# KEYENCE
# LJ-V7000 Provider

Version 1.0.0

# User's Guide

Aug 30, 2018

Remarks:
Do not change the settings with other applications while the device is connected by the provider.
This document is translated from Japanese into English by the machine translation.

## [Revision History]

| Version | Date | Contents |
|---|---|---|
| 1.0.0 | 2018-08-30 | First Edition |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## [Hardware]

| Model | Version | Notes |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## Contents

# 1. Introduction

This document is the user's guide for the LJ-V7000 provider, a CAO provider for the LJ-V7000 series of two-dimensional laser displacement monitors manufactured by KEYENCE.

LJ-V7000 providers communicate with Ethernet connected LJ-V7000 series controllers to control measurements and acquire measurement results.

This document describes the functionality of this LJ-V7000 provider and the methods it implements.

# 2. Outline of provider

## 2.1. Outline

LJ-V7000 provider provides communication functions by executing commands in the KEYENCE Communication Library via CaoController::Execute.

**Table 2-1 LJ-V7000 provider**

| | |
|---|---|
| File name | CaoProvKEYENCELJV7000.dll |
| ProgID | CaoProv.KEYENCE.LJ-V7000 |
| Registration | Regsvr32 CaoProvKEYENCELJV7000.dll |
| Deregistration | Regsvr32 /u CaoProvKEYENCELJV7000.dll |

**Table 2-2 Dependent Modules**

| No. | File name | Description |
|---|---|---|
| 1 | LJV7_IF.dll | KEYENCE's Ethernet, USB-communication library |
| 2 | vcredist_x86.exe | Microsoft Visual C++2008 redistributable package (x86)<br>(You must install before you register a provider) |

## 2.2. Method and property

### 2.2.1. CaoWorkspace::AddController method

Initializes KEYENCE libraries and connects to products.

Up to six devices can be connected.

Syntax AddController(<bstrCtrlName: VT_BSTR>, <bstrProvName: VT_BSTR>,
<bstrPcName:VT_BSTR >[,<bstrOption:VT_BSTR>])

| | | | |
|---|---|---|---|
| BstrCtrlName | : | [in] | CAO controller name<br>Specify any character string that does not overlap on a per-connection unit.<br>* If you specify the same name from a different application or another PC, the error is 0x80000205.<br>If you specify an empty string (""), the CAO engine automatically assigns a unique CAO controller name. |
| bstrProvName | : | [in] | Provider name. Fixed value ="CaoProv.KEYENCE.LJ-V7000". |
| bstrPcName | : | [in] | Provider's execution machine name<br>Specify an empty string (""), if on the same machine. |
| bstrOption | : | [in] | Option text="<Option 1>, <Option 2>,... " |

The following list specifies the option string.

**Table 2-3 Option strings of CaoWorkspace::AddController**

| Option | Description |
|---|---|
| Conn = eth:<IP address>[:port number] | Required. Specify the IP address of the controller of the displacement meter to be connected.<br>Only the port number can be omitted. (Default: 24691)<br>Example :"Conn=eth:192.168.0.1:24691" |
| DeviceID=<ID> | Required. Set the ID unique to the controller to be connected.<br>Set IDs in the range of 0 to 5. If you multiple connection IDs, do not duplication the IDs.<br>Example :"DeviceID=1" |
| Head =<head count> | Required. Specify the number of heads (1 or 2) connected to the controller to be connected.<br>Example :"Head=1" |
| HSDComm = <High-speed data communication ON/OFF>[<Port | Specify whether to initiate high-speed data communication during AddController. When high-speed data communication is |

| | |
|---|---|
| number for high-speed data communication>:<Number of profiles to be transmitted at one time>] | OFF (the first parameter is 0), the second and subsequent parameters can be omitted. (Default: 0)<br><High-Speed Data Communication ON/OFF>:<br>0: OFF<br>1: ON<br>Example :<br>"HSDComm=0"<br>"HSDComm=1:24692:10" |

Example

```
Dim caoCtrl As Object
caoCtrl = caoWs.AddController("LJV7",
                              "CaoProv.KEYENCE.LJ-V7000",
                              "",
                              "conn=192.168.0.1:24691, DeviceID=1,
                                    Head=1, HSDComm=1:24692:10")
```

### 2.2.2. CaoController::Execute method

Sends and receives commands. Specify the command name as the first argument and the command parameter as the second argument. Details of each command are given in 3. See 3 Command reference.

Syntax Execute(<bstrCommandName:BSTR>[, <vntParam:VT_VARIANT>])

| | | | |
|---|---|---|---|
| bstrCommandName | : | [in] | Command name |
| vntParam | : | [in] | Command argument |

## 2.3. Event

### 2.3.1. CaoController::OnMessage event

Data output during high-speed data communication is received.

Data reception may delay when sampling frequency of device is high. In that case, receive the multiple profiles collectively by one of the following means.

1. Increase the 2nd argument of StartHighSpeedDataCommunication command (dwProfileCnt).
2. Increase the 3rd argument of HSDComm option of AddController(Number of profiles to be transmitted at one time).

**Table 2-4 Event type list**

| Type | Description | Value content |
|---|---|---|
| 0x<br>(x: DeviceID) | Profile information | The profile measured by the controllers with the IDs specified by the DeviceID options is stored in the order shown in Information Table 2-5. Profile data is received at the same time as profile data is received from the same DeviceID of controllers.<br>The value does not change while high-speed data communication continues. |
| 1x<br>(x: DeviceID) | Profile data | Gets the profile data measured by the controller with the IDs specified by the DeviceID options.<br>You receive data when you accumulate data in the StartHighSpeedDataCommunication arguments or the number of profiles specified by the HSDComm options.<br>Refer to Section 9.2.9.6 of the LJ-V7000 Series Communication Library Reference Manual for the number of profile data. |

**Table 2-5 Stored profile data (VT_ARRAY|VT_VARIANT)**

| Index | Content | Data type |
|---|---|---|
| 0 | Number of profiles stored in one unit of data | VT_UI1 |
| 1 | ON/OFF of profile compression (time axis) | VT_UI1 |
| 2 | Number of data points in one profile | VT_UI2 |
| 3 | First X coordinate | VT_I4 |
| 4 | Spacing of the data points in the X direction | VT_I4 |

## 2.4. Error code

The LJ-V7000 provider defines the following unique error codes:

For ORiN2 common errors, see the Error Codes section in the ORiN2 programming guide.C:¥ORiN2¥CAO¥Doc¥ORiN2_ProgrammersGuide_en.pdf

**Table 2-6 Unique error code list**

| Error name | Error number | Description |
|---|---|---|
| Library error | 0x8010xxxx | A communication library error.<br>See "LJ-V7000 Series Communication Library" for details of the error code. |

# 3. Command reference

This chapter explains the commands of the CaoController::Execute method. Refer to KEYENCE's "LJ-V7000 Series Communication Library Reference Manual" for detailed operation of the commands.

**Table 3-1 CaoController::Execute command list**

| Command | LJ-V7000 Command | Function | Page |
|---|---|---|---|
| Systems control | | | |
| GetError | GetError | Gets system error information for the controller. | 11 |
| ClearError | ClearError | Release the system error of the controller. | 11 |
| Measurement controlling | | | |
| Trigger | Trigger | Issue a trigger | 12 |
| StartMeasure | StartMeasure | Start batch measurement | 12 |
| StopMeasure | StopMeasure | Batch measurement is completed. | 12 |
| AutoZero | AutoZero | Enter ZERO (for high-function modes only) | 13 |
| Timing | Timing | Enter TIMING (for high-function modes only) | 13 |
| Reset | Reset | Enter RESET (for high-function modes only) | 14 |
| ClearMemory | ClearMemory | Clear the memory inside the controller of the displacement meter. | 14 |
| Setting change/read related | | | |
| GetTime | GetTime | Gets the date and time of the controller. | 14 |
| GetActiveProgram | GetActiveProgram | Get the active program | 15 |
| ChangeAcriveProgram | ChangeAcriveProgram | Switch the active program | 15 |
| Acquisition of measurement results | | | |
| GetMeasurementValue | GetMeasurementValue | Gets the current value of the OUT measurement. | 15 |
| GetProfile | GetProfile | Gets the specified profile | 16 |
| GetBatchProfile | GetBatchProfile | Gets the profile of the specified batch measurement | 17 |
| GetProfileAdvance | GetProfileAdvance | Gets the specified profile | 19 |
| GetBatchProfileAdvance | GetBatchProfileAdvance | Gets the profile of the specified batch measurement | 20 |
| Storage function related | | | |
| StartStorage | StartStorage | Start storage | 21 |
| StopStorate | StopStorate | Exit the storage | 21 |
| GetStorageStatus | GetStorageStatus | Retrieve the storage status | 22 |
| GetStorageData | GetStorageData | Gets the OUT measurement in storage | 22 |
| GetStorageProfile | GetStorageProfile | Gets the profile data in storage | 23 |
| GetStorageBatchProfile | GetStorageBatchProfile | Gets batch profile data in storage | 25 |
| High-speed data communications | | | |
| StartHighSpeed DataCommunication | HighSpeedDataEthernet CommunicationInitialize<br>PreStartHighSpeedData Communication<br>StartHighSpeedData Communication | Start high-speed data communication | 26 |

| | | | |
|---|---|---|---|
| StopHighSpeed DataCommunication | StopHighSpeedData Communication HighSpeedData CommunicationFinalize | Stop high-speed data communication | 26 |

## 3.1. Systems control

### 3.1.1. CaoController::Execute("GetError") command

Retrieves the specified number of system error codes that have occurred in the controller.

Syntax GetError (<byRcvMax>)

byRcvMax : Maximum number of system errors to be acquired (VT_UI1)

Return value : The number of system error codes generated by the controllers and the acquired error codes (VT_ARRAY|VT_VARIANT)

The data is stored as shown in Table 3-2.

**Table 3-2 GetError data**

| Index | Content | Data type |
|---|---|---|
| 0 | Number of system errors occurring in the controller | VT_UI1 |
| 1~<byRcvMax> | Acquired error code | VT_UI2 |

Example

```
Dim retVal As Variant
retVal = caoCtrl.Execute("GetError", 5)
```

### 3.1.2. CaoController::Execute("ClearError") command

Releases the specified system error code. See Section 9.2.3 of the "LJ-V7000 Series Communication Library Reference Manual" for information on cancelable error codes.

Syntax ClearError (<wErrCode>)

wErrCode : System error code (VT_UI2) to be released

Return value : None

Example

```
caoCtrl.Execute "ClearError", &H0085
```

## 3.2. Measurement controlling

### 3.2.1. CaoController::Execute("Trigger") command

Issue a trigger.

Syntax Trigger()

Argument : None

Return value : None

Example

```
caoCtrl.Execute "Trigger"
```

### 3.2.2. CaoController::Execute("StartMeasure") command

Start batch measurement.

Syntax StartMeasure()

Argument : None

Return value : None

Example

```
caoCtrl.Execute "StartMeasure"
```

### 3.2.3. CaoController::Execute("StopMeasure") command

End the batch measurement.

Syntax StopMeasure()

Argument : None

Return value : None

Example

```
caoCtrl.Execute "StopMeasure"
```

### 3.2.4. CaoController::Execute("AutoZero") command

Issues an autozero request for the specified OUT measurement.

Syntax AutoZero (<byOnOff>, <dwOut>)

| | | |
|---|---|---|
| byOnOff | : | ON/OFF specify (VT_UI1) |
| dwOut | : | OUT measurement (bit specify) to be processed (VT_UI4)<br>E.g., dwOut = 0b0000000000010010 (0x0012) if OUT2,5 is the subject |
| Return value | : | None |

Example

```
caoCtrl.Execute "AutoZero", Array(1, &H0012)
```

### 3.2.5. CaoController::Execute("Timing") command

Issues a timing request for the specified OUT measurement.

Syntax Timing (<byOnOff>, <dwOut>)

| | | |
|---|---|---|
| byOnOff | : | ON/OFF specify (VT_UI1) |
| dwOut | : | OUT measurement (bit specify) to be processed (VT_UI4)<br>E.g., dwOut = 0b0000000000010010 (0x0012) if OUT2,5 is the subject |
| Return value | : | None |

Example

```
caoCtrl.Execute "Timing", Array(1, &H0012)
```

### 3.2.6. CaoController::Execute("Reset") command

Issues a reset request for the specified OUT measurement.

Syntax Reset (<dwOut>)

| | | |
|---|---|---|
| dwOut | : | OUT measurement (bit specify) to be processed (VT_UI4)<br>E.g., dwOut = 0b0000000000010010 (0x0012) if OUT2,5 is the subject |
| Return value | : | None |

Example

```
caoCtrl.Execute "AutoZero", &H0012
```

### 3.2.7. CaoController::Execute("ClearMemory") command

Clears the data stored in the controller.

Syntax ClearMemory ()

| | | |
|---|---|---|
| Argument | : | None |
| Return value | : | None |

Example

```
caoCtrl.Execute "ClearMemory"
```

## 3.3. Setting change/read related

### 3.3.1. CaoController::Execute("GetTime") command

Gets the date and time of the controller.

Syntax GetTime()

| | | |
|---|---|---|
| Argument | : | None |
| Return value | : | Controller date and time (VT_DATE) |

Example

```
Dim retVal As Variant
retVal = caoCtrl.Execute("GetTime")
```

### 3.3.2. CaoController::Execute("ChangeActiveProgram") command

Switches the active program number.

Syntax ChangeActiveProgram(<byProgNo>)

| | | |
|---|---|---|
| ByProgNo | : | Program No. after switching(VT_UI1) |
| Return value | : | None |

Example

```
caoCtrl.Execute "ChangeActiveProgram", 2
```

### 3.3.3. CaoController::Execute("GetActiveProgram") command

Gets the current active program number.

Syntax GetActiveProgram()

| | | |
|---|---|---|
| Argument | : | None |
| Return value | : | Active Program No. (VT_UI1) |

Example

```
Dim retVal As Variant
retVal = caoCtrl.Execute("GetActiveProgram")
```

## 3.4. Acquisition of measurement results

### 3.4.1. CaoController::Execute("GetMeasurementValue") command

Obtain the latest measurement value.

Syntax GetMeasurementValue ()

Argument : None

Return value : All OUT measurements (VT_ARRAY|VT_VARIANT)

The data is stored as shown in Table 3-3.

**Table 3-3 GetMeasurementValue data**

| Index | Content | Data type |
|---|---|---|
| 0 | Whether the OUT1 measurements are valid or invalid | VT_UI1 |
| 1 | Results of OUT1 intersection decisions | VT_UI1 |
| 2 | OUT1 measure | VT_R4 |
| ... | ... | ... |
| 45 | Whether the OUT16 measurements are valid or invalid | VT_UI1 |
| 46 | Results of OUT16 intersection decisions | VT_UI1 |
| 47 | OUT16 measure | VT_R4 |

Example

```
Dim retVal As Variant
Dim outVal As Single
retVal = caoCtrl.Execute("GetMeasurementValue")
outVal = retVal(47)     'Get measurement value of OUT16
```

### 3.4.2. CaoController::Execute("GetProfile") command

Gets the profile of the specified high-speed mode.

Syntax GetProfile (<byTargetBank>,<byPosMode>,<dwGetProfNo>,<byGetProfCnt>,<byErase>)

byTargetBank : Which of the active or deactivating planes you obtain from? (VT_UI1)

byPosMode : How to specify the profile acquisition position (VT_UI1)

dwGetProfNo : Profile No. to be acquired when 2 is specified by byPosMode. (VT_UI4)

byGetProfCnt : Number of profiles to read (VT_UI1)

byErase : Is the read profile deleted from the previous profile? (VT_UI1)

Return value : Profile data (VT_ARRAY|VT_VARIANT)

The data is stored as shown in Table 3-4.

**Table 3-4 GetProfile data**

| Index1 | Index2 | Content | Data type |
|---|---|---|---|
| 0 | 0 | Number of profiles stored in one unit of data | VT_UI1 |
| | 1 | ON/OFF of profile compression (time axis) | VT_UI1 |
| | 2 | Number of data points in one profile | VT_UI2 |
| | 3 | First X coordinate | VT_I4 |
| | 4 | Spacing of the data points in the X direction | VT_I4 |
| | 5 | Latest profile number at the time of acquisition | VT_UI4 |
| | 6 | Profile No. of the oldest profile held by the controller | VT_UI4 |
| | 7 | Profile No. of the oldest profile read | VT_UI4 |
| | 8 | Number of profiles read this time | VT_UI1 |
| 1 | 0~ | Profile data<br>The acquired profile data is stored in order.<br>For information on the number of data units and the order of data store, refer to Section 9.2.9.6 and 9.2.9.7 of the LJ-V7000 Series Communication Library Reference Manual. | VT_I4 |

Example

```
Dim retVal As Variant
Dim infoVal As Variant
Dim profVal As Variant
Dim data As Integer
retVal = caoCtrl.Execute("GetProfile", Array(0, 0, 0, 10, 0))
infoVal = retVal(0)
profVal = retVal(1)
data = profVal(infoVal(0) * infoVal(2) * 9) 'Get 1st data of 10th profile
```

### 3.4.3. CaoController::Execute("GetBatchProfile") command

Gets the profile of the batch measurement for the specified high-speed mode.

Syntax GetBatchProfile (<byTargetBank>,<byPosMode>,<byGetBatchNo>,
<dwGetProfNo>,<byGetProfCnt>,<byErase>)

byTargetBank : Which of the active or deactivating planes you obtain from? (VT_UI1)

byPosMode : How to specify the batch position

(VT_UI1)

byGetBatchNo : Profile No. to be acquired when 2 is specified by byPosMode. (VT_UI4)

dwGetProfNo : Acquisition start profile No. in the batch(VT_UI4)

byGetProfCnt : Number of profiles to read (VT_UI1)

byErase : Is the read profile deleted from the previous profile? (VT_UI1)

Return value : Profile data (VT_ARRAY|VT_VARIANT)
The data is stored as shown in Table 3-5.

**Table 3-5 GetBatchProfile data**

| Index1 | Index2 | Content | Data type |
|---|---|---|---|
| 0 | 0 | Number of profiles stored in one unit of data | VT_UI1 |
| | 1 | ON/OFF of profile compression (time axis) | VT_UI1 |
| | 2 | Number of data points in one profile | VT_UI2 |
| | 3 | First X coordinate | VT_I4 |
| | 4 | Spacing of the data points in the X direction | VT_I4 |
| | 5 | Latest batch number at the time of acquisition | VT_UI4 |
| | 6 | Number of profiles in the latest batch | VT_UI4 |
| | 7 | The oldest batch number held by the controller | VT_UI4 |
| | 8 | Number of profiles in the oldest batch held by the controller | VT_UI4 |
| | 9 | Batch number read this time | VT_UI4 |
| | 10 | Number of profiles in the batch read this time | VT_UI4 |
| | 11 | What is the oldest profile in the batch you read? | VT_UI4 |
| | 12 | Number of profiles read this time | VT_UI1 |
| | 13 | Is batch measurement of the latest batch number completed? | VT_UI1 |
| 1 | 0~ | Profile data<br>The acquired profile data is stored in order.<br>For information on the number of data units and the order of data store, refer to Section 9.2.9.6 and 9.2.9.7 of the LJ-V7000 Series Communication Library Reference Manual. | VT_I4 |

Example

```
Dim retVal As Variant
Dim infoVal As Variant
```

```
Dim profVal As Variant
Dim data As Integer
retVal = caoCtrl.Execute("GetBatchProfile", Array(0, 0, 0, 0, 10, 0))
infoVal = retVal(0)
profVal = retVal(1)
data = profVal(infoVal(0) * infoVal(2) * 9) 'Get 1st data of 10th profile
```

### 3.4.4. CaoController::Execute("GetProfileAdvance") command

Gets the profile of the specified high-function mode.

Syntax GetProfileAdvance ()

Argument : None

Return value : Profile data (VT_ARRAY|VT_VARIANT)

The data is stored as shown in Table 3-6.

**Table 3-6 GetProfileAdvance data**

| Index1 | Index2 | Content | Data type |
|---|---|---|---|
| 0 | 0 | Number of profiles stored in one unit of data | VT_UI1 |
| | 1 | ON/OFF of profile compression (time axis) | VT_UI1 |
| | 2 | Number of data points in one profile | VT_UI2 |
| | 3 | First X coordinate | VT_I4 |
| | 4 | Spacing of the data points in the X direction | VT_I4 |
| 1 | 0~ | Profile data<br>The acquired profile data is stored in order. For information on the number of data units and the order of data store, refer to Section 9.2.9.6 and 9.2.9.7 of the LJ-V7000 Series Communication Library Reference Manual. | VT_I4 |
| 2 | 0~47 | OUT measurement data<br>It is stored in the same data order as Table 3-3. | VT_UI1 or VT_R4 |

Example

```
Dim retVal As Variant
Dim infoVal As Variant
Dim outVal As Variant
Dim data As Integer
retVal = caoCtrl.Execute("GetProfileAdvance")
infoVal = retVal(0)
outVal = retVal(2)
data = outVal(3 * 4 + 2) 'Get measurement value of OUT5
```

### 3.4.5. CaoController::Execute("GetBatchProfileAdvance") command

Gets the profile of the batch measurement for the specified high-function mode.

Syntax GetBatchProfileAdvance (<byPosMode>,<byGetBatchNo>,<dwGetProfNo>,<byGetProfCnt>)

| | | |
|---|---|---|
| byPosMode | : | How to specify the batch position<br>(VT_UI1) |
| byGetBatchNo | : | Profile No. to be acquired when 2 is specified by byPosMode.<br>(VT_UI4) |
| dwGetProfNo | : | Acquisition start profile No. in the batch(VT_UI4) |
| byGetProfCnt | : | Number of profiles to read (VT_UI1) |
| Return value | : | Profile data (VT_ARRAY\|VT_VARIANT)<br>The data is stored as shown in Table 3-7. |

**Table 3-7 GetBatchProfileAdvance data**

| Index1 | Index2 | Content | Data type |
|---|---|---|---|
| 0 | 0 | Number of profiles stored in one unit of data | VT_UI1 |
| | 1 | ON/OFF of profile compression (time axis) | VT_UI1 |
| | 2 | Number of data points in one profile | VT_UI2 |
| | 3 | First X coordinate | VT_I4 |
| | 4 | Spacing of the data points in the X direction | VT_I4 |
| | 5 | Batch number read this time | VT_UI4 |
| | 6 | Number of profiles in the batch read this time | VT_UI4 |
| | 7 | What is the oldest profile in the batch you read? | VT_UI4 |
| | 8 | Number of profiles read this time | VT_UI1 |
| 1 | 0~ | Profile data<br>The acquired profile data is stored in order.<br>For information on the number of data units and the order of data store, refer to Section 9.2.9.6 and 9.2.9.7 of the LJ-V7000 Series Communication Library Reference Manual. | VT_I4 |
| 2 | 0~ | OUT measurement data for each profile<br>The data of the following number of elements is stored<br><16 OUT's worth of data (in the same data order as in Table 3-3)>×<number of profiles read> | VT_UI1 or<br>VT_R4 |
| 3 | 0~47 | OUT measurement data for the batch to be acquired<br>It is stored in the same data order as Table 3-3. | VT_UI1 or<br>VT_R4 |

| | | | |
|---|---|---|---|
| 4 | 0~47 | Latest OUT measurement data at the time of command execution<br>It is stored in the same data order as Table 3-3. | VT_UI1 or VT_R4 |

Example

```
Dim retVal As Variant
Dim infoVal As Variant
Dim outVal As Variant
Dim data As Integer
retVal = caoCtrl.Execute("GetBatchProfileAdvance", Array(0, 0, 0, 10))
infoVal = retVal(0)
outVal = retVal(2)
data = outVal(48 * 9 + 3 * 4 + 2) 'Get measurement value of OUT5 of 10th profile
```

## 3.5. Storage function related

### 3.5.1. CaoController::Execute("StartStorage") command

Start storage.

Syntax StartStorage()

Argument : None

Return value : None

Example

```
caoCtrl.Execute "StartStorage"
```

### 3.5.2. CaoController::Execute("StopStorage") command

Storage ends.

Syntax StopStorage()

Argument : None

Return value : None

Example

```
caoCtrl.Execute "StopStorage"
```

### 3.5.3. CaoController::Execute("GetStorageStatus") command

Gets the status of the storage.

Syntax GetStorageStatus (<dwReadArea>)

| | | |
|---|---|---|
| dwReadArea | : | Read storage plane<br>(VT_UI4) |
| Return value | : | Storage information (VT_ARRAY\|VT_VARIANT)<br>The data is stored as shown in Table 3-8. |

**Table 3-8 GetStorageStatus data**

| Index | Content | Data type |
|---|---|---|
| 0 | Free storage | VT_UI1 |
| 1 | Program No. of the target storage plane | VT_UI1 |
| 2 | Storage object | VT_UI1 |
| 3 | Number of storage points | VT_UI4 |
| 4 | Number of storage planes | VT_UI4 |
| 5 | No. of the active storage surface | VT_UI4 |

Example

```
Dim retVal As Variant
Dim statusVal As Byte
retVal = caoCtrl.Execute("GetStorage", 0)
statusVal = retVal(2)   'Get storage target
```

### 3.5.4. CaoController::Execute("GetStorageData") command

Obtain the OUT measurement data of the storage.

Syntax GetStorageData (<dwSurface >,< dwStartNo >,< dwDataCnt >)

| | | |
|---|---|---|
| dwSurface | : | Read storage plane (VT_UI4) |
| dwStartNo | : | Data No. to start reading(VT_UI4) |
| dwDataCnt | : | Number of data points to be read (VT_UI4) |
| Return value | : | Storage-OUT Measurement data (VT_ARRAY\|VT_VARIANT) |

The data is stored as shown in Table 3-9.

**Table 3-9 GetStorageData data**

| Index1 | Index2 | Content | Data type |
|---|---|---|---|
| 0 | 0 | Free storage | VT_UI1 |
| | 1 | Program No. of the target storage plane | VT_UI1 |
| | 2 | Storage object | VT_UI1 |
| | 3 | Number of storage points | VT_UI4 |
| | 4 | Data number from which reading started | VT_UI4 |
| | 5 | Number of data read this time | VT_UI4 |
| | 6 | Standard time | VT_DATE |
| 1 | 0~ | Value counted in units of 10 ms from the reference time of each data<br>The number of read data points and the counter value are stored. | VT_UI4 |
| 2 | 0~ | Storage OUT measurement data<br>The data of the following number of elements is stored<br><16 OUT's worth of data (in the same data order as in Table 3-3)>×<the number of data points read> | VT_UI1 or<br>VT_R4 |

Example

```
Dim retVal As Variant
Dim outVal As Variant
retVal = caoCtrl.Execute("GetStorageData", Array(0, 0, 10))
outVal = retVal(48 * 6 + 3 * 1 + 1) 'Get judge result of OUT2 of 7th profile
```

### 3.5.5. CaoController::Execute("GetStorageProfile") command

Retrieves the storage profile data.

Syntax GetStorageProfile (<dwSurface >,< dwStartNo >,< dwDataCnt >)

| | | |
|---|---|---|
| dwSurface | : | Read storage plane (VT_UI4) |
| dwStartNo | : | Data No. to start reading(VT_UI4) |
| dwDataCnt | : | Number of data points to be read (VT_UI4) |
| Return value | : | Storage profile data (VT_ARRAY\|VT_VARIANT)<br>The data is stored as shown in Table 3-10. |

**Table 3-10 GetStorageProfile data**

| Index1 | Index2 | Content | Data type |
|---|---|---|---|
| 0 | 0 | Free storage | VT_UI1 |
| | 1 | Program No. of the target storage plane | VT_UI1 |
| | 2 | Storage object | VT_UI1 |
| | 3 | Number of storage points | VT_UI4 |
| | 4 | Data number from which reading started | VT_UI4 |
| | 5 | Number of data read this time | VT_UI4 |
| | 6 | Standard time | VT_DATE |
| | 7 | Number of profiles stored in one unit of data | VT_UI1 |
| | 8 | ON/OFF of profile compression (time axis) | VT_UI1 |
| | 9 | Number of data points in one profile | VT_UI2 |
| | 10 | First X coordinate | VT_I4 |
| | 11 | Spacing of the data points in the X direction | VT_I4 |
| 1 | 0~ | Value counted in units of 10 ms from the reference time of each data<br>The number of read data points and the counter value are stored. | VT_UI4 |
| 2 | 0~ | Latest OUT measurement results at the time of profile acquisition<br>The data of the following number of elements is stored<br><16 OUT's worth of data (in the same data order as in Table 3-3)>×<the number of data points read> | VT_UI1 or VT_R4 |
| 3 | 0~ | Profile data<br>The acquired profile data is stored in order.<br>For information on the number of data units and the order of data store, refer to Section 9.2.9.6 and 9.2.9.7 of the LJ-V7000 Series Communication Library Reference Manual. | VT_I4 |
| 4 | 0~ | OUT measurement results for each profile<br>The data of the following number of elements is stored<br><16 OUT's worth of data (in the same data order as in Table 3-3)>×<the number of data points read> | VT_UI1 or VT_R4 |

Example

```
Dim retVal As Variant
Dim infoVal As Variant
Dim profVal As Variant
Dim data As Integer
retVal = caoCtrl.Execute("GetStorageProfile", Array(0, 0, 10))
infoVal = retVal(0)
profVal = retVal(1)
data = profVal(infoVal(7) * infoVal(9) * 9 + 1) 'Get 2nd data of 10th profile
```

### 3.5.6. CaoController::Execute("GetStorageBatchProfile") command

Obtain the OUT measurement data of the storage.

Syntax GetStorageBatchProfile (<dwSurface >,< dwGetBatchNo >,< dwGetBatchTopProfNo >, < byGetProfCnt >)

| | | |
|---|---|---|
| dwSurface | : | Read storage plane (VT_UI4) |
| dwGetBatchNo | : | Batch number to be read(VT_UI4) |
| dwGetBatchTopProfNo | : | Acquisition start profile No. in the batch(VT_UI4) |
| byGetProfCnt | | Number of profiles to read (VT_UI1) |
| Return value | : | OUT measurement data (VT_ARRAY\|VT_VARIANT)<br>The data is stored as shown in Table 3-11. |

**Table 3-11 GetStorageBatchProfile data**

| Index1 | Index2 | Content | Data type |
|---|---|---|---|
| 0 | 0 | Free storage | VT_UI1 |
| | 1 | Program No. of the target storage plane | VT_UI1 |
| | 2 | Storage object | VT_UI1 |
| | 3 | Number of storage points | VT_UI4 |
| | 4 | Batch number read this time | VT_UI4 |
| | 5 | Number of profiles in the batch read this time | VT_UI4 |
| | 6 | What is the oldest profile in the batch you read? | VT_UI4 |
| | 7 | Number of profiles read this time | VT_UI1 |
| | 8 | Standard time | VT_DATE |
| | 9 | Number of profiles stored in one unit of data | VT_UI1 |
| | 10 | ON/OFF of profile compression (time axis) | VT_UI1 |
| | 11 | Number of data points in one profile | VT_UI2 |
| | 12 | First X coordinate | VT_I4 |
| | 13 | Spacing of the data points in the X direction | VT_I4 |
| 1 | 0~ | Profile data<br>The acquired profile data is stored in order.<br>For information on the number of data units and the order of data store, refer to Section 9.2.9.6 and 9.2.9.7 of the LJ-V7000 Series Communication Library Reference Manual. | VT_UI4 |
| 2 | 0~ | OUT measurement results for each profile<br>The data of the following number of elements is stored<br><16 OUT's worth of data (in the same data order as in Table 3-3)>×<the number of data points read> | VT_UI1 or VT_R4 |

| | | | |
|---|---|---|---|
| 3 | - | Value counted in units of 10 ms from the reference time of each data | VT_UI4 |
| 4 | 0~47 | OUT measurement results for batches<br>It is stored in the same data order as Table 3-3. | VT_UI1 or<br>VT_R4 |

Example

```
Dim retVal As Variant
Dim infoVal As Variant
Dim profVal As Variant
Dim data As Integer
retVal = caoCtrl.Execute("GetStorageProfile", Array(0, 0, 10))
infoVal = retVal(0)
profVal = retVal(1)
data = profVal(infoVal(9) * infoVal(11) * 9 + 1) 'Get 2nd data of 10th profile
```

## 3.6. High-speed data communications

### 3.6.1. CaoController::Execute("StartHighSpeedDataCommunication") command

Start high-speed data communication. This command executes the following three commands in the LJ-V7000 Communication Library.

1. HighSpeedDataEthernetCommunicationInitialize
2. PreStartHighSpeedDataCommunication
3. StartHighSpeedDataCommunication

Syntax StartHighSpeedDataCommunication (<wHighSpeedPortNo >,< dwProfileCnt >)

| | | |
|---|---|---|
| wHighSpeedPortNo | : | Port number (VT_UI2) used for high-speed data communication |
| dwProfileCnt | : | Number of Profiles to be sent collectively (VT_UI4)<br>If you receive the specified number of profiles, OnMessage events occur. |
| Return value | : | None |

Example

```
caoCtrl.Execute "StartHighSpeedDataCommunication", Array(24692, 10)
```

### 3.6.2. CaoController::Execute("StopHighSpeedDataCommunication") command

Start high-speed data communication. This command executes the following two commands in the LJ-V7000 Communication Library.

1. StopHighSpeedDataCommunication
2. HighSpeedDataCommunicationFinalize

Syntax StopHighSpeedDataCommunication ()

Argument : None

Return value : None

Example

```
caoCtrl.Execute "StopHighSpeedDataCommunication"
```