arguments

num	Contains the number of information lists that were retrieved. Set to -1 if retrieval fails.

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected.
1003	Illegal operation	Settings file upload/download in progress (including other ActiveX controls). (download/upload of any type).
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

GetDictionaryInfo method (Page 97)

9.1.5. GetDictionaryInfo method

Function

Retrieves data from the library settings number list retrieved with the GetDictionaryList method (Page 96).

Syntax

*object.***GetDictionaryInfo(ByVal** *index* **As Integer, ByRef** *dicNo* **As Integer) As Integer**

Arguments

index	Specify the list number (0 to (lists retrieved - 1)). "Lists retrieved" is the value returned by calling the GetDictionaryList method (Page 96).
dicNo	Contains the library settings number. Set to -1 if retrieval fails.

Return value

0	Processing succeeded	
1002	Illegal argument	List number is out of range.
1003	Illegal operation	All calls to the GetDictionaryList method (Page 96) have failed.

Related items

GetDictionaryList method (Page 96)

Using the Remote Desktop

Receiving Results Data Output

Receiving Image Data Output

Retrieving Archived Image Data

Retrieving Information of Setting Files

▶ **Sending/Receiving/Deleting Files**

Issuing Non-Procedural Commands

Non-communication Functions

10.Sending/Receiving/ Deleting Files

Any file can be uploaded/downloaded/deleted between the SD card in the CV-X Series Controller and the PC (file transfer/deletion function).

- Hereinafter, the act of transferring files on the controller's SD card to the PC will be referred to as "downloading", and from the PC to the SD card as "uploading".
- The API allows you to verify which files to send/receive by retrieving a list of files in a specified folder on the SD card in the controller.
- For example, developers can create an application that retrieves and displays this list allowing users to select the actual file to download/upload.
- The controller can send/receive data files (program settings, registered images, library settings, global settings, learning data) which are required to perform inspections.
- Files specified by the controller can be deleted.

Point

- The communication link must be setup before you can use the functions in this chapter. Read "4. Common Settings for Communications" (Page 24) to initialize communication functions and connect to the controller.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

10.1. Retrieving a File List

This section explains how to retrieve a file list of the files in a specified folder on the SD card in the controller.

The list provides information on the file's type (folder or file), name, read attributes, date of creation, and size.

The data is obtained through a two-step process; transfer of the file list and retrieval of data for each item in the list.

Related items

GetFileList method (Page 104), GetFileInfo method (Page 105)

Procedures

Connect to the controller. (Refer to "4.3. Connecting to the Controller" (Page 26)).

(1) Transfer the file list.
Specify the folder on the SD card in the controller, and then call the GetFileList method to transfer the list for the files in the folder and allow retrieval of the number of items in the list.

(2) Call the GetFileInfo method to retrieve the data for each item in the list transferred in (1).
Specify the index of the list item and call this method in a loop for the number of items in the list.

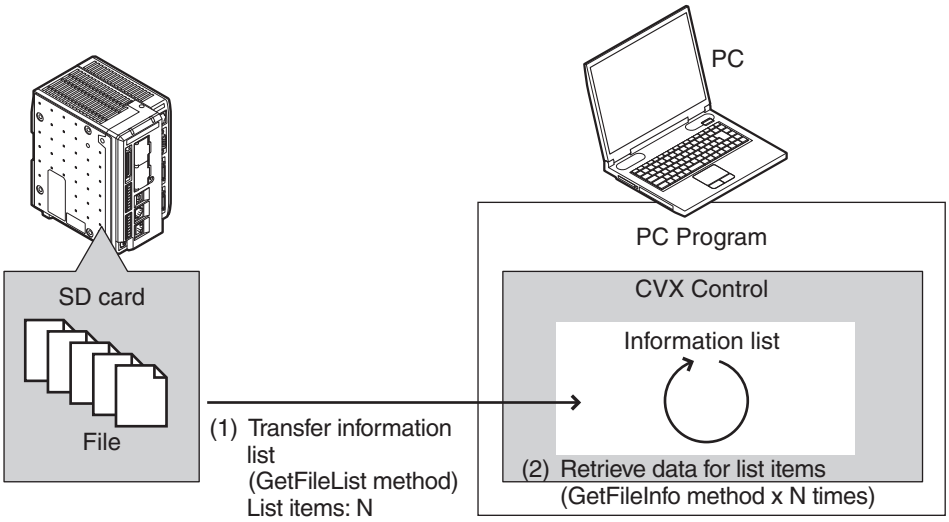


Figure 11-1. Retrieving the file list

Reference The locations and file names where the setting data files are stored are as follows.

File type	Location	File name
Program setting	Drive No.:/cv-x/setting/Program setting No./	inspect.dat
Registered image	Drive No.:/cv-x/setting/Program setting No./	refCameraNo._Registered image No.bmp or jpg
Learning data	Drive No.:/cv-x/setting/Program setting No./ T tool No.	SCD.tbd
Defective sample image	Drive No.:/cv-x/setting/Program setting No./ T tool No.	Defective sample No._NGMASTER_CAMCamera No._NG.bmp
Library	SD1:\cv-x\setting	dicLibrary No.dat
Global setting	SD1:\cv-x\setting	env.dat

Sample source code

The following code retrieves the list of files in folder "\cv-x\result" on SD Card 2 when the Bn_GetFileList button is clicked. The process is aborted if an error occurs.

```
Private Sub Bn_GetFileList_Click(sender As System.Object, _
e As System.EventArgs) Handles Bn_GetFileList.Click
  'Establish the connection with the controller beforehand
  Dim state As Integer                    'for storing the return value
  Dim listNum As Integer                  'for storing the number of lists

  'Transfer the file list
  listNum = 0
  state = CVX.GetFileList("SD2:\cv-x\result", listNum)

  If state = 0 Then                       'processing if this method succeeds
       Call MsgBox("Success")
  ElseIf state = 1100 Then                'processing when a communication exception
                                          occurs
       Call MsgBox("Communication exception")
                                          'connection to the controller must be re-
                                          established
       Exit Sub
  Else
       Call MsgBox("Execution error")     'processing when an execution error occurs
       Exit Sub
       'branch based on error code as necessary
  End If

  If listNum < 1 Then                     'exit if there are no list items
       Exit Sub
  End If

  'Retrieve each item in the transferred list
  Dim index As Integer                    'index for list items
  Dim name As String = ""                 'file information: file or folder name
  Dim kind As Short                       'file information: type (file or folder)
  Dim attr As Integer                     'file information: read attribute
  Dim dateTime As Date                    'file information: date of creation
  Dim size As Integer                     'file information: size (bytes)

  For index = 0 To listNum - 1
       state = CVX.GetFileInfo(index, name, kind, _
            attr, dateTime, size)
       If state = 0 Then
            'processing if successful
            'Ex.: save retrieved data, reflect in control for display purposes, etc.
       Else
            'processing if execution error occurs
            'branch based on error code as necessary
       End If
  Next index
End Sub
```

10.2. Sending/Receiving Files

File transfer (download/upload) starts when the method is called, and an event is notified when the transfer is completed.

Related items

DownloadFile method (Page 106), UploadFile method (Page 107), OnFileTransferred event (Page 109)

Procedures

Connect to the controller. (Refer to "4.3. Connecting to the Controller" (Page 26)).

(1) Start downloading (or uploading).
Specify the file name at the source and destination, then call the DownloadFile method (or UploadFile method) to start the file transfer.

(2) When the file transfer is complete, the OnFileTransferred event occurs.
The event notifies the processing results and type of transfer performed.

Sample source code

The first subroutine starts transferring the file "DL_sample_src.txt" on the SD card in the controller when the Bn_DownloadFile button is clicked. The file will be named "DL_sample_dst.txt" at the destination (PC).

The second subroutine starts transferring the file "UL_sample_src.txt" on the PC when the Bn_UploadFile button is clicked. The file will be named "UL_sample_dst.txt" at the destination (SD card in the controller).

When the file transfer is complete, a message box appears showing the result and type of transfer performed.

```
Private Sub Bn_DownloadFile_Click(sender As System.Object, _
e As System.EventArgs) Handles Bn_DownloadFile.Click
  'Establish the connection with the controller beforehand
  Dim state As Long                    'for storing the return value
  Dim srcFile, dstFile As String
                                       'file name (source, destination)

  srcFile = “SD2:\cv-x\DL_sample_src.txt”
                                       'downloading starts
  dstFile = “C:\cv-x\DL_sample_dst.txt”
  state = CVX.DownloadFile(srcFile, dstFile)

  If state = 0 Then                    'processing if this method succeeds
       Call MsgBox(“Success”)
  ElseIf state = 1100 Then             'processing when a communication exception
                                       occurs
       Call MsgBox(“Communication exception”)
                                       'connection to the controller must be re-
                                       established
  Else
       Call MsgBox(“Execution error”)  'processing when an execution error occurs
       'branch based on error code as necessary
  End If
End Sub
```

```
Private Sub Bn_UploadFile_Click(sender As System.Object, _
e As System.EventArgs) Handles Bn_UploadFile.Click
  'omitted (same as download)

  srcFile = "C:\cv-x\UL_sample_src.txt"
                                        'uploading starts
  dstFile = "SD2:\cv-x\UL_sample_dst.txt"
  state = CVX.UploadFile(srcFile, dstFile)

  'omitted (same as download)
End Sub

Private Sub CVX_OnFileTransferred(sender As System.Object, _
e As AxCVXLib._DCVXEvents_OnFileTransferredEvent) Handles _
CVX.OnFileTransferred
  Dim strKind As String                 'type of file transfer
  If e.kind = 0 Then
       strKind = "download complete: "
  Else
       strKind = "upload completed: "
  End If

  If e.state = 0 Then                   'processing if this method succeeds
       Call MsgBox(strKind & "Success")
  ElseIf e.state = 1100 Then            'processing when a communication exception
                                        occurs
       Call MsgBox(strKind & "Communication exception")
                                        'connection to the controller must be re-
                                        established
  Else
       Call MsgBox(strKind & "Execution error")
                                        'processing when an execution error occurs
       'branch based on error code as necessary
  End If
End Sub
```

10.3. API Reference

10.3.1. FileTransferBusy property

Function

Indicates whether the file transfer (DownloadFile method (Page 106), UploadFile method (Page 107)) is in progress (read-only).

This property is updated when the file retrieval method is called, and when the OnFileTransferred event (Page 109) is notified.

Syntax

*object.***FileTransferBusy**

Data type

Boolean

Arguments

True	File transfer in progress.
False	File transfer not in progress.

Initial value

False

Related items

DownloadFile method (Page 106), UploadFile method (Page 107), OnFileTransferred event (Page 109)

10.3.2. GetFileList method

Function

Retrieves the number of lists sent by the controller. These lists provide information on the folders and files under the user-specified folder.

Retrieve the data for each folder and file by calling the GetFileInfo method (Page 105).

Point

- This method can retrieve up to 1,000 lists (in random order).
- The list contains the current directory ".".
- If an SD card drive (SD1, SD2) or a USB HDD is specified, the list will contain the volume label if one has been assigned.

Reference

- Refer to "10.1. Retrieving a File List" (Page 99) for more details on the use of this feature.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

Syntax

*object.***GetFileList(ByVal** *folder* **As String, ByRef** *num* **As Integer) As Integer**

Arguments

folder	Specify the full path to the folder the list is to be retrieved for. Specify the drive name as "SD1:\", "SD2:\" or "USB:\". Enter up to 255 characters.
num	Contains the number of file lists that were retrieved (0 to 1000). Set to -1 if retrieval fails.

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected.
1002	Illegal argument	Illegal number of characters in folder name (NULL or exceeded maximum character count).
1003	Illegal operation	File upload/download in progress (all forms of uploads/downloads including those handled by other ActiveX controls)
1007	Not available	The SD card has not been inserted or the USB HDD has not been connected.
1009	No target data	The specified folder does not exist.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

GetFileInfo method (Page 105)

10.3.3. GetFileInfo method

Function

Retrieves the data for each list item (folder or file) from the file information list retrieved with the GetFileList method (Page 104).

Reference Refer to "10.1. Retrieving a File List" (Page 99) for more details on the use of this feature.

Syntax

object.**GetFileInfo(ByVal** *index* **As Integer, ByRef** *name* **As String, _**
ByRef *kind* **As Short, ByRef** *attribute* **As Integer, _**
ByRef *date* **As Date, ByRef** *size* **As Long) As Long**

Arguments

index	Specify the list number (0 to (lists retrieved - 1)). "Lists retrieved" is the value returned by the previous call to the GetFileList method.
name	Contains the name of the folder or file (including the extension). The name contains up to 255 characters. Contains null (empty) if the retrieval fails.
kind	Contains the type, folder or file. Set to -1 if retrieval fails. 0: folder 1: file
attribute	Contains the read attribute. Set to -1 if retrieval fails. 0: not read-only 1: read-only
date	Contains the file creation date. Set with 01-01-1980 00:00:00 if retrieval fails.
size	Contains the file size (units in bytes, max. 2 GB). Set with -1 if item is a folder, or if retrieval fails.

Return value

0	Processing succeeded	
1002	Illegal argument	List number is out of range.
1003	Illegal operation	All calls to the GetFileList method (Page 104) have failed.

Related items

GetFileList method (Page 104)

10.3.4. DownloadFile method

Function

Downloads a file from the SD card or USB HDD in the controller.

This method only posts a request. An event is notified when the operation is completed.

The operating status will be reflected in FileTransferBusy property (Page 103).

Point

If the file already exists at the destination, it will be overwritten.

Reference

- Refer to "10.2. Sending/Receiving Files" (Page 101) for more details on the use of this feature.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

Syntax

*object.***DownloadFile(ByVal** *srcFile* **As String, ByVal** *dstFile* **As String) As Integer**

Arguments

srcFile	Specify the full path of the file name at the source (on the controller). Specify the drive name as "SD1:\", "SD2:\" or "USB:\". Enter up to 255 characters.
dstFile	Specify the full path of the file name at the destination (on the PC). Enter up to 255 characters.

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected. The destination drive is read-only. Not enough memory space at destination. A folder with the specified file name already exists. The SD card is being accessed by another communication function.
1002	Illegal argument	Illegal number of characters in file name (NULL or exceeded maximum character count). Illegal file naming format.
1003	Illegal operation	File upload/download in progress (all forms of uploads/downloads including those handled by other ActiveX controls)
1007	Not available	The SD card has not been inserted or the USB HDD has not been connected.
1009	No target data	File does not exist at the source.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

When processing is completed, the OnFileTransferred event (Page 109) occurs.

Event results

0	Processing succeeded	
1001	Processing failed	Access to the destination file failed.
1004	Out of space	Not enough available space at destination.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

10.3.5. UploadFile method

Function

Uploads a file to the SD card or USB HDD in the controller.

- This method only posts a request. An event is notified when the operation is completed.
- The operating status will be reflected in FileTransferBusy property (Page 103).

Point If the file already exists at the destination, it will be overwritten.

Reference
- Refer to "10.2. Sending/Receiving Files" (Page 101) for more details on the use of this feature.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.

Syntax

*object.***UploadFile(ByVal** *srcFile* **As String, ByVal** *dstFile* **As String) As Integer**

Arguments

srcFile	Specify the full path of the file name at the source (on the PC). Enter up to 255 characters.
dstFile	Specify the full path of the file name at the destination (on the controller). Specify the drive name as "SD1:\", "SD2:\" or "USB:\". Enter up to 255 characters.

For characters that cannot be used in file names, refer to "Illegal Characters" (Page 127).

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected. A folder with the specified file name already exists. The destination drive is read-only. The SD card is being accessed by another communication function.
1002	Illegal argument	Illegal number of characters in file name (NULL or exceeded maximum character count). Illegal file naming format.
1003	Illegal operation	File upload/download in progress (all forms of uploads/downloads including those handled by other ActiveX controls)
1007	Not available	The SD card has not been inserted or the USB HDD has not been connected.
1009	No target data	File does not exist at the source.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Related items

When processing is completed, the OnFileTransferred event (Page 109) occurs.

Event results

0	Processing succeeded	
1001	Processing failed	Access to the destination file failed.
1004	Out of space	Not enough space on the SD card or USB HDD.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

10.3.6. OnFileTransferred event

Function

This event occurs when the file transfer (DownloadFile method (Page 106), UploadFile method (Page 107)) is completed.

Point This event is common for multiple methods. To distinguish which method the event refers to, check the arguments.

Reference Refer to "10.2. Sending/Receiving Files" (Page 101) for more details on the use of this feature.

Syntax

Private Sub *object*_**OnFileTransferred (ByVal** *sender* **As Object, _**
ByVal *e* **As AxCVXLib._DCVXEvents_OnFileTransferredEvent) _**
Handles *object*.**OnFileTransferred**

Arguments: *e.state* **As Integer,** *e.kind* **As Integer**

Arguments

state	Contains the processing result. Refer to "Processing result" for details.
kind	Contains the type of method that was completed. 0: DownloadFile 1: UploadFile

Processing result

0	Processing succeeded

For details on errors, refer to the [Event results] for each method.

Related items

DownloadFile method (Page 106), UploadFile method (Page 107), FileTransferBusy property (Page 103)

10.3.7.DeleteFile method

Function

Deletes files from the controller's SD card or USB HDD.

When an empty folder is specified, folder deletion is also possible.

Syntax

object.DeleteFile(ByVal dstFile As String) As Integer

Argument

dstFile	Specify the full pathname of the file to be deleted.Up to 255 characters can be entered. Specify "SD1:\", "SD2:\" or "USB:\" for the drive name. For characters that cannot be used in file names, refer to "Illegal Characters" in "Appendix".

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected. The destination drive is read-only. Another communication function is accessing the memory card of the controller.
1002	Illegal argument	The number of the characters of the file name is out of range. (Null or exceeding maximum character count)
1007	Unavailable	The SD card has not been inserted or the USB HDD has not been connected.
1009	No target data	The specified file for deletion does not exist.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Event results

0	Processing succeeded	
1001	Processing failed	Access to the destination file failed.
1100	Communication exceptions	Communication failure due to a fault in the communication path.

Using the Remote Desktop

Receiving Results Data Output

Receiving Image Data Output

Retrieving Archived Image Data

Retrieving Information of Setting Files

Sending/Receiving/Deleting Files

▶ **Issuing Non-Procedural Commands**

Non-communication Functions

11. Issuing Non-Procedural Commands

Users can issue non-procedural commands to the CV-X Series Controller and receive responses via communications (non-procedural commands)

- The non-procedural commands are a set of useful functions that allow users to control and send/retrieve data and status results from the connected CV-X controller.

Point

- In order to use this function, users must understand the functions of the non-procedural commands. For details on non-procedural commands, refer to the related manuals.
- Non-procedural commands executed from the CVX Control do not interfere with non-procedural commands input from other sources such as Ethernet, etc. The controller processes non-procedural commands from each input source in the order they are received.
- The communication link must be setup before you can use the functions in this chapter. Read "4. Common Settings for Communications" (Page 24) to initialize communication functions and connect to the controller.
- Refer to "Limitations When Using Multiple CVX Controls Simultaneously" (Page 128) for limitations that apply when using this function from multiple CVX Controls.
- When executing non-procedural commands, fluctuations will occur due to the load state of the controller at the time. If performing processing with the presumption that the command completed, confirm the completion of the command with the returned value and then begin the next process.

11.1. Executing Commands

Methods that execute non-procedural commands fall into two categories. Users can select either one depending on the length of execution time.

Synchronous methods (ExecuteCommand)

Synchronous methods issue the non-procedural command and then wait for a response that contains the execution result.
These methods are appropriate to execute non-procedural commands with relatively short execution times.
Most methods are of the synchronous type.

Asynchronous methods (ExecuteCommandAsync)

Asynchronous methods only issue the non-procedural command. These methods receive a response indicating success or failure of the command issuance.
When processing is completed on the controller, the CVX Control receives a non-procedural command response and issues a completion event.
These methods are appropriate to execute non-procedural commands with relatively long execution times.

Reference

- Refer to the following pages for details on the use of each of these methods.
- The following commands take a while to be executed. Therefore, ExecuteCommandAsync method is recommended to execute them.
  - Save Program SS
  - Change programs PW
  - Image Registration BS
  - Save statistical data TS
  - Save archived image HS
  - Image capture BC

11.1.1Executing synchronous methods

When a synchronous method is called, the method sends the non-procedural command and receives a response.
This process will be explained using the Switch to run mode command (R0 command) as an example.

Procedures

Connect to the controller. (Refer to "4.3. Connecting to the Controller" (Page 26)).
(1) Call the ExecuteCommand method.
(2) The following code retrieves the execution result from the return value.

Sample source code

Go to run mode when the Bn_ChangeMode button is clicked.

```
Private Sub Bn_ChangeMode_Click(sender As System.Object, _
e As System.EventArgs) Handles Bn_ExecuteCommand.Click
  'Establish the connection with the controller beforehand
  Dim state As Integer                    'for storing the return value
  Dim response As String = ""             'for storing the command response
  state = CVX.ExecuteCommand("R0", response)
                                          'issues "RO" command.

  If state = 0 Then                       'processing if this method succeeds
       Call MsgBox("Success")
  ElseIf state = 1100 Then                'processing when a communication exception
                                          occurs
       Call MsgBox("Communication exception")
                                          'connection to the controller must be
                                          re-established
  Else
       Call MsgBox("Execution error")     'processing when an execution error occurs
       'branch based on error code as necessary
  End If
End Sub
```

11.1.2.Executing asynchronous methods

When an asynchronous method is called, the method only sends the non-procedural command.
Completion of command processing is notified by an event.
This process will be explained using the Image capture (BC command) as an example.

Procedures

Connect to the controller. (Refer to "4.3. Connecting to the Controller" (Page 26)).

(1) Call the ExecuteCommandAsync method and send the non-procedural command (BC).

(2) When the command is executed and the response is received, the OnCommandFinished event occurs.
The OnCommandFinished event is common for all asynchronous methods.
The event notifies the processing results and type of command.

Sample source code

The following code sends the Image capture command when the Bn_CaptureScreen button is clicked.

Upon receiving the command response, the code displays a message box showing the result and type of transfer performed.

```
Private Sub Bn_CaptureScreen_Click(sender As System.Object, _
e As System.EventArgs) Handles Bn_CaptureScreen.Click
  'Establish the connection with the controller beforehand
  Dim state As Integer                    'for storing the return value
  state = CVX.ExecuteCommandAsync("BC")

  If state = 0 Then                       'processing if this method succeeds
       Call MsgBox(“Success”)
  ElseIf state = 1100 Then                'processing when a communication exception
                                          occurs
       Call MsgBox(“Communication exception”)
                                          'connection to the controller must be
                                          re-established
  Else
       Call MsgBox(“Execution error”)     'processing when an execution error occurs
       'branch based on error code as necessary
  End If
End Sub

Private Sub CVX_OnCommandFinished(sender As System.Object, _
e As AxCVXLib._DCVXEvents_OnCommandFinishedEvent) Handles _
CVX.OnCommandFinished

  If e.state = 0 Then                     'processing if this method succeeds
       Call MsgBox(Command() & “Success”)
  ElseIf e.state = 1100 Then              'processing when a communication exception
                                          occurs
       Call MsgBox(Command() &“Communication exception”)
                                          'connection to the controller must be
                                          re-established
  Else
       Call MsgBox(Command() &“Execution error”)
                                          'processing when an execution error occurs
       'branch based on error code as necessary
  End If
End Sub
```

11.2. Common Errors

The following errors can occur commonly for all methods for non-procedural commands.
The error is reflected in the return value for methods, or processing results for events.

Point

- Errors caused by a non-procedural command will have a value of (2000 + error code for the non-procedural command).
- For details on non-procedural command error causes, refer to the related manuals.

Table 12-2. Common Error Codes

Error Codes	Name	Explanation
1001	Processing failed (*1)	Communication functions are not initialized. The controller is not connected. Non-procedural command cannot be executed.
1003	Illegal operation (*1)	Non-procedural command execution in progress (including other ActiveX controls)
1100	Communication exception (*2)	Communication failure due to a fault in the communication path.
2003	Command operation inhibited (*3)	This error occurs if a non-procedural command is executed when the controller is reset, while changing operating modes, or while changing programs.
2081	Command ignored (*3)	Command cannot be executed while device input is disabled.

*1*Occurs only for method return values.
*2*If a communication exception (1100) occurs, the connection is disconnected. Reconnect to the controller.
*3*Does not occur with return values for asynchronous methods.

11.3. API Reference

11.3.1.CommandBusy property

Function

Indicates whether an asynchronous method is in progress (read-only).

This property is updated when an asynchronous method is called, and when the OnCommandFinished event (Page 119) is notified.

Syntax

`object.`**CommandBusy**

Data type

Boolean

Arguments

True	Asynchronous method in progress.
False	Asynchronous method not in progress.

Initial value

False

Related items

OnCommandFinished event (Page 119)

11.3.2.ExecuteCommand method

Function

Executes the specified non-procedural command.

This method always receives a response whether the command executes successfully or not.

Point

- Executes as a synchronous method.
- A time delay may occur before the response is received if the non-procedural command executed has a long execution time.

Reference

- Refer to "11.1.1. Executing synchronous methods" (Page 112) for more details on the use of this feature.
- The controller version must support the target non-procedural command in order to execute this method.

Syntax

*object.***ExecuteCommand(ByVal** *command* **As String, _**
ByRef *response* **As String) As Integer**

Arguments

command	Specify the command as a string.
response	Contains the command response. Contains null (empty) if the method fails.

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected.
1002	Illegal argument	Illegal number of characters in command string (NULL string or exceeded maximum string length).
2003	Execution Errors	Complies with the non-procedural command specification.
2022	Argument error	Complies with the non-procedural command specification.

Refer to "11.2. Common Errors" (Page 115) for other errors not listed above.

Related items

None

11.3.3.ExecuteCommandAsync method

Function

Executes the specified non-procedural command.

All commands will be executed asynchronously.

Syntax

object.**ExecuteCommandAsync(ByVal** command **As String) As Integer**

Argument

command	Command string to be executed

Return value

0	Processing succeeded	
1001	Processing failed	Communication functions are not initialized. The controller is not connected.
1002	The number of the characters of the command is out of range. (Null or exceeding the maximum character count)	
2022	Argument error	According to the specifications of the non-procedural commands.
2003	Execution error	According to the specifications of the non-procedural commands.

Related items

When the CommandBusy property and execution are completed, the OnCommandFinished event occurs.

11.3.4.OnCommandFinished event

Function

Occurs when the asynchronous function that makes use of the non-procedural command service is completed.

Reference

- This event is common for multiple methods. To distinguish which method the event refers to, check the arguments.
- Refer to "11.1.2. Executing asynchronous methods" (Page 114) for more details on the use of this feature.

Syntax

Private Sub *object*_**OnCommandFinished (ByVal** *sender* **As Object, _**
ByVal *e* **As AxCVXLib._DCVXEvents_OnCommandFinishedEvent) _**
Handles *object*.**OnCommandFinished**

Arguments: *e.state* **As Integer,** *e.command* **As String**

Arguments

state	Contains the processing result. Refer to "Processing result" for details.
command	Contains the command assigned by ExecuteCommandAsync.

Event results

0	Processing succeeded

For details on errors, refer to the [Event results] for each method.

Related items

ExecuteCommandAsync method (Page 118)

12. Non-communication Functions

This chapter explains those functions of the CVX Control that do not rely on communications.

Using the Remote Desktop

Receiving Results Data Output

Receiving Image Data Output

Retrieving Archived Image Data

Retrieving Information of Setting Files

Sending/Receiving/Deleting Files

Issuing Non-Procedural Commands

▶ **Non-communication Functions**

12.1. Reading the Version Number of the CVX Control

Use the Version property to read the version of the CVX Control.
User programs should utilize this to check the version of the control before using functions added in recent releases.

12.2. Adjusting the Position and Size of the CVX Control

The control has properties for position and size that can be set and read.

Adjusting the Position of the CVX Control

- The position of the CVX Control depends on the Top and Left properties, the control also has Location, Bottom, and Right properties.

Adjusting the Size of the CVX Control

- Members of the Size property, the size of the CVX Control depends on the Width and Height properties.

Point

- These properties are set in units of pixels.
- The resolution of the controller's monitor screen is 800 x 600 (H x V) pixels or 1024 x 768 (H x V) pixels.
- The CVX Control automatically holds the user-specified size setting to an aspect ratio of 4:3.
- This is done by expanding or shrinking the controller screen to fit within the user-specified size. For example, if the CVX Control size is set to 400 x 300 (H x V) pixels, the control will appear reduced to 1/4th of the actual monitor screen size.

Reference

For technical details on properties, read the technical literature from Microsoft Corporation.

12.3. Hiding the Control

To hide the CVX Control, set the Visible property to False.
Use this feature when the Remote Desktop is not being used.

Reference

For technical details on properties, refer to the technical literature from Microsoft Corporation.

12.4. Mouse and Keyboard Event Driven Processing

The following events occur when the mouse and keyboard are operated over the CVX Control.
Use this feature to perform event driven processing based on user operations.

12.4.1.Mouse events

Button press

MouseDownEvent event

Button release

MouseUpEvent event

Click

ClickEvent event

Double click

DblClick event

Move

MouseMoveEvent event

12.4.2.Keyboard events

Key press:

KeyDownEvent event

Key release:

KeyUpEvent event

Reference

- For technical details on properties, refer to the technical literature from Microsoft Corporation.

12.5. API Reference

12.5.1.Version property

Function

Indicates the version of the CVX Control (read-only).

Syntax

*object.***Version**

Data type

String

Arguments

string	Contains the version of the CVX Control as a text string delimited by periods. (Example: "1.0")

Initial value

Version of CVX Control (fixed value)

Related items

None

Appendix

Appendix

Error Codes

If an error occurs while a function is executing, the CVX Control will identify the error in the method's return value or in the event status. The following list shows the errors notified by the CVX Control.

Table 0-1. List of Error Codes

Error Codes	Name	Explanation
1001	Processing failed	Processing failed (initialization not done, etc.)
1002	Illegal argument	An illegal argument was specified.
1003	Illegal operation	The operation requested is illegal (duplicate method calls, etc.)
1004	Out of space	Not enough memory space.
1005	File exists	The file already exists.
1006	Processing in progress	Processing in progress.
1007	Not available	The device is unavailable.
1008	Operation not allowed	The requested operation has been disabled.
1009	No target data	Cannot find the target data (or file).
1011	Not supported	The controller's version is old.
1100	Communication exception (*2)	Communication failure due to a fault in the communication path.
2002	Command error (*3)	The requested command does not exist.
2003	Command operation inhibited (*3)	The received command cannot be performed.
2022	Argument error (*3)	Value specified is out of range.
2080	Incorrect password (*3)	The specified password is incorrect.
2081	Command ignored (*3)	Command cannot be executed while device input is disabled.

*1 These error codes can occur for different reasons depending on the function.
See the method and event definitions for details on errors that can occur for each function.

*2 If a communication exception (1100) occurs, the connection is disconnected.
Reconnect to the controller.

*3 Errors caused by a non-procedural command will have a value of (2000 + error code for the non-procedural command).

Illegal Characters

The following characters cannot be used in folder or file names for method arguments.

If an illegal character is used, the method's return value will notify error code 1002 (Argument error).

1. Illegal characters for folder names

/, *, ?, ", <, >, |, :

Point ":" can be used to denote a drive name such as "SD1:\".

2. Illegal characters for file names

/, *, ?, ", <, >, |, :, \

Point The use of a period or space as a single text string is also not allowed.

Limitations When Using Multiple CVX Controls Simultaneously

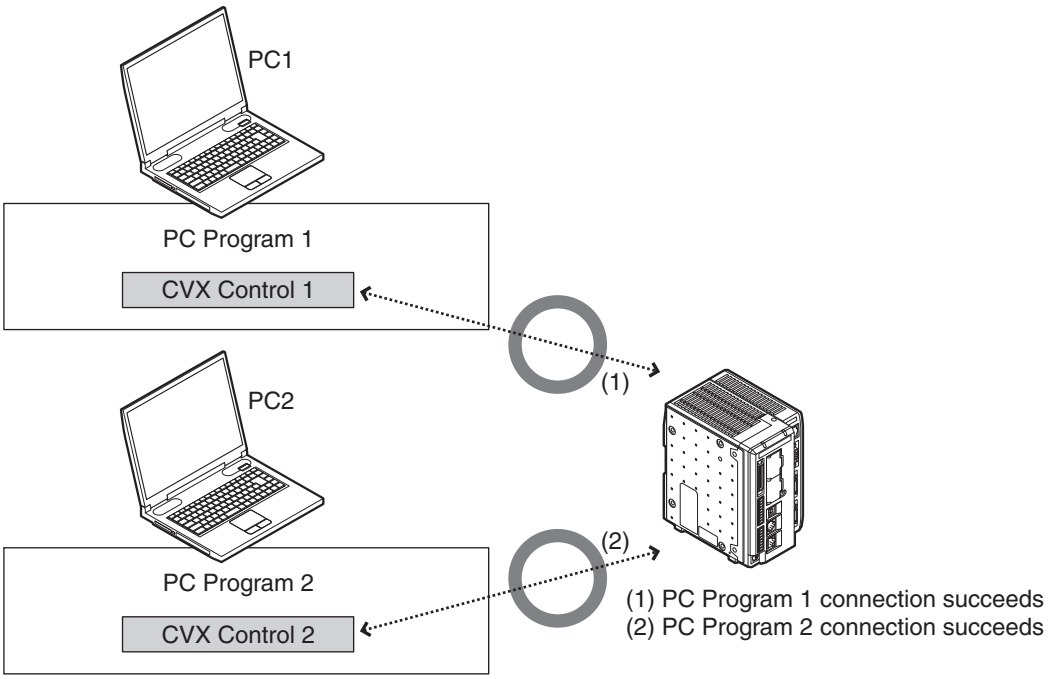
The following limitations apply when executing the commands in Chapters 5 to 12 from multiple CVX Controls.

Read "4.5.Connection Limitations" (Page 28) in this manual for an understanding on this functionality.

1. Remote desktop

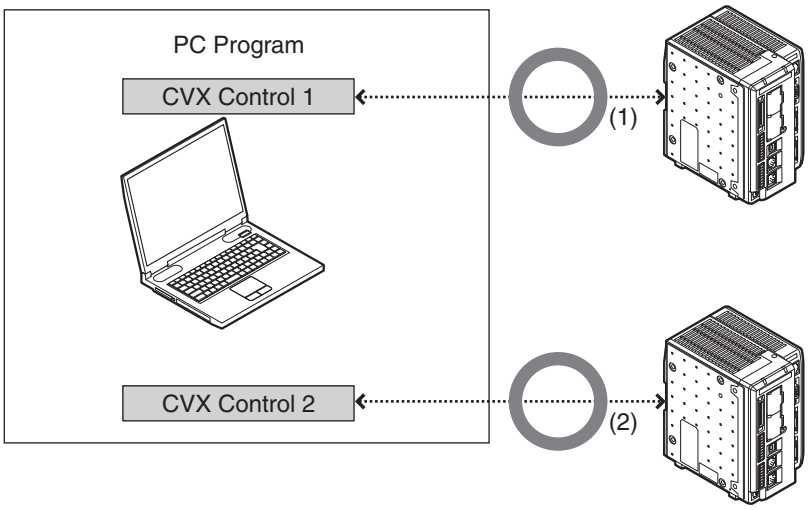
No limitations apply when executing the remote desktop function from multiple CVX Controls at the same time.

In the example shown below, both CVX Controls can execute the remote desktop function at the same time.



Point Controller screen updates may be delayed because the controller has to transfer screens to each control.

Multiple CVX Controls placed on the same application form can be used to execute separate controllers.

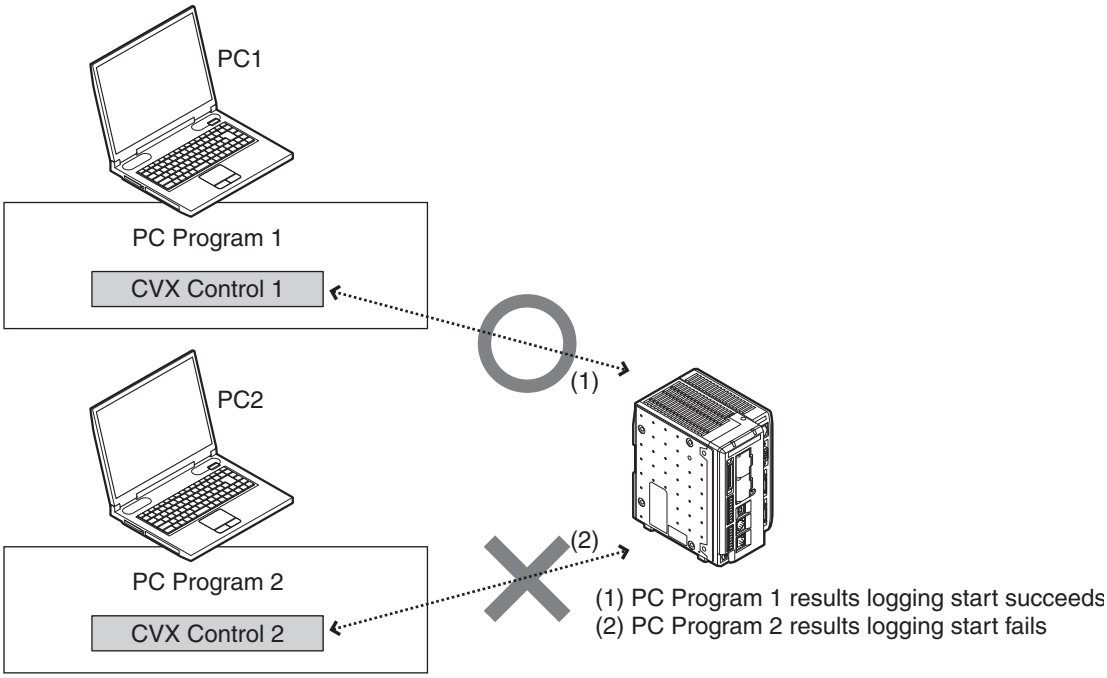


2. Results logging, Image logging

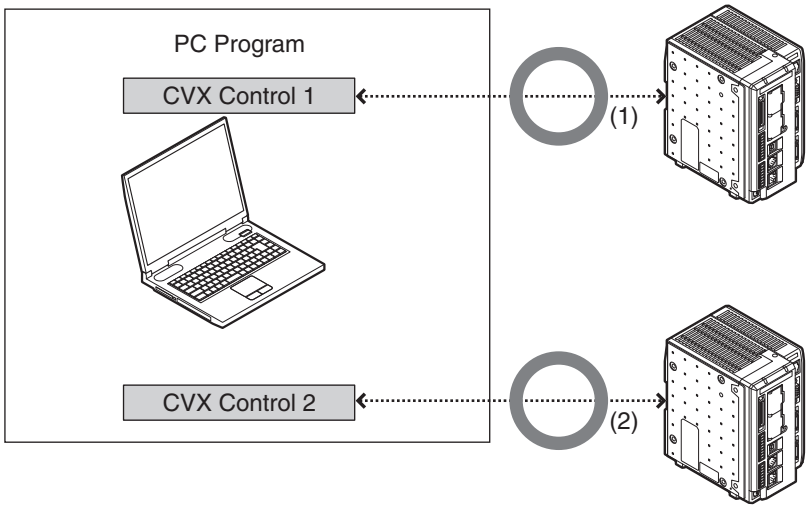
No limitations apply when logging different items (such as results and images) at the same time.
The limitation shown below applies when the same type of data is logged simultaneously (such as two controls where each is logging results).

Point The same limitation that applies to the CVX Control also applies to the CV-X Series Terminal-Software which can log results and images.

Only one logging function can be used for a single controller at one time.
If the logging start method (StartResultLog method, etc.) is called while logging is already in progress, the return value will notify error code 1001 (Processing failed error).



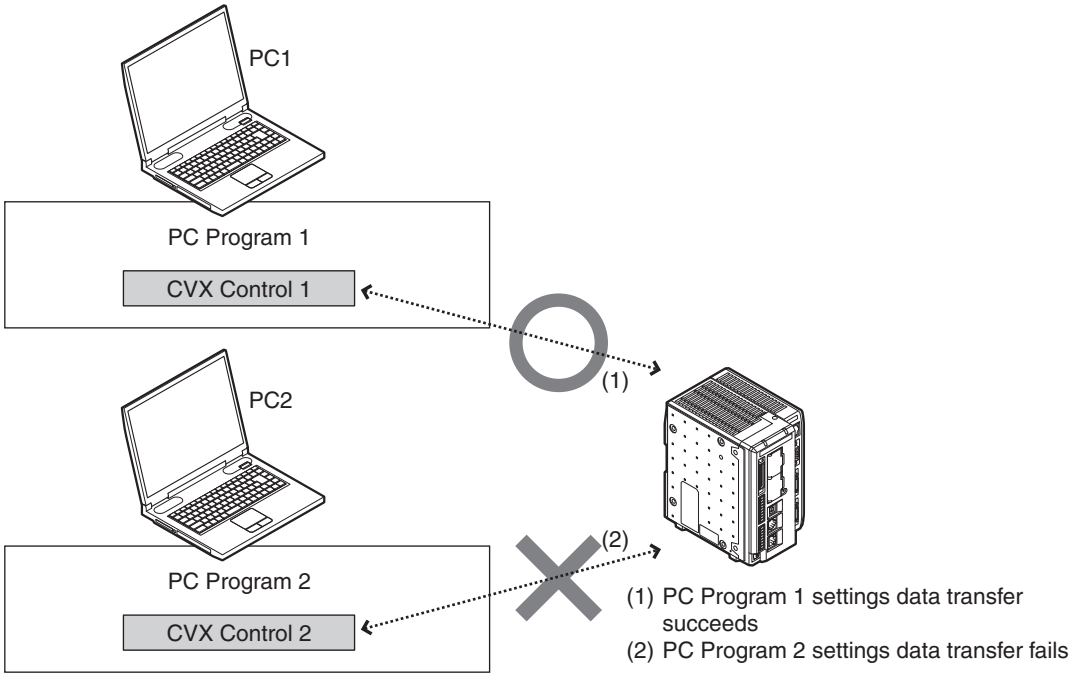
Multiple CVX Controls placed on the same application form can be used to execute separate controllers similar to operation with the remote desktop function.



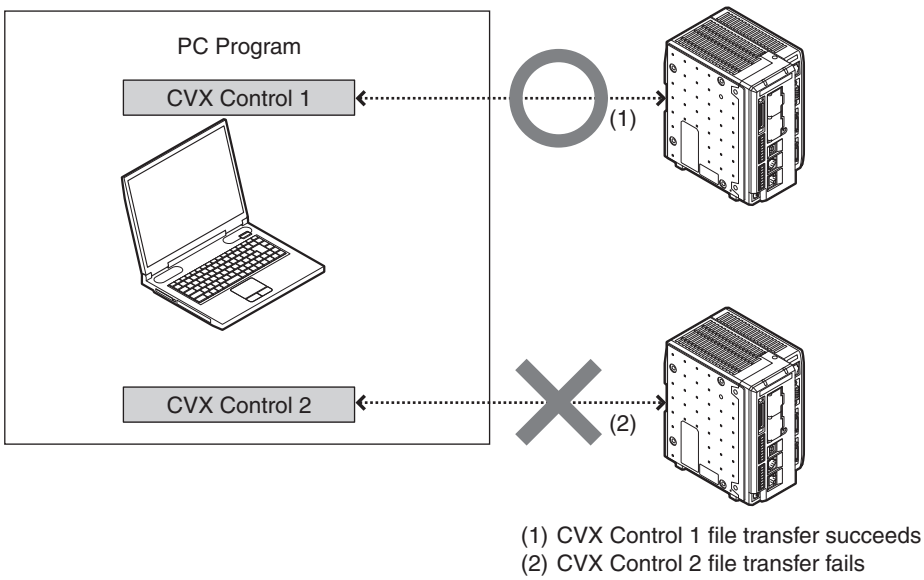
3. Other Communication Functions (Archived Image Data Transfers, File Transfers, Non-procedural Commands)

No limitations apply to simultaneous execution of functions that deal with different data types (such as archived image data transfers and file transfers).
The limitation shown below applies when the same type of data communication is used simultaneously (such as two controls where each is transferring files).

As with the logging function, only one communication function can be used for a single controller at one time.
If another communication function (DownloadFile method, etc.) is called while a communication function is already in progress, the return value will notify error code 1001 (CV-H1X Processing failed error).



Unlike the remote desktop function and logging functions, a single application with multiple CVX Controls can only execute one communication function at a time, even if each control is linked to a separate controller.
If another communication function (UploadFile method, etc.) is called while a communication function is already in progress, the return value will notify error code 1003 (Illegal operation error).



Revision History

Printing date	Edition	Revision
October 2012	First edition	
March 2013	2nd edition	Error corrections
March 2013	3rd edition	
April 2014	4th edition	
July 2014	5th edition	
October 2014	6th edition	
October 2016	7th edition	
June 2017	8th edition	

WARRANTIES AND DISCLAIMERS

(1) KEYENCE warrants the Products to be free of defects in materials and workmanship for a period of one (1) year from the date of shipment. If any models or samples were shown to Buyer, such models or samples were used merely to illustrate the general type and quality of the Products and not to represent that the Products would necessarily conform to said models or samples. Any Products found to be defective must be shipped to KEYENCE with all shipping costs paid by Buyer or offered to KEYENCE for inspection and examination. Upon examination by KEYENCE, KEYENCE, at its sole option, will refund the purchase price of, or repair or replace at no charge any Products found to be defective. This warranty does not apply to any defects resulting from any action of Buyer, including but not limited to improper installation, improper interfacing, improper repair, unauthorized modification, misapplication and mishandling, such as exposure to excessive current, heat, coldness, moisture, vibration or outdoors air. Components which wear are not warranted.

(2) KEYENCE is pleased to offer suggestions on the use of its various Products. They are only suggestions, and it is Buyer's responsibility to ascertain the fitness of the Products for Buyer's intended use. KEYENCE will not be responsible for any damages that may result from the use of the Products.

(3) The Products and any samples ("Products/Samples") supplied to Buyer are not to be used internally in humans, for human transportation, as safety devices or fail-safe systems, unless their written specifications state otherwise. Should any Products/Samples be used in such a manner or misused in any way, KEYENCE assumes no responsibility, and additionally Buyer will indemnify KEYENCE and hold KEYENCE harmless from any liability or damage whatsoever arising out of any misuse of the Products/Samples.

(4) **OTHER THAN AS STATED HEREIN, THE PRODUCTS/SAMPLES ARE PROVIDED WITH NO OTHER WARRANTIES WHATSOEVER. ALL EXPRESS, IMPLIED, AND STATUTORY WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF PROPRIETARY RIGHTS, ARE EXPRESSLY DISCLAIMED. IN NO EVENT SHALL KEYENCE AND ITS AFFILIATED ENTITIES BE LIABLE TO ANY PERSON OR ENTITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, PUNITIVE, SPECIAL OR CONSEQUENTIAL DAMAGES (INCLUDING, WITHOUT LIMITATION, ANY DAMAGES RESULTING FROM LOSS OF USE, BUSINESS INTERRUPTION, LOSS OF INFORMATION, LOSS OR INACCURACY OF DATA, LOSS OF PROFITS, LOSS OF SAVINGS, THE COST OF PROCUREMENT OF SUBSTITUTED GOODS, SERVICES OR TECHNOLOGIES, OR FOR ANY MATTER ARISING OUT OF OR IN CONNECTION WITH THE USE OR INABILITY TO USE THE PRODUCTS, EVEN IF KEYENCE OR ONE OF ITS AFFILIATED ENTITIES WAS ADVISED OF A POSSIBLE THIRD PARTY'S CLAIM FOR DAMAGES OR ANY OTHER CLAIM AGAINST BUYER.** In some jurisdictions, some of the foregoing warranty disclaimers or damage limitations may not apply.

BUYER'S TRANSFER OBLIGATIONS:

If the Products/Samples purchased by Buyer are to be resold or delivered to a third party, Buyer must provide such third party with a copy of this document, all specifications, manuals, catalogs, leaflets and written information provided to Buyer pertaining to the Products/Samples.

Specifications are subject to change without notice.

KEYENCE CORPORATION

1-3-14, Higashi-Nakajima, Higashi-Yodogawa-ku, Osaka, 533-8555, Japan PHONE: +81-6-6379-2211

www.keyence.com

AUSTRIA
Phone: +43 22 36-3782 66-0

BELGIUM
Phone: +32 1 528 1222

BRAZIL
Phone: +55-11-3045-4011

CANADA
Phone: +1-905-366-7655

CHINA
Phone: +86-21-3357-1001

CZECH REPUBLIC
Phone: +420 222 191 483

FRANCE
Phone: +33 1 56 37 78 00

GERMANY
Phone: +49 6102 36 89-0

HONG KONG
Phone: +852-3104-1010

HUNGARY
Phone: +36 1 802 73 60

INDIA
Phone: +91-44-4963-0900

INDONESIA
Phone: +62-21-2966-0120

ITALY
Phone: +39-02-6688220

KOREA
Phone: +82-31-789-4300

MALAYSIA
Phone: +60-3-7883-2211

MEXICO
Phone: +52-55-8850-0100

NETHERLANDS
Phone: +31 40 20 66 100

PHILIPPINES
Phone: +63-(0)2-981-5000

POLAND
Phone: +48 71 36861 60

ROMANIA
Phone: +40 269-232-808

SINGAPORE
Phone: +65-6392-1011

SLOVAKIA
Phone: +421 2 5939 6461

SLOVENIA
Phone: +386 1-4701-666

SWITZERLAND
Phone: +41 43-45577 30

TAIWAN
Phone: +886-2-2721-8080

THAILAND
Phone: +66-2-369-2777

UK & IRELAND
Phone: +44-1908-696900

USA
Phone: +1-201-930-0100

VIETNAM
Phone: +84-4-3772-5555

A4WW1-MAN-1037

Copyright (c) 2015 KEYENCE CORPORATION. All rights reserved. 124271E 1067-8 234GB Printed in Japan

234GB-8