HeartBeat プロバイダ

ON-OFF シグナル生成機能

Version 1.1.3

ユーザーズ ガイド

August 08, 2022

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0	2016-11-15	初版.
1.1.0	2018-05-02	インターバル時間測定機能（IntervalChecker）に対応. CaoExtension クラス及びシステム変数追加
1.1.1	2019-03-18	内部ロジックの修正.
1.1.2	2020-11-20	終了処理の高速化実施.
1.1.3	2022-08-08	内部ロジックの修正

【対応機器】

機種	バージョン	注意事項
非依存	-	-

目次

1. はじめに

本書は，生存確認を ON-OFF シグナルの繰り返し実行で実現する CAO プロバイダである HeartBeat プロバイダのユーザーズガイドです．

本書は，この HeartBeat プロバイダの機能と実装されているメソッドについて説明します．

2. プロバイダの概要

2.1. 概要

HeartBeat プロバイダは指定した周期で ON-OFF シグナルを OnMessage イベントで通知する，或いは変数の値として状態取得できる機能を提供するプロバイダです．

HeartBeat プロバイダを用いて，外部に対して生存確認通知する機能を実現できます．

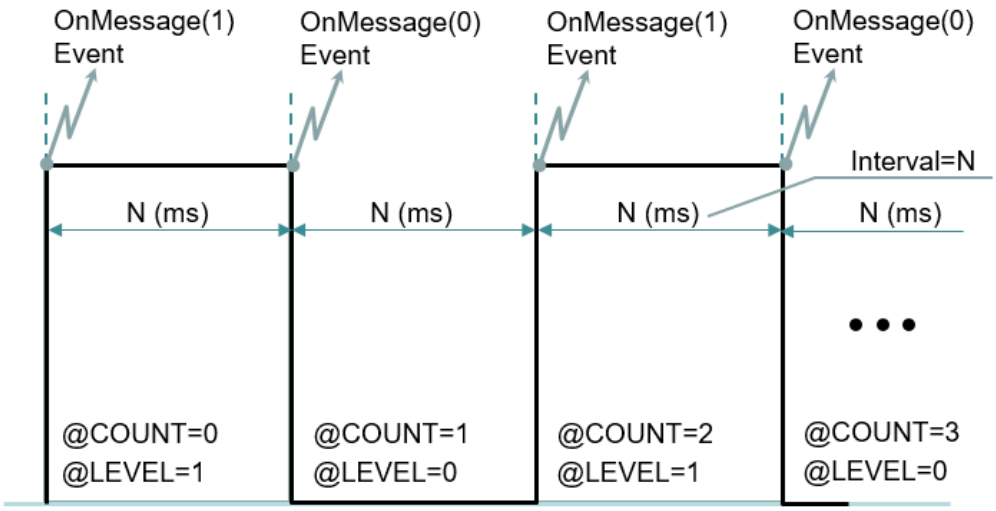


表 2-1 HeartBeat プロバイダ

ファイル名	CaoProvHeartBeat.dll
ProgID	CaoProv.HeartBeat
レジストリ登録[1]	regsvr32 CaoProvHeartBeat.dll
レジストリ登録の抹消	regsvr32 /u CaoProvHeartBeat.dll

[1] ORiN SDK でインストールした場合は手動で登録／抹消する必要はありません．

2.2. メソッド・プロパティ

2.2.1. CaoWorkspace::AddController メソッド

HeartBeat プロバイダでは Controller オブジェクトの生成時に ON-OFF シグナル(HeartBeat)を生成する方法と繰り返し間隔を設定します.

```
AddController
(
    "<コントローラ名>",    // コントローラ名
    "CaoProv.HeartBeat",   // プロバイダ名 (固定)
    "<マシン名>",          // プロバイダの実行マシン名
    "<オプション>"         // オプション文字列
)
```

以下にオプション文字列に指定するリストを示します.

表 2-2 CaoWorkspace::AddController のオプション文字列

オプション	意味
Interval=<Beat 間隔の時間>	ON と OFF の時間間隔をミリ秒で指定します. 有効範囲: 50～1000000（デフォルト: 500）
AutoStart[=<自動開始>]	自動で ON-OFF を開始するかの指定. 有効範囲: 0 または 1（デフォルト: 1） 0 – OFF 1 – ON
Mode=<動作モード>	動作モードを指定します. 有効範囲: 0 または 1（デフォルト: 0） 0 – ON → OFF の繰り返し 1 – OFF → ON の繰り返し
Priority=<スレッド優先度> または @ThreadPriority=<スレッド優先度>	ON-OFF シグナル(HeartBeat)を生成するスレッド優先度を指定します. 有効範囲: 0～6（デフォルト: 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 メソッド

このメソッドでは,「表 2-4 コントローラクラス システム変数一覧」にある変数名を指定することができます. それ以外の変数名を使用したときは, エラーを返します.

```
AddVariable
(
    "<変数名>",              // 変数名
    "<オプション>"           // オプション文字列（未使用）
)
```

2.2.3. CaoController::get_VariableNames プロパティ

「表 2-4 コントローラクラス システム変数一覧」に示しているシステム変数名の一覧を取得します.

2.2.4. CaoController::OnMessage イベント

HeartBeat プロバイダが動作中の間で ON または OFF のシグナルの変化があった場合に, CaoController クラスの OnMessage イベントとして ON または OFF の状態を通知します.

このとき, Message::Value プロパティの値は ON(1) または OFF(0)が格納されます.

以下に Message::Number プロパティとデータ種別の対応を示します.

表 2-3 Message::Number プロパティの値とデータ種別の対応

Number プロパティ	データ種別	データ型
0	状態変化	VT_I4

2.2.5. CaoController::AddExtension メソッド

HeartBeatプロバイダでは生成されたON-OFFシグナルの精度測定を可能にするためのインターバル時間測定機能(IntervalChecker)を CaoExtension オブジェクトで実現しています.

インターバル時間測定機能(IntervalChecker)を作成するためには CaoController オブジェクトに対して AddExtension を実行します.

以下に AddExtension の引数仕様を示します.

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

< bstrCtrlName > : [in] オブジェクト名
任意の名前を指定します.

<bstrOption> : [in] オプション文字列
以下のオプションをコンマ区切りで指定できます.

LimitU : インターバル時間の許容上限値(単位:ミリ秒)を指定します.(省略可)
値は整数値で範囲が 1 から 2147483647 が指定できます. LimitL より大きい値を指定して下さい.
省略時はデフォルト値として 1000 が使用されます.
例)
LimitU=500

LimitL : インターバル時間の許容下限値(単位:ミリ秒)を指定します.(省略可)
値は整数値で範囲が 0 から 2147483647 が指定できます. LimitU より小さい値を指定して下さい.
省略時はデフォルト値として 0 が使用されます.
例)
LimitL=100

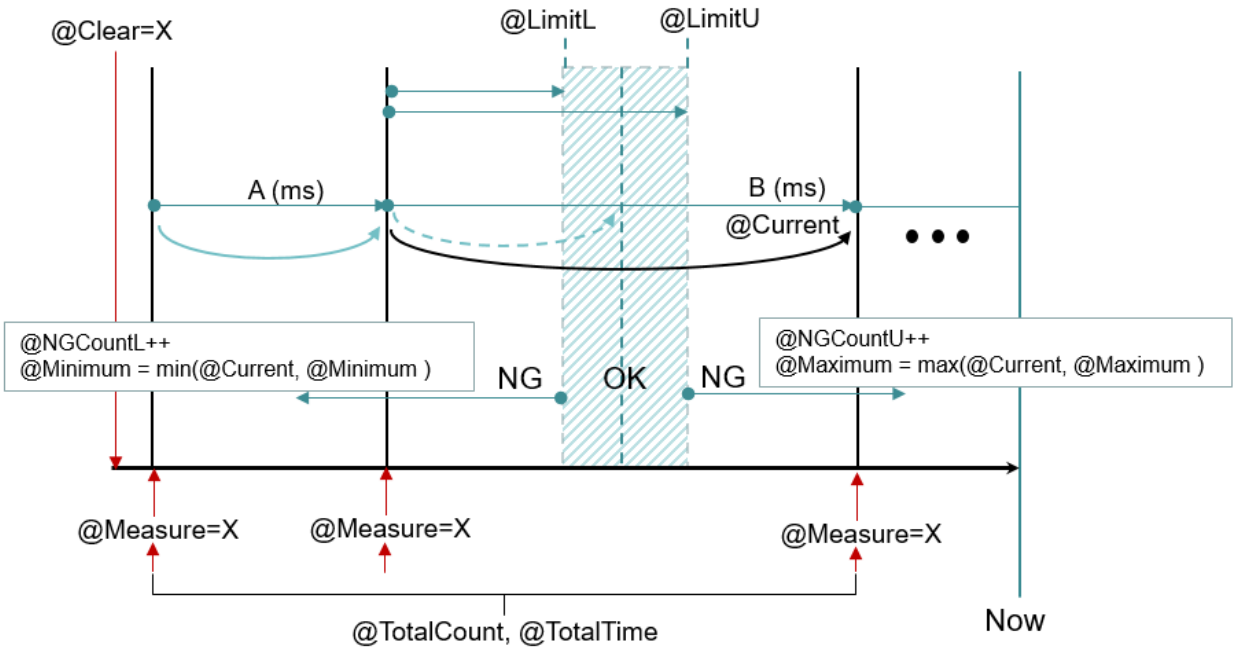
以下に AddExtension メソッドを実行するときの例を示します.

```
AddExtension
```

```
(
        // チェッカ名 ＝ Chk1
        "Chk1",
        // インターバル時間の下限値は 100ms, 上限値は 3000ms と指定
        " LimitL=100, LimitU=3000"
);
```

2.2.6. CaoExtension::AddVariable メソッド

CaoExtension オブジェクトでは ON-OFF シグナルの精度測定を可能にするためのインターバル時間測定機能（IntervalChecker）に対して，専用の変数を用意しており，これらのアクセスによりインターバル時間測定が行えます．



以下に AddVariable の引数仕様を示します．

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

< bstrCtrlName >　：　[in]　変数名

以下のシステム変数名或いはユーザ変数名を指定できます．

システム変数　：　‘@’で始まる以下の予約文字列を指定します．

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

@TickCount
@Serial

ユーザ変数 : '@'で始まらない任意の文字列を指定します. この場合は, オプション文字列の Type を指定することが必須です.

<bstrOption> : [in] オプション文字列
以下のオプションをコンマ区切りで指定できます.

Type : ユーザ変数の場合, 振る舞いを決めるためにシステム変数名を指定します.
@LimitU
@LimitL
@Current
@Maximum
@Minimum
@NGCountU
@NGCountL
@TotalCount
@TotalTime
@Measure
@Clear
@TickCount
@Serial
このオプションはシステム変数に対しては効果がありません.
例)
Type=@Measure

以下に AddVariable メソッドを実行するときの例を示します.

```
AddVaraible
(
    "@Measure",                 // システム変数
    ""                          //
);

AddVaraible
(
    "MyMeasure",                // ユーザ変数
    "Type=@Measure"             // システム変数の@Measure
```

);

CaoExtension オブジェクトで実装されているシステム変数は「表 2-5 拡張クラス システム変数一覧」を参照して下さい.

2.2.7. CaoExtension::get_VariableNames プロパティ

「表 2-5 拡張クラス システム変数一覧」に示しているシステム変数名の一覧を取得します.

2.2.8. CaoVariable::put_Value プロパティ

この put_Value プロパティは, Put が可能な変数に対してのみ実行することができます(2.3 変数一覧を参照).

2.2.9. CaoVariable::get_Value プロパティ

この get_Value プロパティは, Get が可能な変数に対してのみ実行することができます(2.3 変数一覧を参照).

2.3. 変数一覧

2.3.1. コントローラ(Controller)クラス

表 2-4 コントローラクラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@VERSION	VT_BSTR	プロバイダ バージョン.	○	-
@COUNT	VT_I4	現在の ON,OFF カウントの総数を返す.	○	-
@LEVEL	VT_I4	現在の状態を返す. -1 – 不定 0 – OFF 1 – ON	○	-
@ACTIVATED	VT_I4	現在の動作状態の取得と設定を反映します. [Get] 0 – 停止中, 1 - 動作中 [Put] 0 – 停止要求, 1 – 開始要求 開始要求を行うと, @COUNT, @LEVEL の状態は初期化されます.	○	○
@GLOBAL_TIME	VT_DATE	現在のグローバル時間(UTC)を返す. (ミリ秒の情報も含まれています)	○	-
@LOCAL_TIME	VT_DATE	現在のローカル時間(UTC+タイムゾーン・オフセット)を返す. (ミリ秒の情報も含まれています)	○	-
@TickCount	VT_UI4	システム電源挿入時からの経過時間(ミリ秒)を返します.	○	-

2.3.2. 拡張(Extension)クラス

表 2-5 拡張クラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@LimitU	VT_I4	インターバル時間の許容上限値(単位:ミリ秒)に対応します. @Measure 変数への書込み間隔がこの値を超えた場合は@NGCountU が+1 加算されます. 拡張クラス作成時の LimitU オプションに対応します.	○	○
@LimitL	VT_I4	インターバル時間の許容下限値(単位:ミリ秒)に対応します. @Measure 変数への書込み間隔がこの値を超えた場合は@NGCountL が+1 加算されます. 拡張クラス作成時の LimitL オプションに対応します.	○	○
@Current	VT_I4	現在のインターバル時間(単位:ミリ秒)に対応します. 最新の@Measure への書込み間隔の時間を返します.	○	-
@Maximum	VT_I4	現在までのインターバル時間の最大値に対応します.	○	-
@Minimum	VT_I4	現在までのインターバル時間の最小値に対応します.	○	-
@NGCountU	VT_I4	インターバル時間の許容上限値を上回った総数に対応します.	○	-
@NGCountL	VT_I4	インターバル時間の許容下限値を下回った総数に対応します.	○	-
@TotalCount	VT_I4	インターバル時間の測定回数に対応します. (@Measure 変数への書込みの総数)	○	-
@TotalTime	VT_UI4	測定開始時刻から最後にインターバル測定を行った時刻までの総時間(単位:ミリ秒)に対応します. @Clear への書込み後,最初の@Measure への書込みから測定開始になります.	○	-

@Measure	不定	インターバル測定タイミングを指定するための変数に対応します. この変数への任意の値書込みが行われた間隔がインターバル時間になります. その結果が下記変数に格納されます. @Current, @Maximum, @Minimum, @NGCountU, @NGCountL, @TotalCount, @TotalTime	-	○
@Clear	不定	インターバル測定をクリアし,再度測定開始可能にします. この変数への任意の値書込みで測定結果がクリアされ,次の@Measure への書込みから測定開始になります.	-	○
@TickCount	VT_UI4	システム電源挿入時からの経過時間(ミリ秒)を返します.	○	-
@Serial	VT_UI4	0 から順に連続した値を返します. この変数の値読出しで内部カウンターが加算されます. @Clear への書込みでリセットされ 0 から再び開始されます.	○	-

3. サンプルプログラム

以下に HeartBeat を打つサンプルを示します.

List 3-1　Sample.frm

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

' CAO エンジンの生成
Private Sub Form_Load()
Set eng = New CaoEngine
Set caoWS = eng.Workspaces(0)
' HeartBeat と接続
Set crdPingStatus = caoWS.AddController("Sample", "CaoProv.HeartBeat", "", "Interval=500")
End Sub

' イベントプロシージャ
Private Sub caoCtrl_OnMessage(ByVal pICaoMess As CAOLib.ICaoMessage)
Text1.Text = pICaoMess.Value
End Sub
```