

# AIO Provider CONTEC AIO Board

Version 1.0.6

## User's Guide

December 21, 2021

[Remarks]

**[Revision history]**

Version	Date	Content
1.0.0.0	2011-7-12	First edition.
1.0.1.0	2012-5-29	Meta mode was added.
1.0.1	2012-7-17	Version rule of the document was changed.
1.0.2	2013-2-7	Variables for setting /obtaining analogue output range was changed. Error codes returned by AIO API were changed
1.0.3	2014-4-10	Added an Addcontroller option
1.0.4	2017-6-7-	Analog input sampling function added.
1.0.5	2017-9-5	Added OnMessage support for multi-process analog input sampling function.
	2017-10-17	Error correction
1.0.6	2021-12-21	Fixed false positives for errors.

**[Hardware]**

Model	Version	Notes
AIO-160802L-LPE		
ADI16-4(FIT)		Connection to I/O control module equipped with USB (CPU-CA10 (USB)) required.
DAI16-4(FIT)		Connection to I/O control module equipped with USB (CPU-CA10 (USB)) required.
ADI12-16(PCI)		
AIO121601E3-PE		
AIO121601M-PCI		
ADA-16-32/2(PCI)F		
ADI16-4(USB)		
AIO-121602AH-PCI		
AIO-160802AY-USB		
AIO-163202F-PE		

## Contents

1. Introduction.....	4
2. Outline of Provider.....	5
2.1. Outline .....	5
2.2. Methods and properties .....	5
2.2.1. CaoWorkspace::AddController method .....	5
2.2.2. CaoController::Execute method .....	11
2.2.3. CaoController::OnMessage event.....	11
2.2.4. CaoController::AddVariable method .....	12
2.2.5. CaoController::get_VariableNames property.....	12
2.2.6. CaoVariable::get_Value property.....	12
2.2.7. CaoVariable::put_Value property .....	12
2.3.2. CaoController::Execute Command details .....	15
2.4. Variable list .....	33
2.4.1. Controller class.....	33
2.5. Error code .....	35
2.6. CAO-AIO API reference table .....	36
3. Sample Program .....	39
3.1. Specified analog input channel AD conversion data acquisition sample (simple AI input function).....	39
3.2. Digital Input State Change Event Reception Sample .....	39
3.3. Analog input sampling frequency storage event reception sample (high-function AI input function) ...	40
3.4. Analog Input Automatic Sampling Sample (Advanced AI Input Function).....	41

## 1. Introduction

This document is a user's guide of the AIO provider which is used to access CONTEC AIO board. Refer to CONTEC API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB) for details.

NOTE: The AIO device driver of the AIO board needs to be installed to use the AIO provider.  
Install the driver from API-PAC (W32) for PCI board or from API-USBP (WDM) for  
USB. After installing it, register the provider in the registry with reference to Table 2-1.

## 2. Outline of Provider

### 2.1. Outline

The AIO provider executes CONTEC API corresponding to CAO API at the time the CAO API is executed. Refer to Table 2-6 for CAO API and corresponding CONTEC API.

**Table 2-1 AIO provider**

File name	CaoProvAIO.dll
ProgID	CaoProv.CONTEC.AIO
Registry registration <sup>1</sup>	regsvr32 CaoProvAIO.dll
Remove registry registration	regsvr32 /u CaoProvAIO.dll

### 2.2. Methods and properties

#### 2.2.1. CaoWorkspace::AddController method

The AIO provider establishes (opens) connection to the AIO board when the Controller object is created.

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

bstrCtrlName : [in] Controller name  
 bstrProvName : [in] Provider name (Fixed to "CaoProv.CONTEC.AIO")  
 bstrPcName : [in] Provider execution machine name  
 bstrOption : [in] Option character string

The machine name can be an empty string.

Following is a list of option string items.

**Table 2-2 Option character string of CaoWorkspace::AddController**

Option	Meaning
DeviceName=[<Device name>]	Device name of the board to be connected <sup>*1</sup> Default: "" (no value specified) For the case of "" (no value specified), connection to the first detected available device will be established. Note: Specify the device supporting the AIO board ID. <sup>*1</sup>

<sup>1</sup> The AIO board driver must be installed to register the AIO provider.

ScanCount=[<Number of retries>]	<p>Number of retries (Range: 0 to 32767)</p> <p>Default: 4 times</p> <p>Number of retries performed when connection to the detected device fails for the case DeviceName option is set to "" (no value specified).</p>
Interval=[<Sampling interval>] <sup>*2</sup>	<p>Sampling interval (Range: 0 to 65535)</p> <p>Default: 0 (off)</p> <p>Specify the sampling interval (ms) to acquire OnMessage event when the digital input<sup>*1</sup> byte changes.</p>
Mask=[<Mask value>] <sup>*2</sup>	<p>Mask value (Range: 0 to 255)</p> <p>Default: 255 (no mask)</p> <p>Reduce occurrence of unnecessary events by masking the input byte when the Interval option is enabled.</p>
Coexistence=[<coexistence>]	<p>Co-existence setting.</p> <p>When this option is disabled (False), if a controller object connects to any device where other process has already connected, an error<sup>*1</sup> will occur and the connection will fail.</p> <p>False: Disabled</p> <p>True: Enabled (default setting)</p>
ResetDevice=[<Device Reset>]	<p>Resetting the device, initializing the driver</p> <p>False: Disabled</p> <p>True: Enabled (default setting)</p> <p>Note: All parameters in the driver will be returned to their initial values. Note: All parameters in the driver will be returned to their initial values. If the device is running, it will be forced to stop.<sup>*1</sup></p> <p>"Disabled" is recommended for multi-process environments.</p>
AiInputMethod=[< AI input method>]	<p>Analog input method<sup>*3</sup></p> <p>0: Single-ended input</p> <p>1: Differential Input</p> <p>When omitted (device initial value):</p> <p>Varies depending on device type and jumper (JP) setting<sup>*1</sup></p>

AiChannels=[< Number of AI channels used AI input method >]	Number of analog input channels used <sup>*3</sup> Range: 1 to Maximum number of channels When omitted (device initial value): 1 <sup>*1</sup>
AiRangeAll=[< AI Range All Channel Setting >]	Analog Input Range All Channel Setting <sup>*4</sup> Setting range: The range that can be set depends on the device. <sup>*1</sup> When omitted (device initial value): It depends on the type of device. <sup>*1</sup> Reference: This option can also be set in the controller class system variable "@RANGE_AI".
AiMemoryType=[< AI memory format >]	Memory format setting for analog input data storage <sup>*3</sup> 0: FIFO 1: RING When omitted (device initial value): FIFO <sup>*1</sup>  Note: This provider supports only the "device buffer mode," which is the initial value of the device startup, for the conversion data transfer method, so the memory format for the "user buffer mode" cannot be set.
AiClockType=[< AI clock types >]	Sets the type of analog input data clock. <sup>*1*3</sup> 0: Internal clock 1: Eternal clock 10: Event controller output 20: CH0 Comparison count match 0 (Only AIO-121601M-PCI can be set.) 21: CH1 Comparison count match 0 (Only AIO-121601M-PCI can be set.) 22: CH0 Comparison count match 1 (Only AIO-121601M-PCI can be set.) 23: CH1 Comparison count match 1(Only AIO-121601M-PCI can be set.) 24: CH0 Count Up (Only AIO-121601M-PCI can be set.) 25: CH1 Count Up (Only AIO-121601M-PCI can be set.) 26: CH0 countdown (Only AIO-121601M-PCI can be set.) 27: CH1 countdown (Only AIO-121601M-PCI can be set.)

	<p>28: CH0 Counter Clear (Only AIO-121601M-PCI can be set.)                  29: CH1 Counter Clear (Only AIO-121601M-PCI can be set.)                  30: Carrie Borough (Only AIO-121601M-PCI can be set.)                  31: Timer (Only AIO-121601M-PCI can be set.)                  When omitted (device initial value): Internal clock*<sup>1</sup></p> <p>[Description]</p> <ul style="list-style-type: none"> <li>• When using the internal clock as the clock, the sampling speed can be specified with the "AiSamplingClock" option.*<sup>1</sup></li> <li>• If you want to use the event controller output as a clock, you need to execute the "SetEcuSignal" command.*<sup>1</sup></li> </ul> <p>-When using the multifunction device AIO-121601M-PCI -</p> <ul style="list-style-type: none"> <li>• If you want to set the clock to match the comparison count, you need to execute the "SetCntmNotifyCountUp" command.*<sup>1</sup></li> <li>• If you want to set the clock to count up or count down, you need to execute "StartCntmCount".*<sup>1</sup></li> <li>• If you want to set the clock to count clear, you need to execute the "SetCntmZeroClearCount" command.*<sup>1</sup></li> <li>• To set the clock to CarryBorrow, you need to execute the "SetCntmNotifyCarryBorrow" command.*<sup>1</sup></li> <li>• To set the clock to a timer, you need to execute the "SetCntmNotifyTimer" command.*<sup>1</sup></li> </ul>
<p>AiSamplingClock=[&lt; AI conversion speed &gt;]</p>	<p>Analog input Data conversion speed setting when using internal clock*<sup>3</sup></p> <p>Unit: μsec</p> <p>Setting range: The range that can be set depends on the device.*<sup>1</sup></p> <p>When omitted (device initial value): Varies by device*<sup>1</sup></p> <p>*The conversion speed can be specified in "AiClockType" when the internal clock is set. If the internal clock is not used, it is not necessary to specify the conversion speed.</p>
<p>AiClockEdge=[&lt; AI input timing&gt;]</p>	<p>Analog Input Input timing setting when using external clock</p> <p>0: trailing edge                  1: rising edge                  Default: 0(default)</p>



	<p>*In "AiClockType", it is possible to specify whether to use the falling edge or the rising edge as the input timing when the external clock is set. If you do not use an external clock, you do not need to specify this option.</p>
<p>AiAutoSampling=          &lt;Start&gt;[:&lt;SamplingTimes&gt;][:&lt;Event&gt;[:&lt;DataType&gt;]]</p>	<p>Analog input automatic sampling conversion start setting          Set whether or not to automatically start the AD conversion of the sampling data immediately after the connection by AddController under the conditions of each parameter.          Square brackets (" [ ]") indicate optional.</p> <p>&lt;Start&gt;          1: Start automatic conversion          0: Do not start automatic conversion          Default: 0(default)</p> <p>The following parameters are valid only when the &lt;Start&gt; value is "1".</p> <p>&lt;SamplingTimes&gt;          Set the specified sampling frequency value.          Internally, it executes the "SetAiEventSamplingTimes" command.          Default: No execution(default)</p> <p>&lt;Event&gt;          Sets the event factor of CaoController::OnMesseage in decimal or hexadecimal.          Example) Specified sampling frequency storage event              In hexadecimal notation: "0x80" or "0X80".              In decimal notation: "128".          Internally, the "SetAiEventConditions" command (see p. 23) is executed.          Default: No execution(default)          Note: Event factor: 0 means no execution.</p>

	<p>&lt;DataType&gt;</p> <p>Valid when &lt;Event&gt; value is non-zero.</p> <p>Sets the conversion data type when the CaoController::OnMesseage event occurs. Internally, the "SetAiEventConditions" command (see P23) is executed.</p> <p>Default: 0(Voltage or current value)</p> <p>*The sampling conversion start/stop conditions for this option are fixed as follows.</p> <ul style="list-style-type: none"> <li>• Conversion start condition: Software</li> <li>• Conversion stop condition: Command</li> </ul>
--	---

\*1: Refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB) for details.

\*2: This is enabled only on the models equipped with digital input. Refer to CONTEC product manual for details.

\*3: This function can be used only with models equipped with analog input. For details, please refer to the CONTEC product manual.


\*4: This function is available only for models equipped with analog input and capable of range setting by function execution. For details, please refer to the CONTEC product manual.

\*5: This function can be used only with models equipped with analog output. For details, please refer to the CONTEC product manual.

\*6: This function is available only for the models with analog output and the models that can set the range by function execution. For details, please refer to the CONTEC product manual.

### 2.2.2. CaoController::Execute method

Refer to **Table 2-3** for details of available command names and parameters.

 Execute( < bstrCommand:BSTRT > [,<vntParam:VARIANT>[,< pVal:VARIANT>]] )

bstrCommand : [in] Command name

vntParam : [in] Parameters

pVal : [out] Acquisition data

### 2.2.3. CaoController::OnMessage event

Exchanges data with a client as an OnMessage event of CaoController class. At this time, received data is stored directly in Message::Value property.

Event number	Meaning	Value
1	For the case the Interval option is set to other than off (0), this event occurs when the bit, which is not masked by the Mask option, changes at the time the digital input <sup>*1*2</sup> byte changes.	Digital input byte Type: VT_I2
100	Analog input AD conversion start condition satisfied <sup>*1*3*4</sup> .	None
101	Analog Input Repeat End <sup>*1*3*4</sup> .	Number of repeats Type: VT_I4
102	End of analog input device operation <sup>*1*3*4</sup> .	Analog input sampling conversion data value The data type is determined by <DataTpe> when the "SetAiEventConditions" command is executed <ul style="list-style-type: none"> <li>• At voltage or current value: Type VT_ARRAY   VT_R4</li> <li>• When a binary value : Type VT_ARRAY   VT_I4</li> <li>• When no data is acquired (event notification only): Type VT_EMPTY</li> </ul> *When the sampling frequency is zero, VT_EMPTY is used.  Example: Number of channels used is '8' and sampling frequency is 'n'.  ----- Element No   Sampling data ----- 0   VT_R4: CH0 converted data 1   VT_R4: CH1 converted data 2   VT_R4: CH2 converted data
103	Analog input specified sampling frequency storage <sup>*1*3*4</sup> . Get the specified sampling data for the number of channels used from the device memory. The conversion data is stored in the conversion data type ("binary value" or "voltage or current value") set by the "SetAiEventConditions" command.	
104	Analog input overflow <sup>*1*3*4</sup> .	
105	Analog input sampling clock error <sup>*1*3*4</sup> .	
106	Analog input AD conversion error <sup>*1*3*4</sup> .	

		3   VT_R4: CH3 converted data
		4   VT_R4: CH4 converted data
		5   VT_R4: CH5 converted data
		6   VT_R4: CH6 converted data
		7   VT_R4: CH7 converted data
		8   VT_R4: CH0 converted data
		:   :
		:   :
		n×8-1   VT_R4: CH7 converted data
		-----
		Note: Failed to get the conversion data value
		Error code storage Type: VT_I4

\*1: Refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB) for details.

\*2: This is available only on the models equipped with digital input. Refer to CONTEC product manual for details.

\*3: This function is available only for models equipped with analog input. For details, please refer to the CONTEC product manual.

\*4: To enable the event, you need to set the condition in advance using the "SetAiEventCondition" command.

#### 2.2.4. CaoController::AddVariable method

This method creates a variable object used to access the AIO board.

Only the variables given in 2.4.1 can be used. If a variable other than those is specified, this method will return an error.

**Syntax** AddVariable( <bstrName:BSTR > [, <bstrOption:BSTR>] )

bstrName : [in] Arbitrary name

bstrOption : [in] Option character string (not used)

#### 2.2.5. CaoController::get\_VariableNames property

Acquires the variable name list in 2.4.1.

#### 2.2.6. CaoVariable::get\_Value property

Acquires information corresponding to a variable. For the implementation status and acquired data of each variable, refer to 2.4.1.

#### 2.2.7. CaoVariable::put\_Value property

Configures information corresponding to a variable. For the implementation status and setting data of each variable, refer to 2.4.1.

## 2.3. Command list

### 2.3.1. Controller class

**Table 2-3 CaoConteroller::Execute Command list**

コマンド名	機能	頁
SetAiStartTrigger	Analog Input Sampling Start Condition Setting	P.15
SetAiStartLevel	Analog input sampling Level setting at the start of level comparison (binary)	P.16
SetAiStartLevelEx	Analog Input Sampling Level Setting at Start of Level Comparison (Voltage or Current)	P.16
SetAiStartInRange	Analog Input Sampling Level Setting at Start of In Range Comparison (Binary)	P.17
SetAiStartInRangeEx	Analog Input Sampling Level Setting at Start of In-range Comparison (Voltage or Current)	P.17
SetAiStartOutOfRange	Analog Input Sampling Level Setting at the Start of Outrange Comparison (Binary)	P.18
SetAiStartOutOfRangeEx	Analog Input Sampling Level Setting at the Start of Outrange Comparison (Voltage or Current)	P.18
SetAiStopTrigger	Analog Input Sampling Stop Condition Setting	P.19
SetAiStopTimes	Analog Input Sampling Frequency Setting	P.20
SetAiStopLevel	Analog Input Sampling Level Setting at Start of Level Comparison (Binary)	P.20
SetAiStopLevelEx	Analog Input Sampling Level Setting at Start of Level Comparison (Voltage or Current)	P.20
SetAiStopInRange	Analog Input Sampling Level Setting at Start of In Range Comparison (Binary)	P.21
SetAiStopInRangeEx	Analog Input Sampling Level Setting at Start of In-range Comparison (Voltage or Current)	P.21
SetAiStopOutOfRange	Analog Input Sampling Level Setting at the Start of Outrange Comparison (Binary)	P.22
SetAiStopOutOfRangeEx	Analog Input Sampling Level Setting at Start of Outrange Comparison (Voltage or Current)	P.22
SetAiStopDelayTimes	Analog Input Sampling Stop Delay Count Setting	P.23
SetAiRepeatTimes	Analog Input Sampling Repeat Count Setting	P.23
SetAiEventConditions	Analog Input Setting of Event Occurrence Condition	P.23
SetAiEventSamplingTimes	Analog Input Sampling Sampling frequency setting when using specified sampling frequency storage event	P.24
StartAiSamplingAsync	Analog input sampling AD conversion start (asynchronous operation)	P.24

StartAiSamplingSync	Analog input sampling AD conversion start (synchronous operation)	P.25
StopAiSampling	Analog input sampling AD conversion stop	P.25
GetAiSamplingCount	Analog input Sampling count acquisition	P.25
GetAiStopTriggerCount	Analog input Sampling count when stop trigger is input	P.26
GetAiSamplingData	Analog input Read data for specified sampling (binary)	P.26
GetAiSamplingDataEx	Analog input Read the specified sampling data (voltage or current)	P.26
GetAiStatus	Get analog input status	P.27
GetAiResolution	Get analog input resolution	P.27
ResetAiMemory	Reset analog input device memory	P.27
ResetAiStatus	Reset analog input status	P.28
SetEcuSignal	Set event controller signals	P.28
SetCntmNotifyCountUp	Specify Notification by Counter Count Match and Set Comparison Register	P.28
SetCntmZeroClearCount	Clear Counter Count Value to Zero	P.29
SetCntmNotifyCarryBorrow	Setting Counter Carry/Borrow Notification	P.29
SetCntmNotifyTimer	Configuring Timer Notification	P.29
StartCntmCount	Start Counter Operation	P.29
StopCntmCount	Stop Counter Operation	P.30
PresetCntm	Presetting Counters	P.30
ReadCntmCount	Reading Counter Values	P.30
ReadCntmStatusEx	Reading Counter Status	P.30
SetCntmZMode	Configuring Counter Z-Phase Usage	P.31
SetCntmZLogic	Configuring Counter Z-Phase Logic	P.31
SetCntmCountDirection	Configuring Counter Counting Direction	P.31
SetCntmOperationMode	Configuring Counter Operation Mode	P.32
SetCntmDigitalFilter	Counter Digital Filter Settings	P.32

## 2.3.2. CaoController::Execute Command details

# SetAiStartTrigger

<b>Syntax</b>	<i>object.</i> SetAiStartTrigger ( <Trigger> )
<b>Arguments</b>	<p>&lt;Trigger&gt; = VT_I2: Conversion start condition *1</p> <p>Set the value from the following range. The value that can be set depends on the device.</p> <ul style="list-style-type: none"> <li>0: Software</li> <li>1: Rising edge of external trigger</li> <li>2: Falling edge of external trigger</li> <li>3: Level Comparison</li> <li>4: In-range comparison</li> <li>5: Outrange comparison</li> <li>10: Event controller output</li> <li>20: CH0 Comparison count match 0 (Only AIO-121601M-PCI can be set)</li> <li>21: CH1 Comparison count match 0 (Only AIO-121601M-PCI can be set)</li> <li>22: CH0 Comparison count match 1 (Only AIO-121601M-PCI can be set)</li> <li>23: CH1 Comparison count match 1 (Only AIO-121601M-PCI can be set)</li> <li>24: CH0 clearing a count (Only AIO-121601M-PCI can be set)</li> <li>25: CH1 clearing a count (Only AIO-121601M-PCI can be set)</li> <li>26: Carrie Borough (Only AIO-121601M-PCI can be set)</li> <li>27: Timer (Only AIO-121601M-PCI can be set)</li> </ul>
<b>Return value</b>	None
<b>Description</b>	<p>Set the analog input AD conversion sampling start condition.*1</p> <ul style="list-style-type: none"> <li>• When the conversion start condition is set to level comparison, use the "SetAiStartLevel" or "SetAiStartLevelEx" command to set the level comparison start.</li> <li>• When setting the conversion start condition to in-range comparison, use the "SetAiStartInRange" or "SetAiStartInRangeEx" command to set the start of in-range comparison.</li> <li>• When setting the conversion start condition to outrange comparison, use the "SetAiStartOutOfRange" or "SetAiStartOutOfRangeEx" command to set the outrange comparison start.</li> <li>• When using the event controller output as a clock, use the "SetEcuSignal" command to connect the event controller.</li> </ul> <p>[For users of the multifunction device AIO-121601M-PCI] *1</p> <ul style="list-style-type: none"> <li>• When setting the conversion start condition to a comparison count match When setting the conversion start condition to a comparison count match, use the "CntmNotifyCountUp" command to set the comparison count match occurrence condition.</li> <li>• When setting the conversion start condition to count clear To set the conversion start condition to carry-borrow, use the "CntmNotifyCarryBorrow" command to set the carry-borrow occurrence condition.</li> <li>• When setting the conversion start condition to carry-over To set the conversion start condition to carry-borrow, use the "CntmNotifyCarryBorrow" command to set the carry-borrow occurrence condition.</li> <li>• When setting the conversion start condition to a timer To set the conversion start condition to a timer, use the "CntmNotifyTimer" command to set the timer event occurrence condition.</li> </ul>

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB).

---

## SetAiStartLevel

---

<b>Syntax</b>	<i>object.</i> <b>SetAiStartLevel</b> ( <Channel>, <Level>, <Direction> )
<b>Arguments</b>	<p>&lt;Channel&gt; = VT_I2: AI channel number for level comparison*<sup>1</sup></p> <p>&lt;Level&gt; = VT_I4: Data for level comparison*<sup>1</sup>  This parameter is set as a binary value from the following range. The value that can be set differs depending on the device.  Devices with 12-bit resolution: 0 to 4095  Devices with 16-bit resolution: 0 to 65535</p> <p>&lt;Direction&gt; = VT_I2: Direction of level comparison*<sup>1</sup>  Set the value from the following range.  0: Both  1: Rising edge  2: Falling edge</p>
<b>Return value</b>	None
<b>Description</b>	<p>Sets the level at the start of level comparison during analog input AD conversion sampling as a binary value.*<sup>1</sup>  This setting is required when the conversion start condition is set to level comparison by the "SetAiStartTrigger" command.  * If the conversion start condition is other than level comparison, it is not necessary to execute this command.</p>

---

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## SetAiStartLevelEx

---

<b>Syntax</b>	<i>object.</i> <b>SetAiStartLevelEx</b> ( <Channel>, <Level>, <Direction> )
<b>Arguments</b>	<p>&lt;Channel&gt; = VT_I2: AI channel number for level comparison*<sup>1</sup></p> <p>&lt;Level&gt; = VT_R4: Data for level comparison*<sup>1</sup>  This parameter is set by voltage or current.</p> <p>&lt;Direction&gt; = VT_I2: Direction of level comparison*<sup>1</sup>  Set from the following range.  0: Both  1: Rising edge  2: Falling</p>
<b>Return value</b>	None
<b>Description</b>	<p>Set the level at the start of level comparison at the time of analog input AD conversion sampling by voltage or current.*<sup>1</sup>  This setting is required when the conversion start condition is set to level comparison by the "SetAiStartTrigger" command.  *If the conversion start condition is other than level comparison, it is not necessary to execute this command.</p>

---

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---



---

## SetAiStartInRange

---

<b>Syntax</b>	<i>object.</i> SetAiStartInRange( <Channel>, <Level1>, <Level2>, <StateTimes> )
<b>Arguments</b>	<p>&lt;Channel&gt; = VT_I2: AI channel number for in-range comparison*<sup>1</sup></p> <p>&lt;Level1&gt;, &lt;Level2&gt; = VT_I4: The range for in-range comparison. *<sup>1</sup>  Set the range in binary values from the following range. The range that can be set depends on the device.  Devices with 12-bit resolution: 0 to 4095  Devices with 16-bit resolution: 0 to 65535</p> <p>&lt;StateTimes&gt; = VT_I4: State retention times*<sup>1</sup>  After the in-range comparison is established, the period during which the state is held is set by the number of sampling times.  The actual conversion starts after the sampling time of StateTimes has elapsed.</p>
<b>Return value</b>	None
<b>Description</b>	<p>Sets the level at the start of in-range comparison when sampling analog input AD conversion by a binary value. *<sup>1</sup>  This setting is required when the conversion start condition is set to level comparison by the "SetAiStartTrigger" command..  *If the conversion start condition is other than level comparison, it is not necessary to execute this command..</p>

---

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---



---

## SetAiStartInRangeEx

---

<b>Syntax</b>	<i>object.</i> SetAiStartInRangeEx( <Channel>, <Level1>, <Level2>, <StateTimes> )
<b>Arguments</b>	<p>&lt;Channel&gt; = VT_I2: AI channel number for in-range comparison*<sup>1</sup></p> <p>&lt;Level1&gt;, &lt;Level2&gt; = VT_R4: The range for in-range comparison. *<sup>1</sup>  These are set by voltage or current.</p> <p>&lt;StateTimes&gt; = VT_I4: State retention count*<sup>1</sup>  After the in-range comparison is established, the period during which the state is held is set by the number of sampling times.  The actual conversion starts after the sampling time of StateTimes has elapsed.</p>
<b>Return value</b>	None
<b>Description</b>	<p>Sets the level at the start of in-range comparison when sampling analog input AD conversion by voltage or current. *<sup>1</sup>  This setting is required when the conversion start condition is set to level comparison by the "SetAiStartTrigger" command..  *If the conversion start condition is other than level comparison, it is not necessary to execute this command..</p>

---

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

---

## SetAiStartOutOfRange

---

<b>Syntax</b>	<i>object.</i> SetAiStartOutOfRange( <Channel>, <Level1>, <Level2>, <StateTimes> )
<b>Arguments</b>	<p>&lt;Channel&gt; = VT_I4: AI channel number to perform outrange comparison*<sup>1</sup></p> <p>&lt;Level1&gt;, &lt;Level2&gt; = VT_I4: Range for outrange comparison*<sup>1</sup>          These are binary values that can be set from the following ranges. The range that can be set depends on the device.          Devices with 12-bit resolution: 0 to 4095          Devices with 16-bit resolution: 0 to 65535</p> <p>&lt;StateTimes&gt; = VT_I4: State retention times*<sup>1</sup>          This parameter is used to set the number of sampling times to hold the state after an out-of-range comparison is made.          The actual conversion starts after the sampling time of StateTimes has elapsed.</p>
<b>Return value</b>	None
<b>Description</b>	<p>Sets the level at the start of outrange comparison when sampling analog input AD conversion by a binary value.*<sup>1</sup>          This setting is required when the conversion start condition is set to outrange comparison by the "SetAiStartTrigger" command.          * If the conversion start condition is other than outrange comparison, it is not necessary to execute.</p>

---

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## SetAiStartOutOfRangeEx

---

<b>Syntax</b>	<i>object.</i> SetAiStartOutOfRangeEx( <Channel>, <Level1>, <Level2>, <StateTimes> )
<b>Arguments</b>	<p>&lt;Channel&gt; = VT_I4: AI channel number for outrange comparison *<sup>1</sup></p> <p>&lt;Level1&gt;, &lt;Level2&gt; = VT_R4: Range for outrange comparison *<sup>1</sup>          Set by voltage or current.</p> <p>&lt;StateTimes&gt; = VT_I4: State retention count*<sup>1</sup>          Set the period of time to hold the state after the outrange comparison is established by the number of sampling.          The actual conversion starts after the sampling time of StateTimes has elapsed.</p>
<b>Return value</b>	None
<b>Description</b>	<p>Sets the level at the start of outrange comparison when sampling analog input AD conversion by voltage or current.*<sup>1</sup>          This setting is required when the conversion start condition is set to outrange comparison by the "SetAiStartTrigger" command.          *If the conversion start condition is other than outrange comparison, it is not necessary to execute.</p>

---

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

# SetAiStopTrigger

<b>Syntax</b>	<b><i>object.</i> SetAiStopTrigger ( &lt;Trigger&gt; )</b>
<b>Arguments</b>	<p>&lt;Trigger&gt; = VT_I2: Conversion stop condition*<sup>1</sup></p> <p>Set the value from the following range. The value that can be set depends on the device.</p> <ul style="list-style-type: none"> <li>0: End of conversion set number of times</li> <li>1: External trigger rising edge</li> <li>2: Falling edge of external trigger</li> <li>3: Level comparison</li> <li>4: Command ("StopAiSampling")</li> <li>5: In-range comparison</li> <li>6: Outrange comparison</li> <li>10: Event controller output</li> <li>20: CH0 Comparison count match 0 (Only AIO-121601M-PCI can be set.)</li> <li>21: CH1 Comparison count match 0 (Only AIO-121601M-PCI can be set.)</li> <li>22: CH0 Comparison count match 1 (Only AIO-121601M-PCI can be set.)</li> <li>23: CH1 Comparison count match 1 (Only AIO-121601M-PCI can be set.)</li> <li>24: CH0 clearing a count (Only AIO-121601M-PCI can be set.)</li> <li>25: CH1 clearing a count (Only AIO-121601M-PCI can be set.)</li> <li>26: Digital filter error (Only AIO-121601M-PCI can be set.)</li> <li>27: Abnormal input error (Only AIO-121601M-PCI can be set.)</li> <li>28: Carrie Borough (Only AIO-121601M-PCI can be set.)</li> <li>29: Timer (Only AIO-121601M-PCI can be set.)</li> </ul>
<b>Return value</b>	None
<b>Description</b>	<p>Sets the stop condition for analog input AD conversion sampling.*<sup>1</sup></p> <ul style="list-style-type: none"> <li>• To set the conversion stop condition to the end of conversion a set number of times, use the "SetAiStopTimes" command to set the number of sampling times.</li> <li>• To set the conversion stop condition to level comparison, use the "SetAiStopLevel" or "SetAiStopLevelEx" command to set the level comparison stop.</li> <li>• If the conversion stop condition is set to in-range comparison, use the "SetAiStopInRange" or "SetAiStopInRangeEx" command to set the in-range comparison stop.</li> <li>• When setting the conversion stop condition to outrange comparison, use the "SetAiStopOutOfRange" or "SetAiStopOutOfRangeEx" command to set the outrange comparison stop.</li> <li>• When using the event controller output as a clock, connect the event controller with the "SetEcuSignal" command.</li> </ul> <p>[For users of the multifunction device AIO-121601M-PCI]*<sup>1</sup></p> <ul style="list-style-type: none"> <li>• When setting the conversion stop condition to compare count match <ul style="list-style-type: none"> <li>Use the "CntmNotifyCountUp" command to set the condition for the occurrence of the comparison count match.</li> <li>• To set the conversion stop condition to count clear <ul style="list-style-type: none"> <li>Use the "CntmZeroClearCount" command to clear the count.</li> </ul> </li> </ul> </li> <li>• When setting the conversion stop condition to carryborough <ul style="list-style-type: none"> <li>Use the "CntmNotifyCarryBorrow" command to set the conditions under which carryborrow occurs.</li> </ul> </li> <li>• To set the conversion stop condition to a timer <ul style="list-style-type: none"> <li>Use the "CntmNotifyTimer" command to set the timer event occurrence conditions.</li> </ul> </li> </ul>

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## SetAiStopTimes

---

<b>Syntax</b>	<i>object.</i> SetAiStopTimes ( <Times> )
<b>Arguments</b>	<Times> = VT_I4: Conversion stop sampling frequency * <sup>1</sup>
<b>Return value</b>	None
<b>Description</b>	Sets the number of times the analog input AD conversion sampling stops.* <sup>1</sup> This setting is required when the conversion stop condition is set to the end of the set number of conversions by the "SetAiStopTrigger" command. * If the conversion stop condition is other than the end of the set number of conversions, you do not need to execute this command.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## SetAiStopLevel

---

<b>Syntax</b>	<i>object.</i> SetAiStopLevel ( <Channel>, <Level>, <Direction> )
<b>Arguments</b>	<Channel> = VT_I2: AI channel number for level comparison* <sup>1</sup>  <Level> = VT_I4: Data for level comparison* <sup>1</sup> This parameter is set as a binary value from the following range. The value that can be set differs depending on the device. Devices with 12-bit resolution: 0 to 4095 Devices with 16-bit resolution: 0 to 65535  <Direction> = VT_I2: Direction of level comparison* <sup>1</sup> Set the value from the following range. 0: Both 1: Rising edge 2: Falling edge
<b>Return value</b>	None
<b>Description</b>	Sets the level at which level comparison stops when sampling analog input AD conversion by a binary value.* <sup>1</sup> This setting is required when the conversion stop condition is set to level comparison by the "SetAiStopTrigger" command. * If the conversion stop condition is other than level comparison, it is not necessary to execute this command.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## SetAiStopLevelEx

---

<b>Syntax</b>	<i>object.</i> SetAiStopLevelEx ( <Channel>, <Level>, <Direction> )
<b>Arguments</b>	<Channel> = VT_I2: AI channel number for level comparison * <sup>1</sup>  <Level> = VT_R4: Data for level comparison* <sup>1</sup> Set by voltage or current.  <Direction> = VT_I2: Direction of level comparison * <sup>1</sup>

	Set the value from the following range. 0: Both 1: Rising edge 2: Falling edge
<b>Return value</b>	None
<b>Description</b>	Sets the level at which level comparison stops when sampling analog input AD conversion by voltage or current.* <sup>1</sup> This setting is required when the conversion stop condition is set to level comparison by the "SetAiStopTrigger" command. * If the conversion stop condition is other than level comparison, it is not necessary to execute this command.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## SetAiStopInRange

<b>Syntax</b>	<i>object.</i> SetAiStopInRange( <Channel>, <Level1>, <Level2>, <StateTimes> )
<b>Arguments</b>	<Channel> = VT_I2: AI channel number for in-range comparison* <sup>1</sup>  <Level1>, <Level2> = VT_I4: Range for in-range comparison* <sup>1</sup> Set the range as a binary value from the following range. The value that can be set depends on the device. Devices with 12-bit resolution: 0 to 4095 Devices with 16-bit resolution: 0 to 65535  <StateTimes> = VT_I4: State retention times* <sup>1</sup> Set the number of sampling times to hold the state after an in-range comparison is established. The actual conversion stops after the sampling time of StateTimes has elapsed.
<b>Return value</b>	None
<b>Description</b>	Sets the level at which the in-range comparison stops when sampling analog input AD conversion by a binary value.* <sup>1</sup> This setting is required when the conversion stop condition is set to in-range comparison by the "SetAiStopTrigger" command. * If the conversion stop condition is other than in-range comparison, it is not necessary to execute.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## SetAiStopInRangeEx

<b>Syntax</b>	<i>object.</i> SetAiStopInRangeEx( <Channel>, <Level1>, <Level2>, <StateTimes> )
<b>Arguments</b>	<Channel> = VT_I2: AI channel number for in-range comparison* <sup>1</sup>  <Level1>, <Level2> = VT_R4: Range for in-range comparison* <sup>1</sup> Set by voltage or current.  <StateTimes> = VT_I4: State retention count* <sup>1</sup> After the in-range comparison is established, the period of time to hold the state is set by the number of sampling times. The actual conversion stops after the sampling time of StateTimes has elapsed.

<b>Return value</b>	None
<b>Description</b>	<p>Sets the level at which in-range comparison stops when sampling analog input AD conversion by voltage or current.*<sup>1</sup></p> <p>This setting is required when the conversion stop condition is set to in-range comparison by the "SetAiStopTrigger" command.</p> <p>* If the conversion stop condition is other than in-range comparison, it is not necessary to execute.</p>

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## SetAiStopOutOfRange

<b>Syntax</b>	<i>object.</i> SetAiStopOutOfRange( <Channel>, <Level1>, <Level2>, <StateTimes> )
<b>Arguments</b>	<p>&lt;Channel&gt; = VT_I2: AI channel number for outrange comparison *<sup>1</sup></p> <p>&lt;Level1&gt;, &lt;Level2&gt; = VT_I4: Range for outrange comparison *<sup>1</sup></p> <p>Set the value as a binary value from the following range. The value that can be set depends on the device.</p> <p>Devices with 12-bit resolution: 0 to 4095</p> <p>Devices with 16-bit resolution: 0 to 65535</p> <p>&lt;StateTimes&gt; = VT_I4: State retention count*<sup>1</sup></p> <p>The period of time to hold the state after the outrange comparison is established is set by the number of sampling times.</p> <p>The actual conversion stops after the sampling time of StateTimes has elapsed.</p>
<b>Return value</b>	None
<b>Description</b>	<p>Sets the level at the time of stopping out-range comparison during analog input AD conversion sampling as a binary value.*<sup>1</sup></p> <p>This setting is required when the conversion stop condition is set to outrange comparison by the "SetAiStopTrigger" command.</p> <p>* If the conversion stop condition is other than outrange comparison, it is not necessary to execute.</p>

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## SetAiStopOutOfRangeEx

<b>Syntax</b>	<i>object.</i> SetAiStopOutOfRangeEx( <Channel>, <Level1>, <Level2>, <StateTimes> )
<b>Arguments</b>	<p>&lt;Channel&gt; = VT_I2: AI channel number for outrange comparison*<sup>1</sup></p> <p>&lt;Level1&gt;, &lt;Level2&gt; = VT_R4: Range for outrange comparison*<sup>1</sup></p> <p>Set by voltage or current.</p> <p>&lt;StateTimes&gt; = VT_I4: State retention count*<sup>1</sup></p> <p>The period of time to hold the state after the outrange comparison is established is set by the number of sampling times.</p> <p>The actual conversion stops after the sampling time of StateTimes has elapsed.</p>
<b>Return value</b>	None
<b>Description</b>	<p>Sets the level at which out-of-range comparison stops when sampling analog input AD conversion by voltage or current.*<sup>1</sup></p>

This setting is required when the conversion stop condition is set to outrange comparison by the "SetAiStopTrigger" command.

\* If the conversion stop condition is other than outrange comparison, it is not necessary to execute.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## SetAiStopDelayTimes

---

<b>Syntax</b>	<i>object.</i> SetAiStopDelayTimes ( <Times> )
<b>Arguments</b>	<Times> = VT_I4: Conversion stop delay count *1 Set the number of conversion stop delays.
<b>Return value</b>	None
<b>Description</b>	Sets the number of stop delays during analog input AD conversion sampling.*1

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## SetAiRepeatTimes

---

<b>Syntax</b>	<i>object.</i> SetAiRepeatTimes ( <Times> )
<b>Arguments</b>	<Times> = VT_I4: Number of repeats *1 0: The repeat operation is repeated infinitely. 1~: Repeats the operation for the specified number of times.
<b>Return value</b>	None
<b>Description</b>	Sets the number of repeats for analog input AD conversion sampling.*1 Repeating operation is the number of times a series of operations from sampling start to stop is repeated.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## SetAiEventConditions

---

<b>Syntax</b>	<i>object.</i> SetAiEventConditions ( <Event>, <DataType> )
<b>Arguments</b>	<Event> = VT_I4: Event factors *1 Each bit has the following meanings, which can be combined to set multiple event generation factors. 00000002H: AD conversion start condition event 00000010H: Repetition end event 00000020H: Device operation end event 00000080H: Specified sampling frequency storage event 00010000H: Overflow event 00020000H: Sampling clock error event 00040000H: AD conversion error event  <DataType> = VT_I4: Conversion data type Specify the conversion data type of data to be retrieved when CaoController::OnMessage event occurs. 0: Voltage or current value

	1: binary value 2: No acquired data (event notification only)
<b>Return value</b>	None
<b>Description</b>	Sets the event factors of <code>CaoController::OnMessage</code> related to analog input. To use the event factors that store the specified number of sampling times, set the number of sampling times to generate events with the "SetAiEventSamplingTimes" command.  Note: This provider supports only the "device buffer mode," which is the initial value of the device startup, for the conversion data transfer method (the "user buffer mode" is not supported).

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## SetAiEventSamplingTimes

<b>Syntax</b>	<code>object. SetAiEventSamplingTimes ( &lt;SamplingTimes&gt; )</code>
<b>Arguments</b>	<SamplingTimes> = VT_I4: Number of samples to generate an event.*1
<b>Return value</b>	None
<b>Description</b>	Set the sampling frequency when using the analog input specified sampling frequency storage event.*1 The "SetAiEventConditions" command is used to set the number of samples when using the specified sampling frequency storage event factors.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## StartAiSamplingAsync

<b>Syntax</b>	<code>object. StartAiSamplingAsync ()</code>
<b>Arguments</b>	None
<b>Return value</b>	None
<b>Description</b>	Starts analog input AD conversion sampling and returns processing immediately. (Asynchronous operation)*1 Starts AD conversion based on the set conditions. Sampling clock errors and AD conversion errors are automatically reset. The device memory is not reset. If the past conversion data exists in the memory, the new conversion data is stored following the past data.

Note: In this provider, if the event factor has been set in advance by the "SetAiEventCondition" command (Event <> 0), the event factor setting is executed again before starting AD conversion.

This is to give priority to the event factor <Event> and conversion data type <DataType> set by the process that starts device operation with this command, even if event factors have been set for the same device by other processes in a multi-process operating environment.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..



---

## StartAiSamplingSync

---

<b>Syntax</b>	<b>object. StartAiSamplingSync ( &lt;Timeout&gt; )</b>
<b>Arguments</b>	<Timeout> = VT_I4: timeout period *1 Specifies the timeout period in milliseconds before the function returns. If 0 is specified, the function will continue to wait until it stops.
<b>Return value</b>	None
<b>Description</b>	This function starts analog input AD conversion sampling and returns the process after the timeout period has elapsed. (synchronous operation) *1 AD conversion is started based on the set conditions. Sampling clock errors and AD conversion errors are automatically reset. The device memory is not reset. If previous conversion data exists in the memory, the new conversion data is stored following the previous data.

Note: In this provider, if the event factor has been set in advance by the "SetAiEventCondition" command (Event <> 0), the event factor setting is executed again before starting AD conversion.

This is to give priority to the event factor <Event> and conversion data type <DataType> set by the process that starts device operation with this command, even if event factors have been set for the same device by other processes in a multi-process operating environment.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## StopAiSampling

---

<b>Syntax</b>	<b>object. StopAiSampling ()</b>
<b>Arguments</b>	None
<b>Return value</b>	None
<b>Description</b>	Stops analog input AD conversion sampling. *1 If multiple channels are being converted, sampling will stop after all channels have been converted at the time the command is executed.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## GetAiSamplingCount

---

<b>Syntax</b>	<b>object. GetAiSamplingCount ()</b>
<b>Arguments</b>	None
<b>Return value</b>	<Data> =VT_I4: サンプル回数
<b>Description</b>	Analog Input Obtains the number of samples (positions) in the device memory (driver memory). *1

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

---

## GetAiSopTriggerCount

---

<b>Syntax</b>	<i>object</i> . GetAiStopTriggerCount ()
<b>Arguments</b>	None
<b>Return value</b>	<Data> = VT_I4: Sampling frequency
<b>Description</b>	Analog Input Get the sampling frequency when the stop trigger is input.* <sup>1</sup>

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## GetAiSamplingData

---

<b>Syntax</b>	<i>object</i> . GetAiSamplingData ( <SamplingTimes> )
<b>Arguments</b>	<SamplingTimes> = VT_I4: Specified sampling frequency * <sup>1</sup>
<b>Return value</b>	<Data> = VT_ARRAY   VT_VARIANT Array[0]: VT_I4: Actual number of samples taken. Array[1]: VT_EMPTY: When there is no sampling data (Array[0]=0) VT_ARRAY   VT_I4: Array of converted data with (Array[0] * number of AI channels used) elements The conversion data are binary values in the following ranges • Devices with 12-bit resolution: 0 to 4095 • Devices with 16-bit resolution: 0 to 65535
<b>Description</b>	Obtains the conversion data for the specified sampling from the device memory (driver memory) as a binary value.* <sup>1</sup> Stores the data for the number of AI channels used. • For FIFO memory Always reads from the oldest data. Once the data has been loaded, it cannot be loaded again. • For RING memory Retrieves the specified number of samples from the most recent data. The same data can be retrieved repeatedly.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## GetAiSamplingDataEx

---

<b>Syntax</b>	<i>object</i> . GetAiSamplingDataEx ( <SamplingTimes> )
<b>Arguments</b>	<SamplingTimes> = VT_I4: Specified sampling frequency* <sup>1</sup>
<b>Return value</b>	<Data> = VT_ARRAY   VT_VARIANT Array[0]: VT_I4: Actual number of samples taken. Array[1]: VT_EMPTY: When there is no sampling data (Array[0]=0) VT_ARRAY   VT_R4: Array of converted data with (Array[0] * number of AI channels used) elements The conversion data is stored as voltage or current.
<b>Description</b>	Obtains the conversion data for the specified sampling from the device memory (driver memory) in the form of voltage or current values.* <sup>1</sup> Stores the data for the number of AI channels used.

---

- For FIFO memory
  - Always reads from the oldest data.
  - Once the data has been loaded, it cannot be loaded again.
- For RING memory
  - Retrieves the specified number of samples from the most recent data.
  - The same data can be retrieved repeatedly.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## GetAiStatus

<b>Syntax</b>	<code>object.GetAiStatus ()</code>
<b>Arguments</b>	None
<b>Return value</b>	<p>&lt;Status&gt; = VT_I4: Status*<sup>1</sup></p> <p>00000001H: Device in operation</p> <p>00000002H: Waiting for start trigger</p> <p>00000010H: Store more than the specified number of data</p> <p>00010000H: Overflow</p> <p>00020000H: Sampling clock error</p> <p>00040000H: AD conversion error</p> <p>00080000H: driver spec error</p>
<b>Description</b>	Get the analog input status.* <sup>1</sup>

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## GetAiResolution

<b>Syntax</b>	<code>object.GetAiResolution ()</code>
<b>Arguments</b>	None
<b>Return value</b>	<p>&lt;Resolution&gt; = VT_I2: resolution*<sup>1</sup></p> <p>12: 12-bit resolution</p> <p>16: 16-bit resolution</p> <p>0: No analog input function</p>
<b>Description</b>	Get the resolution of the analog input.* <sup>1</sup>

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## ResetAiMemory

<b>Syntax</b>	<code>object.ResetAiMemory ()</code>
<b>Arguments</b>	None
<b>Return value</b>	None
<b>Description</b>	<p>Resets the analog input device memory.*<sup>1</sup></p> <p>This command will reset the following states.</p> <ul style="list-style-type: none"> <li>• Resets the value of the memory management pointers (read and write pointers) to 0.</li> <li>• The number of repeats will be reset to zero.</li> <li>• The sampling count at the time of stop trigger input is reset to 0.</li> </ul>

- The buffer overflow status is reset.
- The status of the specified number of data is reset.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## ResetAiStatus

<b>Syntax</b>	<i>object. ResetAiStatus ()</i>
<b>Arguments</b>	None
<b>Return value</b>	None
<b>Description</b>	Resets the analog input status.* <sup>1</sup> This command will reset the following states. <ul style="list-style-type: none"> <li>•The sampling clock error is reset.</li> <li>•The AD conversion error status is reset.</li> </ul>

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## SetEcuSignal

<b>Syntax</b>	<i>object. SetEcuSignal ( &lt;Destination&gt;, &lt;Source&gt; )</i>
<b>Arguments</b>	<Destination> = VT_I2: Destination signal <Source> = VT_I2: Source signal *For details, please refer to the CONTEC Product Manual and API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB).
<b>Return value</b>	None
<b>Description</b>	Sets the signal settings for the event controller.* <sup>1</sup>

By switching the control signals for each function (Ai, Ao, Cnt), multi-functional operations can be performed. The control of the synchronous bus is also set by this function.

Some combinations are available and some are not.\*<sup>1</sup>

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## SetCntmNotifyCountUp

Only when using AIO-121601M-PCI\*1

<b>Syntax</b>	<i>object. SetCntmNotifyCountUp ( &lt;ChNo&gt;, &lt;RegNo&gt;, &lt;Count&gt; )</i>
<b>Arguments</b>	<ChNo> = VT_I2: Counter CH number <RegNo> = VT_I2: Comparison register number <Count> = VT_UI4: Comparison value to be set in the comparison register Specifiable range: For a 24-bit counter device : 0h <= Count <= FFFFFFFh For 32-bit counter devices : 0h <= Count <= FFFFFFFFh
<b>Return value</b>	None
<b>Description</b>	Specifies the notification by count match and sets the comparison register.* <sup>1</sup> Note: This provider does not use the message notification function by count match.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## SetCntmZeroClearCount

Only when using AIO-121601M-PCI\*1

<b>Syntax</b>	<b><i>object.</i> SetCntmZeroClearCount ( &lt;Channels&gt; )</b>
<b>Arguments</b>	<Channels> = VT_ARRAY   VT_I2: Counter CH number an array containing
<b>Return value</b>	None
<b>Description</b>	This function clears the count value of the counter to zero.* <sup>1</sup> Clears the count value of the specified counter channel to zero. The channel number is passed as an array. This function is valid both before and after the counter operation starts.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## SetCntmNotifyCarryBorrow

Only when using AIO-121601M-PCI\*1

<b>Syntax</b>	<b><i>object.</i> SetCntmNotifyCarryBorrow ()</b>
<b>Arguments</b>	None
<b>Return value</b>	None
<b>Description</b>	カウンタのカウンタのキャリー／ボロー通知の設定を行います。* <sup>1</sup> この関数は、カウンタスタート前に実行してください。 Note: This provider does not use the carry/borrow message notification function.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## SetCntmNotifyTimer

Only when using AIO-121601M-PCI\*1

<b>Syntax</b>	<b><i>object.</i> SetCntmNotifyTimer ( &lt;TimeValue&gt; )</b>
<b>Arguments</b>	<TimeValue> =VT_UI4: Timer value[msec] Specifiable range: 0<=TimeValue<=6553
<b>Return value</b>	None
<b>Description</b>	Sets the notification for the timer.* <sup>1</sup> Note: This provider does not use the message notification function by carry/borrow.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## StartCntmCount

Only when using AIO-121601M-PCI\*1

<b>Syntax</b>	<b><i>object.</i> StartCntmCount ( &lt;Channels&gt; )</b>
<b>Arguments</b>	<Channels> = VT_ARRAY   VT_I2: Counter CH number an array containing
<b>Return value</b>	None
<b>Description</b>	Starts the counting operation for the specified channel.* <sup>1</sup> The channel number is passed as an array. The mode set before the counter start will be set when this function is called. The preset values set before the counter start will be set when this function is called.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## StopCntmCount

Only when using AIO-121601M-PCI\*1

<b>Syntax</b>	<i>object.</i> StopCntmCount ( <Channels> )
<b>Arguments</b>	<Channels> = VT_ARRAY   VT_I2: Counter CH number an array containing
<b>Return value</b>	None
<b>Description</b>	Stops the counting operation for the specified channel. *1 The channel number is passed as an array.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## PresetCntm

Only when using AIO-121601M-PCI\*1

<b>Syntax</b>	<i>object.</i> PresetCntm ( <ChNo>, <PresetData> )
<b>Arguments</b>	<ChNo> = VT_I2: Counter CH number <PresetData> = VT_UI4: Preset value
<b>Return value</b>	None
<b>Description</b>	Presets the count value of the specified channel.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## ReadCntmCount

Only when using AIO-121601M-PCI\*1

<b>Syntax</b>	<i>object.</i> ReadCntmCount ( <Channels> )
<b>Arguments</b>	<Channels> = VT_ARRAY   VT_I2: Counter CH number an array containing
<b>Return value</b>	<CntData> = VT_ARRAY VT_UI4 : Array to store the acquisition count value
<b>Description</b>	Reads the count value of the specified channel.*1 The channel number is passed as an array. In the order in which the channel numbers are stored in the array, the corresponding count values are stored in the array. Since the count value of the specified channel is latched and then read, the count value at the same timing can be obtained even when multiple channels are specified.

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## ReadCntmStatusEx

Only when using AIO-121601M-PCI\*1

<b>Syntax</b>	<i>object.</i> ReadCntmStatusEx ( <ChNo> )
<b>Arguments</b>	<ChNo> = VT_I2: Counter CH number
<b>Return value</b>	<Status> = VT_UI4: Acquisition status value*1 Each bit is defined as follows. bit0: General-purpose input state (1:LOW, 0:HIGH) bit1: Count match register 0 (1: mismatch, 0: match) bit2: Count match register 1 (1: mismatch, 0: match) bit3: UP/DOWN(1: Counting down, 0: Counting up)

bit4: phase B (1:HIGH, 0:LOW)  
 bit5: phase A(1:HIGH, 0:LOW)  
 bit6: phase Z(1:positive logic=HIGH/ negative logic =LOW, 0:positive logic=LOW /  
 negative logic =HIGH)  
 bit7: Borrow(1:Borrow Detection, 0:Borrow Undetected)  
 bit8: Carry(1: Carry Detection, 0: Carry Undetected)  
 bit9: filter-error (1: filter-error detection, 0: filter-error undetected)  
 bit10: Abnormal input error (1: Abnormal input error detection, 0: Abnormal input error  
 undetected)  
 bit11: Disconnection alarm error (1: Disconnection alarm error detection, 0: Disconnection  
 alarm error undetected)

**Description**

Get the status of the specified counter channel.\*1

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## SetCntmZMode

Only when using AIO-121601M-PCI\*1

**Syntax**

*object.* SetCntmZMode ( <ChNo>, <Mode> )

**Arguments**

<ChNo> = VT\_I2: Counter CH number

<Mode> = VT\_I2: How to use\*1

Specifiable range:

1: Not used

2: Next one

3: Every time

**Return value**

None

**Description**

Sets the Z-phase usage method for the specified counter channel (Not used, Next one, or Every time).\*1

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## SetCntmZLogic

Only when using AIO-121601M-PCI\*1

**Syntax**

*object.* SetCntmZLogic ( <ChNo>, <Logic> )

**Arguments**

<ChNo> = VT\_I2: Counter CH number

<Logic> = VT\_I2: logic\*1

Specifiable range:

0: positive logic

1: negative logic

**Return value**

None

**Description**

Sets the Z-phase logic of the specified counter channel (positive logic/negative logic).\*1

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

---

## SetCntmCountDirection

Only when using AIO-121601M-PCI\*1

**Syntax**

*object.* SetCntmCountDirection ( <ChNo>, <Dir> )

**Arguments**

<ChNo> = VT\_I2: Counter CH number

	<Dir> = VT_I2: Counting direction* <sup>1</sup> Specifiable range: 0: down-count 1: upcount
<b>Return value</b>	None
<b>Description</b>	Sets the count direction of the specified counter channel (up count or down count). <sup>*1</sup>

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## SetCntmOperationMode

Only when using AIO-121601M-PCI\*<sup>1</sup>

<b>Syntax</b>	<i>object.</i> SetCntmOperationMode ( <ChNo>, <Phase>, <Mul>, <SyncClr> )
<b>Arguments</b>	<ChNo> = VT_I2: Counter CH number <Phase > = VT_I2: phase number* <sup>1</sup> Specifiable range: 0: single phase 1: two phases 2: Gate Controls <Mul > = VT_I2: multiplication* <sup>1</sup> Specifiable range: 0: multiplication by one 1: multiplication by two 2: Quadruple <SyncClr > = VT_I2: Synchronous clear/asynchronous clear* <sup>1</sup> Specifiable range: 0: Asynchronous Clearing 1: Synchronous Clear
<b>Return value</b>	None
<b>Description</b>	Sets the operation mode of the specified counter channel (phase number, clear, multiplication). <sup>*1</sup>

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..

## SetCntmDigitalFilter

Only when using AIO-121601M-PCI\*<sup>1</sup>

<b>Syntax</b>	<i>object.</i> SetCntmDigitalFilter ( <ChNo>, <FilterValue> )
<b>Arguments</b>	<ChNo> = VT_I2: Counter CH number <FilterValue> = VT_I2: Digital filter coefficient value* <sup>1</sup>
<b>Return value</b>	None
<b>Description</b>	Sets the digital filter value for the specified counter channel.* <sup>1</sup>

\*1: For details, refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB)..



## 2.4. Variable list

### 2.4.1. Controller class

**Table 2-4 Controller class user variable list**

Variable name	Data type	Explanation	Attribute	
			get	put
AI? <sup>*2</sup>	VT_R4	Acquire a voltage value <sup>*1</sup> of analog input CH?. Specify the logical number <sup>*6</sup> after the variable name. Example: "AI1"	√	—
AO? <sup>*3</sup>	VT_R4	Output a specified voltage value <sup>*1</sup> to analog output CH?. Specify the logical number <sup>*6</sup> after the variable name. Example: "AO1"	—	√
DI? <sup>*4</sup>	VT_I2	Acquire a bit value (0 or 1) <sup>*1</sup> of digital input bit. Specify the logical number <sup>*6</sup> after the variable name. Example: "DI1"	√	—
DO? <sup>*5</sup>	VT_I2	Output a bit value (0 or 1) <sup>*1</sup> to digital output bit. Specify the logical number <sup>*6</sup> after the variable name. Example: "DO1"	—	√
DIB? <sup>*4</sup>	VT_I2	Acquire a byte value (0 to 255) <sup>*1</sup> of digital input byte. Specify the logical number <sup>*6</sup> after the variable name. Example: "DIB1"	√	—
DOB? <sup>*5</sup>	VT_I2	Output a byte value (0 to 255) <sup>*1</sup> to digital output byte. Specify the logical number <sup>*6</sup> after the variable name. Example: "DOB1"	—	√

\*1: Refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB) for details.

\*2: This is available only on the models equipped with analog input. Refer to CONTEC product manual for details.

\*3: This is available only on the models equipped with analog output. Refer to CONTEC product manual for details.

\*4: This is available only on the models equipped with digital input. Refer to CONTEC product manual for details.

\*5: This is available only on the models equipped with digital output. Refer to CONTEC product manual for details.

\*6: A variable object can be created with a logical number within the range of 0 to 99; however, the actual range for data acquisition/setting depends on the number of CHs implemented, etc. of the mounted model. Refer to CONTEC product manual for details.

**Table 2-5 Controller class system variable list**

Variable name	Data type	Explanation	Attribute	
			get	put
@MAX_AI <sup>*2</sup>	VT_I2	Acquire the maximum number of analog input channels <sup>*1</sup> .	√	—
@MAX_AO <sup>*3</sup>	VT_I2	Acquire the maximum number of analog output channels <sup>*1</sup> .	√	—
@ProcessId	VT_I4	Acquire process ID.	√	—
@DeviceName	VT_BSTR	Acquire device name <sup>*1</sup> of connected board.	√	—
@RANGE_AO <sup>*4</sup>	VT_I2	Set/Obtains all of analog output range (all channels) <sup>*1</sup> .	√	√
@RANGE_AI <sup>*5</sup>	VT_I2	Set/Obtains all of analog input range (all channels) <sup>*1</sup> .	√	√
@STS_AI <sup>*2</sup>	VT_I4	Obtains the analog input status <sup>*1</sup> .  Acquisition value: 00000001H: Device is operating 00000002H: Waiting for start trigger 00000010H: Storing more than the specified number of data 00010000H: Overflow 00020000H: Sampling clock error 00040000H: AD conversion error 00080000H: Driver specification error  Note: If multiple statuses are occurring at the same time, the sum value will be obtained <sup>*1</sup> .	√	—

\*1: Refer to API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB) for details.

\*2: This is available only on the models equipped with analog input. Refer to CONTEC product manual for details.

\*3: This is available only on the models equipped with analog output. Refer to CONTEC product manual for details.

\*4: This is available only on the models equipped with analog output and the models which can execute range setting by means of function execution. Refer to CONTEC product manual for details.

\*5: This function is available only for models equipped with analog input and capable of range setting by function execution. For details, please refer to the CONTEC product manual.

## 2.5. Error code

The AIO provider has following two types of specific error codes.

### 1) Errors returned by AIO API

This returns an AIO API-issued error which is masked by "0x8010\*\*\*\*".

Ex: AIO API error: 0xFFFF >> AIO API error:0x8010FFFF

For detailed information about AIO API, refer to API-AIO(WDM) Help (PCI) or API-USBP (W32) Help(USB) of CONTEC.co.ltd.

### 2) ORiN2 Common errors

For common errors of ORiN2, refer to the error code section in "[ORiN2 Programming Guide](#)".

## 2.6. CAO-AIO API reference table

The AIO provider executes API function that sets/acquires a value via CaoVariable.

**Table 2-6 Controller class, variable class and corresponding AIO API**

CAO API		AIO API*
Class::method name	Parameter name/Command name/ Variable name	
CaoWorkspace::AddController()	DeviceName	AioQueryDeviceName() AioInit()
	Coexistence	AioGetAiStatus() AioGetAoStatus() AioResetProcess() ... Only when device is stopped*
	ResetDevice	AioResetDevice() < USB devices only (measures to prevent malfunction during cable insertion/removal) > AioGetDeviceType() AioExit() AioInit()
	AiInputMethod	AioSetAiInputMethod()
	AiChannels	AioSetAiChannels()
	AiRangeAll	AioSetAiRangeAll()
	AiMemoryType	AioSetAiMemoryType()
	AiClockType	AioSetAiClockType()
	AiSamplingClock	AioSetAiSamplingClock()
	AiClockEdge	AioSetAiClockEdge
CaoWorkspaces::Remove()	—	AioExit()
CaoController::Execute()	SetAiStartTrigger	AioSetAiStartTrigger()
	SetAiStartLevel	AioSetAiStartLevel()
	SetAiStartLevelEx	AioSetAiStartLevelEx()
	SetAiStartInRange	AioSetAiStartInRange()
	SetAiStartInRangeEx	AioSetAiStartInRangeEx()
	SetAiStartOutOfRange	AioSetAiStartOutOfRange()
	SetAiStartOutOfRangeEx	AioSetAiStartOutOfRangeEx()
	SetAiStopTrigger	AioSetAiStopTrigger()
	SetAiStopTimes	AioSetAiStopTimes()
	SetAiStopLevel	AioSetAiStopLevel()
	SetAiStopLevelEx	AioSetAiStopLevelEx()
	SetAiStopInRange	AioSetAiStopInRange()
	SetAiStopInRangeEx	AioSetAiStopInRangeEx()
	SetAiStopOutOfRange	AioSetAiStopOutOfRange()
	SetAiStopOutOfRangeEx	AioSetAiStopOutOfRangeEx()
SetAiStopDelayTimes	AioSetAiStopDelayTimes()	

	SetAiRepeatTimes	AioSetAiRepeatTimes()
	SetAiEventConditions	AioSetAiCallBackProc()
	SetAiEventSamplingTimes	AioSetAiEventSamplingTimes()
	StartAiSamplingAsync	AioSetAiCallBackProc() ... Event factor: When not '0'. AioGetAiStatus() AioStartAi()
	StartAiSamplingSync	AioSetAiCallBackProc() ... Event factor: When not '0'. AioStartAiSync()
	StopAiSampling	AioStopAi()
	GetAiSamplingCount	AioGetAiSamplingCount()
	GetAiStopTriggerCount	AioGetAiStopTriggerCount()
	GetAiSamplingData	AioGetAiSamplingData()
	GetAiSamplingDataEx	AioGetAiSamplingDataEx()
	GetAiResolution	AioGetAiResolution()
	ResetAiMemory	AioResetAiMemory()
	ResetAiStatus	AioResetAiStatus()
	SetEcuSignal	AioSetEcuSignal()
	SetCntmNotifyCountUp	AioCntmNotifyCountUp()
	SetCntmZeroClearCount	AioCntmZeroClearCount()
	SetCntmNotifyCarryBorrow	AioCntmNotifyCarryBorrow()
	SetCntmNotifyTimer	AioCntmNotifyTimer()
	StartCntmCount	AioCntmStartCount()
	StopCntmCount	AioCntmStopCount()
	PresetCntm	AioCntmPreset()
	ReadCntmCount	AioCntmReadCount()
	ReadCntmStatusEx	AioCntmReadStatusEx()
	SetCntmZMode	AioSetCntmZMode()
	SetCntmZLogic	AioSetCntmZLogic()
	SetCntmCountDirection	AioSetCntmCountDirection()
	SetCntmOperationMode	AioSetCntmOperationMode()
	SetCntmDigitalFilter	AioSetCntmDigitalFilter()
CaoController::OnMessage()	DI relation	AioInputDiByte()
	AI relation	AioGetAiChannels() AioGetAiSamplingData() AioGetAiSamplingDataEx()
CaoVariable::get_Value()	AI?	AioSingleAiEx()
	AIS	AioGetAiSamplingCount() AioGetAiChannels() AioGetAiSamplingDataEx()
	DI?	AioInputDiBit()
	DIB?	AioInputDiByte()
	@MAX_AI	AioGetAiMaxChannels()
	@MAX_AO	AioGetAoMaxChannels()
	@ProcessId	—
	@DeviceName	—
	@RANGE_AO	AioGetAoRange()
	@RANGE_AI	AioSetAiRangeAll()
	@STS_AI	AioGetAiStatus()
CaoVariable::put_Value()	AO?	AioSingleAoEx()

---

	DO?	AioOutputDoBit()
	DOB?	AioOutputDoByte()
	@RANGE_AO	AioSetAoRangeAll()

\* Refer to CONTEC API-AIO (WDM) Help (for PCI) or API-USBP (W32) Help (for USB) for details of AIO API.

## 3. Sample Program

### 3.1. Specified analog input channel AD conversion data acquisition sample (simple AI input function)

The following sample program shows the code to acquire AI CH1 voltage value with the variable "AI1".

#### List 3-1 SampleAi.frm

```
Private caoEng As CaoEngine
Private caoAIOI As CaoController
Private caoVar As CaoVariable

Private Sub Form_Load()

    Set caoEng = New CaoEngine
    Set caoAIOI = caoEng.Workspaces(0).AddController("SampleAi", "CaoProv.CONTEC.AIO", "", _
        "DeviceName=AIO001")

    Set caoVar = caoAIOI.AddVariable("AI1", "")

End Sub

Private Sub cmdGet_Click()

    Dim sngRet As Single

    sngRet = caoVar.Value

    Text1.Text = CStr(sngRet)

End Sub
```

### 3.2. Digital Input State Change Event Reception Sample

The following sample program shows the code to receive event at the time the digital input byte changes while the sampling interval is set to one second.

#### List 3-2 SampleEvent.frm

```
Private caoEng As CaoEngine
Private WithEvents caoAIOI As CaoController

Private Sub Form_Load()

    Set caoEng = New CaoEngine
    Set caoAIOI = caoEng.Workspaces(0).AddController("SampleEvent","CaoProv.CONTEC.AIO", _
        "", "DeviceName=AIO001,Interval=1000")

End Sub

' Received event
Private Sub ctrl_OnMessage(ppCaoMess As CAOLib.ICaoMessage)

    ' Received digital input byte
    text2.Text = ppCaoMess.Value

End Sub
```

End Sub60

### 3.3. Analog input sampling frequency storage event reception sample (high-function AI input function)

The following sample shows how to acquire AI sampling data for two channels in voltage or current values when receiving an AD specified sampling frequency storage event.

#### List 3-3

#### SampleEventAi.frm

```
Private m_caoEng As CaoEngine
Private WithEvents m_caoCtrl As CaoController

Private Sub Form_Load()

    Dim vntArg As Variant

    Set m_caoEng = New CaoEngine
    Set m_caoCtrl = m_caoEng.Workspaces(0).AddController("SampleEventAi",
"CaoProv.CONTEC.AIO", "", "DeviceName=AIO000, AiChannels=2, AiSamplingClock=10000 ")

    ' Sampling Start condition: Software (0)
    vntArg = Array(CLng(0))
    m_caoCtrl.Execute "SetAiStartTrigger", vntArg

    ' Sampling Stop condition: Command (4)
    vntArg = Array(CLng(4))
    m_caoCtrl.Execute "SetAiStopTrigger", vntArg

    ' Sampling frequency of event occurrence : 100
    vntArg = Array(CLng(100))
    m_caoCtrl.Execute "SetAiEventSamplingTimes", vntArg

End Sub

' Sampling start button
Private Sub cmdStartSampling_Click()

    txtText1.Text = ""
    DoEvents

    ' Event factors : <Event    > Specified sampling frequency storage event (00000080H)
    '                               Overflow events (00010000H)
    '                               Sampling clock error event (00020000H)
    '                               AD conversion error event (00040000H)
    '                               <DataType> Voltage or current value ( 0 )
    Dim vntArg As Variant
    vntArg = Array(CLng(&H70080), CLng(0))
    m_caoCtrl.Execute "SetAiEventConditions", vntArg

    ' Device memory reset
    vntArg = Empty
    m_caoCtrl.Execute "ResetAiMemory", vntArg

    ' Status reset
    m_caoCtrl.Execute "ResetAiStatus", vntArg

    ' Sampling start
```



```

        m_caoCtrl.Execute "StartAiSamplingAsync", vntArg
    End Sub

' Sampling stop button
Private Sub cmdStopSampling_Click()

    Dim vntArg As Variant

    ' Sampling stopped
    vntArg = Empty
    m_caoCtrl.Execute "StopAiSampling", vntArg

End Sub

' Received events
Private Sub m_caoCtrl_OnMessage(ByVal pICaoMess As CAOLib.ICaoMessage)

    txtText1.Text = ""
    DoEvents

    Select Case pICaoMess.Number

        Case 103 ' Specified sampling frequency storage event
            If (VarType(pICaoMess.Value) And vbArray) = vbArray Then
                Dim i As Long
                txtText1.Text = "Msg=103 : "
                For i = 0 To UBound(pICaoMess.Value)
                    txtText1.Text = txtText1.Text & CStr(pICaoMess.Value(i)) & ", "
                Next
            End If

        Case 104 ' Overflow events
            txtText1.Text = "Msg=104 : Over flow error."

        Case 105 ' Sampling clock error event
            txtText1.Text = "Msg=105 : Sampling clock error."

        Case 106 ' AD conversion error error event
            txtText1.Text = "Msg=106 : AD conversion error."

    End Select

End Sub

```

### 3.4. Analog Input Automatic Sampling Sample (Advanced AI Input Function)

The following is a sample of acquiring two channels of AI sampling data in voltage or current values using only the AddController startup option setting (without using the Execute command).

#### List 3-4 SampleAiAutoSampling.frm

```

Private m_caoEng As CaoEngine
Private m_caoCtrls As CaoControllers
Private WithEvents m_caoCtrl As CaoController

Private Sub Form_Load()

    Dim vntArg As Variant

```

```
Set m_caoEng = New CaoEngine
Set m_caoCtrls = m_caoEng.Workspaces(0).Controllers
Set m_caoCtrl = m_caoCtrls.Add("SampleAiAutoSampling", _
    "CaoProv.CONTEC.AIO", "", _
    "DeviceName=AIO000, AiChannels=2, AiSamplingClock=10000,
AiAutoSampling=1:100:0x00070080")

End Sub

Private Sub Form_Unload(Cancel As Integer)

    If Not m_caoCtrl Is Nothing Then
        m_caoCtrls.Remove m_caoCtrl.Index
        Set m_caoCtrl = Nothing
    End If

End Sub

' Received events
Private Sub m_caoCtrl_OnMessage(ByVal pICaoMess As CAOLib.ICaoMessage)

    txtText1.Text = ""
    DoEvents

    Select Case pICaoMess.Number

        Case 103 ' Specified sampling frequency storage event
            If (VarType(pICaoMess.Value) And vbArray) = vbArray Then
                Dim i As Long
                txtText1.Text = "Msg=103 : "
                For i = 0 To UBound(pICaoMess.Value)
                    txtText1.Text = txtText1.Text & CStr(pICaoMess.Value(i)) & ", "
                Next
            End If

        Case 104 ' Overflow events
            txtText1.Text = "Msg=104 : Over flow error."

        Case 105 ' Sampling clock error event
            txtText1.Text = "Msg=105 : Sampling clock error."

        Case 106 ' AD conversion error error event
            txtText1.Text = "Msg=106 : AD conversion error."

    End Select

End Sub
```