HeartBeat Provider

ON-OFF Signal Generation Function

Version 1.1.3

User’s guide

August 08, 2022

[Remarks]

[Revision history]

Version	Date	Description
1.0.0	2016-11-15	First edition.
1.1.0	2018-05-02	To support the interval time measurement function (IntervalChecker) Addition of CaoExtension class and system variable was added.
1.1.1	2019-03-18	Internal logic correction.
1.1.2	2020-11-20	Speeded up the end process.
1.1.3	2022-08-08	Internal logic correction.

[Supported models]

Model	Version	Note
Independet	-	-

Contents

1. Introduction

This is a user’s guide of HeartBeat provider included in CAO provider enables living confirmation by repeating ON-OFF signal.

This guide explains about the function of provider and the method of being implemented.

2. Outline of provider

2.1. Outline

HeartBeat provider is a provider that enables to inform the ON-OFF signal with the designated cycle via OnMessage event, or provides the function that allows its status as the value of variable.

It is also possible to inform living confirmation to outside by using HeartBeat provider.

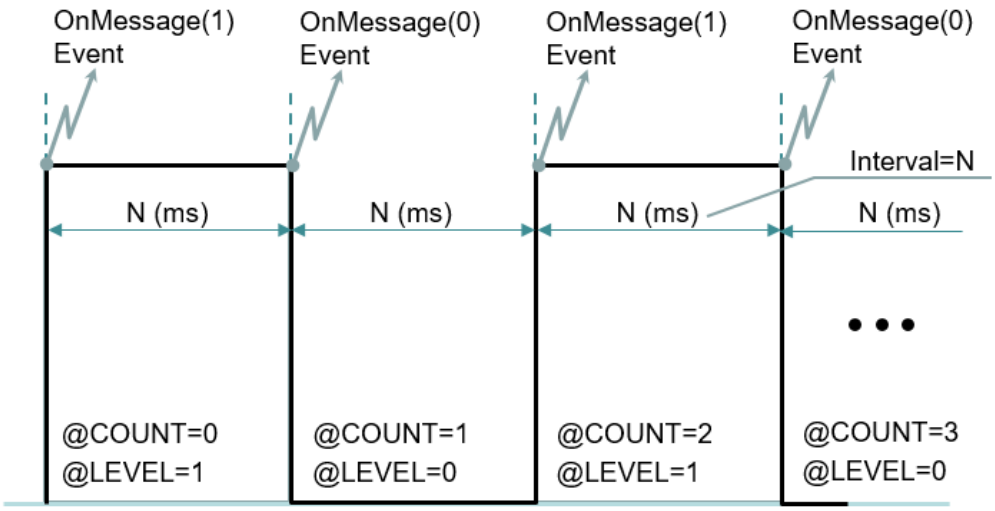


Table 2-1 HeartBeat provider

File name	CaoProvHeartBeat.dll
ProgID	CaoProv.HeartBeat
Registration [1]	regsvr32 CaoProvHeartBeat.dll
Deregistration	regsvr32 /u CaoProvHeartBeat.dll

[1] If it is installed with ORiN SDK, no need to register/delete manually

2.2. Method and Properties

2.2.1. CaoWorkspace::AddController method

HeartBeat provider needs to set the method of generating ON-OFF signal (HeartBeat), as well as repeat interval when generating Controller object.

```
AddController
(
    "<Controller name>",      // Controller name
    "CaoProv.HeartBeat",      // Provide name (fixed)
    "<Machine name >",        // Provider execution machine name
    "<Option>"    // Option character string
)
```

The following table lists the option character strings.

Table 2-2 Option character strings of CaoWorkspace::AddController

Option	Description
Interval=<Time of Beat interval>	Specify the interval between ON and OFF in ms. Valid range: 50 to 1000000 (default: 500)
AutoStart[=<Auto start>]	Specify if starts ON-OFF manually. Valid range: 0 or 1 (default: 1) 0 – OFF 1 – ON
Mode=<Motion mode>	Specify the motion mode. Valid range: 0 or 1 (default: 0) Repeat : 0 – ON → OFF Repeat : 1 – OFF → ON
Priority=<Thread priority> or @ThreadPriority=<Thread priority>	Specify the thread priority to generate ON - OFF signal (HeartBeat). Effective range: 0 to 6 (default: 3) 0:THREAD_PRIORITY_IDLE 1:THREAD_PRIORITY_LOWEST 2:THREAD_PRIORITY_BELOW_NORMAL 3:THREAD_PRIORITY_NORMAL 4:THREAD_PRIORITY_ABOVE_NORMAL 5:THREAD_PRIORITY_HIGHEST 6:THREAD_PRIORITY_TIME_CRITICAL

2.2.2. CaoController::AddVariable method

This method is able to specify the variable name shown in “Table 2-4 Controller class system variable list”. If you use other than those variable names, an error will be returned.

```
AddVariable
(
    "<Variable name >",     // Variable name
    "<Option>"              // Option character string（unused）
)
```

2.2.3. CaoController::get_VariableNames property

Obtain the list of system variable names shown in “Table 2-4

2.2.4. CaoController::OnMessage event

If the signal of ON/ OFF is changed while HeartBeat provider is operating, the state of ON/OFF will be informed as OnMessage event of CaoController class.

At this time, the value of Message::Value property, ON (1) or OFF (0) will be stored.

The following shows that Message:: Number property and its corresponding to the data type.

Table 2-3 Message:: Number property value and its corresponding to data type

Number property	Data classification	Data type
0	The change of the state	VT_I4

2.2.5. CaoController::AddExtension method

The HeartBeat provider uses the CaoExtension object to realize an interval time measurement function (IntervalChecker) to enable accuracy measurement of the generated ON - OFF signal.

To use the interval time measurement function (IntervalChecker), first execute AddExtension on the CaoController object to create a CaoExtension object.

The argument specification of AddExtension is shown as follows.

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

< bstrCtrlName > : [in] Object name
Specify an arbitrary name.

<bstrOption> : [in] Option string
Specify the following options in a comma-separated list:

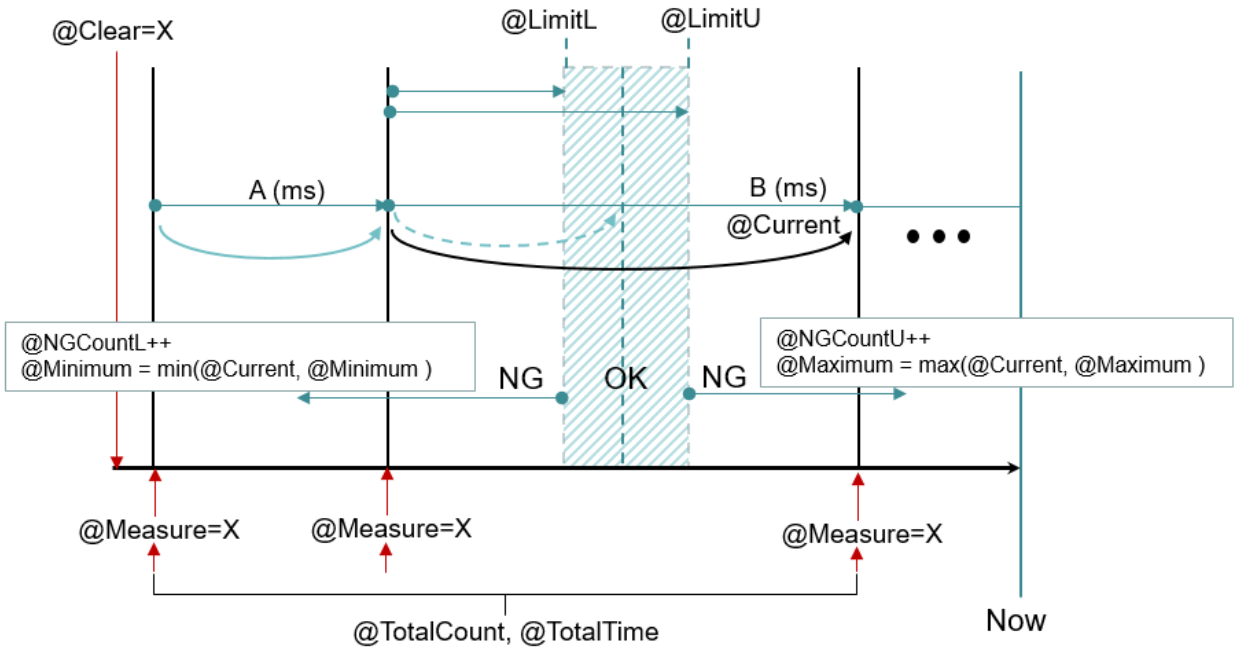
LimitU	:	Specify the allowable upper limit value (unit: millisecond) of the interval time. (Optional) The value can be an integer value and the range from 1 to 2147483647 can be specified. Please specify a value larger than LimitL. By default, 1000 is used as the default value. Example: Limit U = 500
LimitL	:	Specify the allowable lower limit value (unit: millisecond) of the interval time. (Optional) The value is an integer value and the range can be 0 to 2147483647. Please specify a smaller value than LimitU. By default, 0 is used as the default value. Example: Limit L = 100

Here is an example of executing the AddExtension method.

```
AddExtension
(
        // Object name = Chk1
        "Chk1",
        // Specify the lower limit value of the interval time as 100 ms and upper as 3000 ms
        " LimitL=100, LimitU=3000"
);
```

2.2.6. CaoExtension::AddVariable method

For the CaoExtension object, a dedicated variable is prepared for the interval time measurement function (IntervalChecker) to enable accuracy measurement of the ON - OFF signal, and interval time measurement can be performed by these accesses.



The argument specification of AddVariable is shown below.

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

< bstrCtrlName > : [in] Variable name

Specify the following system variable name or user variable name.

System variable : Specify the following reserved character string beginning with '@'.

@LimitU
@LimitL
@Current
@Maximum
@Minimum
@NGCountU
@NGCountL
@TotalCount
@TotalTime
@Measure
@Clear

@TickCount
@Serial

User variable : Specify an arbitrary character string that does not start with '@'. In this case, it is mandatory to specify Type of Option string.

<bstrOption> : [in] Option string

Specify the following options in a comma-separated list:

Type : For user variables, specify the system variable name to determine the behavior.
@LimitU
@LimitL
@Current
@Maximum
@Minimum
@NGCountU
@NGCountL
@TotalCount
@TotalTime
@Measure
@Clear
@TickCount
@Serial
This option has no effect on system variables.
Example:
Type = @Measure

Here is an example of executing the AddVariable method.

```
AddVaraible
(
    "@Measure",          // System variable
    ""                   //
);

AddVaraible
(
    "MyMeasure",         // User variable
    "Type=@Measure"      // Same as system variable @Measure
```

);

For details of the variables implemented in this CaoExtension class, refer to "Table 2-5 Extension class".

2.2.7. CaoExtension::get_VariableNames property

Obtain a list of system variable names shown in "Table 2-5 Extension class".

2.2.8. CaoVariable::put_Value property

This put_Value property can be valid exclusively to the variable which allows Put. (2.3 Variable list)

2.2.9. CaoVariable::get_Value property

This get_Value property can be valid exclusively to the variable which allows Get. (2.3 Variable list)

2.3. Variable list

2.3.1. Controller class

Table 2-4 Controller class system variable list

Variable name	Data type	Description	Attribute	
			get	put
@VERSION	VT_BSTR	Provider version	✓	-
@COUNT	VT_I4	Return the current total count of ON/OFF.	✓	-
@LEVEL	VT_I4	Return the current state. -1 – Undefined 0 – OFF 1 – ON	✓	-
@ACTIVATED	VT_I4	Obtain the current operating state and reflect its current setting. [Get] 0 – suspended, 1 – running [Put] 0 – deactivate request, 1 – activate request Once activate request has been done, the state of @COUNT and @LEVEL can be initialized.	✓	✓
@GLOBAL_TIME	VT_DATE	Returns the current global time (UTC). (Millisecond information is also included)	✓	-
@LOCAL_TIME	VT_DATE	Returns the current local time (UTC + time zone offset). (Millisecond information is also included)	✓	-
@TickCount	VT_UI4	Returns the elapsed time (milliseconds) since system power supply was inserted.	✓	-

2.3.2. Extension class

Table 2-5 Extension class system variable list

Variable name	Data type	Description	Attribute	
			get	put
@LimitU	VT_I4	It corresponds to the allowable upper limit value (unit: millisecond) of the interval time. If the write interval to @Measure variable exceeds this value, @NGCountU is incremented by +1. Corresponds to the LimitU option at extension class creation.	✓	✓
@LimitL	VT_I4	It corresponds to the allowable lower limit value (unit: millisecond) of the interval time. If the write interval to @Measure variable exceeds this value, @NGCountL is incremented by +1. Corresponds to the LimitL option at extension class creation.	✓	✓
@Current	VT_I4	It corresponds to the current interval time (unit: milliseconds). Returns the time of the write interval to the latest @Measure.	✓	-
@Maximum	VT_I4	Corresponds to the maximum interval time up to the present.	✓	-
@Minimum	VT_I4	Corresponds to the minimum interval time up to the present.	✓	-
@NGCountU	VT_I4	It corresponds to the total number exceeding the allowable upper limit value of the interval time.	✓	-
@NGCountL	VT_I4	Corresponds to the total number below the allowable lower limit of the interval time.	✓	-
@TotalCount	VT_I4	Corresponds to the number of measurements of the interval time. (Total number of writes to @Measure variable)	✓	-

@TotalTime	VT_UI4	Corresponds to the total time (unit: millisecond) from the measurement start time to the time when the last interval measurement was performed. After writing to @Clear, measurement begins after writing to the first @Measure.	✓	-
@Measure	Any	Corresponds to variables that specify the interval measurement timing. The interval for writing to this variable is the interval time. The result is stored in the following variable. @Current, @Maximum, @Minimum, @NGCountU, @NGCountL, @TotalCount, @TotalTime	-	✓
@Clear	Any	Clear the interval measurement and make measurement start again. The measurement result is cleared by writing an arbitrary value to this variable, and measurement starts from the next writing to @ Measure.	-	✓
@TickCount	VT_UI4	Returns the elapsed time (milliseconds) since system power supply was inserted.	✓	-
@Serial	VT_UI4	Returns consecutive values in order from 0. An internal counter is added by reading the value of this variable. It is reset by writing to @Clear and starts again from 0.	✓	-

3. Sample program

The following shows the sample of how HeartBeat beats.

List 3-1 Sample.frm

```
Dim caoEng As CaoEngine
Dim caoWs As CaoWorkspace
Dim WithEvents caoCtrl As CaoController

' Generates CAO engine
Private Sub Form_Load()
Set eng = New CaoEngine
Set caoWS = eng.Workspaces(0)
' Connect to HeartBeat
Set crdPingStatus = caoWS.AddController("Sample", "CaoProv.HeartBeat", "", "Interval=500")
End Sub

' Event procedure
Private Sub caoCtrl_OnMessage(ByVal pICaoMess As CAOLib.ICaoMessage)
Text1.Text = pICaoMess.Value
End Sub
```