SysmacStudio プロバイダ

OMRON 製 SysmacStudio

Version 1.0.3

ユーザーズ ガイド

March 8, 2024

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0.0	2012-06-22	初版.
1.0.0	2012-07-17	ドキュメントのバージョンルールを変更.
1.0.1	2012-12-11	Type オプション追加
1.0.2	2018-10-15	コマンドリファレンス修正. SysmacStudio の使い方追加. 接続失敗時の処理を修正.
1.0.3	2023-04-20	不要なファイル出力処理を削除.
	2024-03-08	誤記修正

【対応機器】

機種	バージョン	注意事項

目次

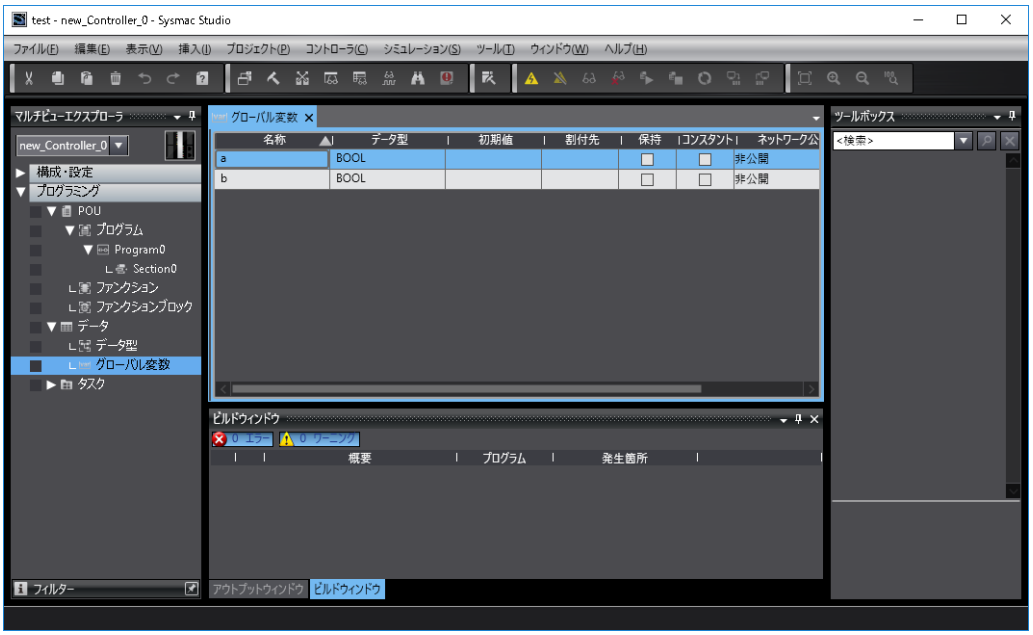
1. はじめに

本書は OMRON 製 SysmacStudio と通信を行う CAO プロバイダである SysmacStudio プロバイダのユーザーズガイドです.

2. Sysmac Studio の設定方法

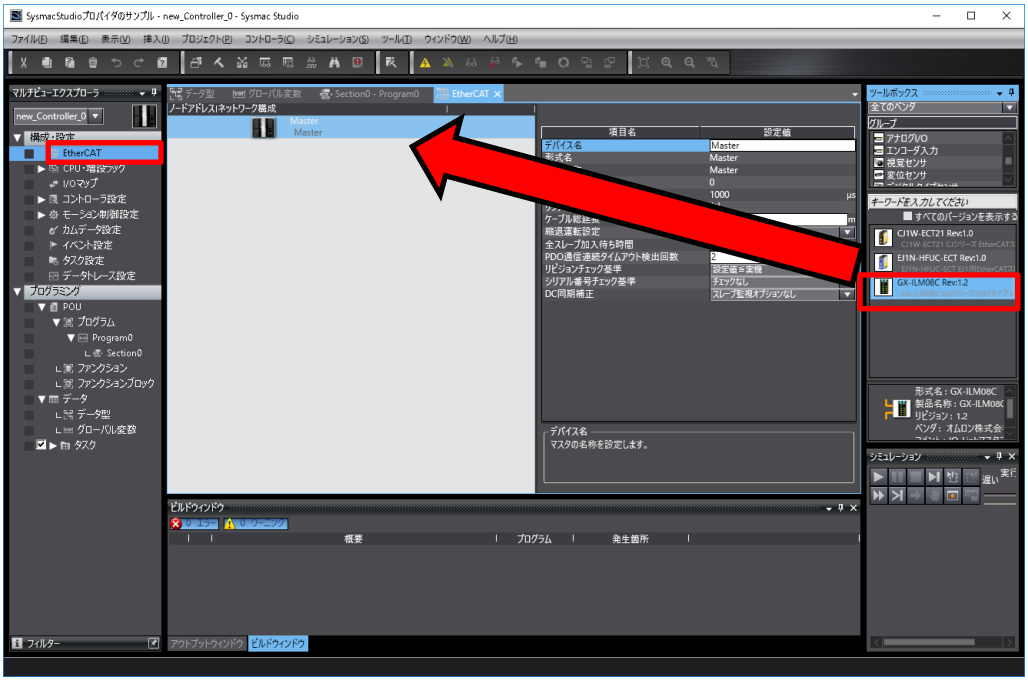
＜VAR 変数の設定方法＞

以下のようにグローバル変数を定義します．グローバル変数 a のパスは「VAR://a」となります．

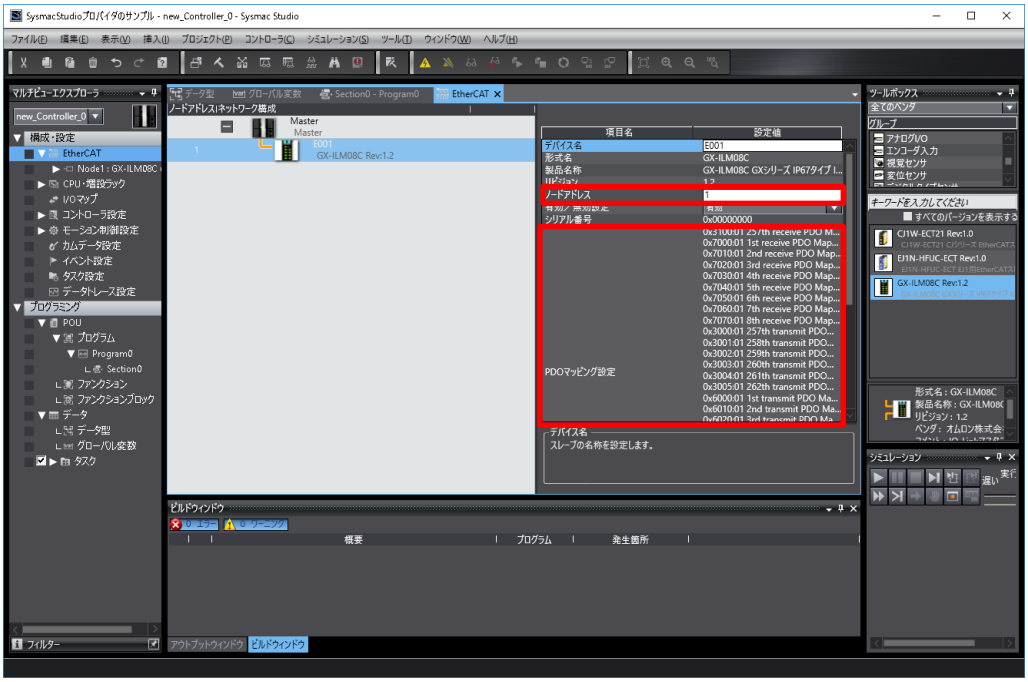


＜PDO 変数の設定方法＞

構成・設定＞EtherCAT を表示させます．ネットワーク構成に追加したい機器をツールボックスから選択し，ドラッグアンドドロップで Master 機器に追加させます．

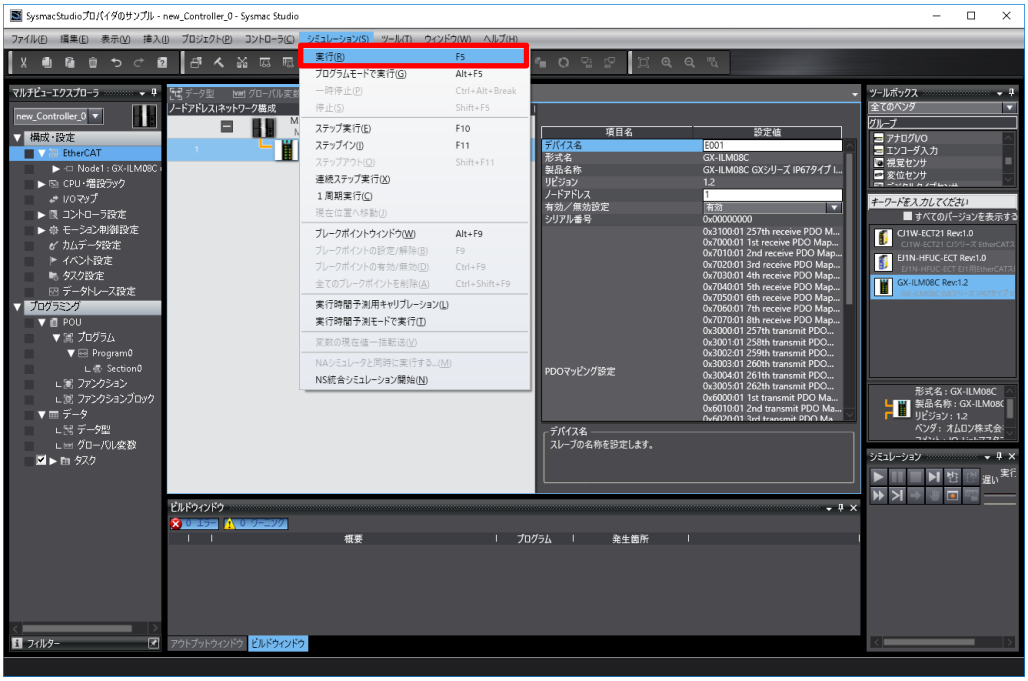


追加した機器のプロパティからノードアドレスを指定します. PDO マッピング設定で表示されている項目が使用可能な PDO 変数です.



＜シミュレーション実行＞

SysmacStudio プロバイダを使用するためには, シミュレーションを実行させる必要があります. ビルド完了後, ツールバーのシミュレーション＞実行を押下して, シミュレーションを開始させてください.



3. プロバイダの概要

3.1. 概要

SysmacStudio プロバイダのファイル形式は DLL(Dynamic Link Library)となっており, その詳細は表 3-1 のようになっています.

表 3-1 SysmacStudio プロバイダ

ファイル名	CaoProvSysStd.dll
ProgID	CaoProv.OMRON.Sysmac.Studio
レジストリ登録[1]	regsvr32 CaoProvSysStd.dll
レジストリ登録の抹消	regsvr32 /u CaoProvSysStd.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.

3.2. メソッド・プロパティ

3.2.1. CaoWorkspace::AddController メソッド

SysmacStudio への接続を行ないます.

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

bstrCtrlName : [in] コントローラ名
bstrProvName : [in] プロバイダ名. 固定値 =“ CaoProv.OMRON.Sysmac.Studio”.
bstrPcName : [in] プロバイダの実行マシン名
bstrOption : [in] オプション文字列

表 3-2 CaoWorkspace::AddController のオプション文字列

オプション	意味
Server=<IP アドレス>	接続する SysmacStudio のある PC の IP アドレスを指定します. （デフォルト:127.0.0.1）
Port[=<ポート番号>]	接続する SysmacStudio のある PC のポート番号を指定します. （デフォルト:7000）

3.2.2. CaoController::AddVariable メソッド

入力デバイスの情報を取得する CaoVariable オブジェクトを取得します. 使用できる入力デバイス名は, 3.3.1 を参照してください.

書式 AddVariable(<bstrName:BSTR>, [<bstrOption:BSTR>])
bstrCtrlName : [in] 変数名
bstrOption : [in] オプション文字列

表 3-3 CaoController::AddVariable のオプション文字列

オプション	意味
Path=<変数のパス>	変数の対象となるメモリのパス. このオプションを指定した場合, 生成された CaoVariable は<VAR 変数>として動作します. PDO オプションと同時に指定された場合はこのオプションは無視されます. 例)グローバル変数 a にアクセスする場合 “Path=VAR://a”

<table>
<tr><td>PDO=<ノードアドレス></td><td>変数の対象となる PDO のノードアドレス.
このオプションは以下の文法に従って指定します.
<NodeNumber>:0x<index>:<subIndex>
<NodeNumber> ECAT スレーブのノードアドレス (1～192)
<Index> PDO マップで定義されているインデックス番号.
16 進数 4 桁で指定します.
<SubIndex> PDO マップで定義されているサブインデックス番号
16 進数 2 桁で指定します.
このオプションを指定した場合, 生成された CaoVariable は<PDO 変数>として動作します. Path オプションと同時に指定された場合はこのオプションが優先されます.

例）ノードアドレス「1」, インデックス番号「0x7000」,
サブインデックス「0」の場合
“PDO = 1:0x7000:00”</td></tr>
<tr><td>Type=<データ型></td><td><VAR 変数>の値の取得及び設定で使用するデータ型を指定します.（デフォルト:VT_UI1）
このオプションは, Path オプションが指定されたときのみ有効になります.
このオプションには以下の文字列を指定します.
<table><tr><th>データ型</th><th>意味</th></tr><tr><td>VT_BOOL</td><td>BOOL</td></tr><tr><td>VT_I1</td><td>1 バイト符号あり整数</td></tr><tr><td>VT_UI1</td><td>1 バイト符号なし整数</td></tr><tr><td>VT_I2</td><td>2 バイト符号あり整数</td></tr><tr><td>VT_UI2</td><td>2 バイト符号なし整数</td></tr><tr><td>VT_I4</td><td>4 バイト符号あり整数</td></tr><tr><td>VT_UI4</td><td>4 バイト符号なし整数</td></tr><tr><td>VT_R4</td><td>4 バイト浮動小数点数</td></tr><tr><td>VT_R8</td><td>8 バイト浮動小数点数</td></tr></table>
例） WORD 型（2 バイト符号なし整数）の場合
”Type=VT_UI2”</td></tr>
</table>

3.2.3. CaoController::Execute メソッド

CaoController クラスに属するプロバイダ固有の拡張コマンドを実行します.

Execute メソッドの引数は, コマンドを BSTR, パラメータを VARIANT 配列で指定します.

書式　[<vntRet:VARIANT> =] Execute(<bstrCmd:BSTR > [,<vntParam:VARIANT>])

bstrCmd	:	[in]	コマンド名
vntParam	:	[in]	パラメータ
vntRet		[out]	返り値

実行時バインディングの機能を使って本クラスに定義していないメソッドを呼び出した場合, 次の仕様で Execute メソッドが自動的に呼ばれます.

vntRet = Obj.CommandName(Param1, Param2, ...)

↓

vntRet = Obj.Execute(“CommandName”, Array(Param1, Param2, ...))

1.第一引数にコマンド名が BSTR 文字列で渡る

2.第二引数に全パラメータが VARIANT 配列で渡る

バインディング機能を使用する場合, CaoConfig.exe > Cao Provider タブから, 指定するプロバイダの「Bind to a ‘Execute’ method」にチェックが入っていることを確認してください.

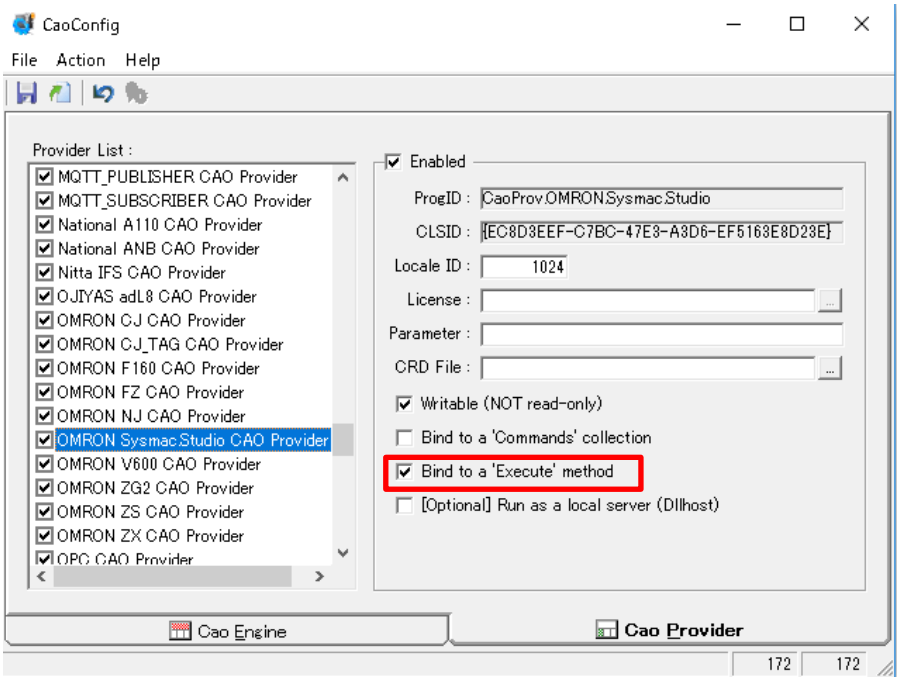


図 3-1 バインディング機能の設定

3.2.4. CaoController::Execute(“GetVarAddr”) コマンド

タグリビジョンと変数のアドレスリストを取得します. 戻り値の詳細は以下の表のとおりです.

書式　GetVarAddr (< Path1 >[, < Path2 >, < Path3 >,・・・])

< Path(n) > : [in] 変数のパス (VT_BSTR)
(n ≦ 100) 最大 100 個までパスを指定することができます.
戻り値 : [out] VarAddr(タグリビジョン＆アドレス情報)
(VT_ARRAY | VT_VARIANT: 2 要素)

表 3-4 GetVarAddr 戻り値のデータ型

データ型				説明	値範囲	
VT_ARRAY |VT_VARIANT				VarAddr	--	
0	VT_ARRAY | VT_UI1: 8 要素			タグリビジョン	0x0000 0000 0000 0000 ～ 0xffff ffff ffff fffe	
1	VT_ARRAY | VT_VARIANT : n 要素			AddrList 変数のアドレスリスト	--	
	0～n	VT_ARRAY | VT_UI4		アドレスの情報	--	
		0	VT_UI4	アドレスタイプ アドレスの意味を表す数値	1	<Path(n)>が変数(VAR_IN_OUT 以外)を表す場合. 取得されるアドレスは, 指定した変数の管理オブジェクトのアドレス.
					2	<Path(n)>がメモリを表す場合. 取得されるアドレスは, 物理アドレス.
					3	<Path(n)>が特殊変数を表す場合. 取得されるアドレスは, 指定した POU の管理オブジェクトのアドレス.
					4	<Path(n)> が変数(VAR_IN_OUT)を表す場合. 取得されるアドレスは, 指定した変数の管理オブジェクトのアドレ

データ型		説明	値範囲
			ス.
1	VT_UI4	アドレス	アドレスタイプ＝「2」の場合，<Path(n)>によって表現される先頭ビットが含まれる 1 ワードの先頭アドレス.
2	VT_UI4	ビットオフセット	アドレスタイプ＝「1」の場合，先頭変数からのビットオフセット(0 以上). アドレスタイプ＝「2」の場合，アドレスからのビットオフセット(0～15).
3	VT_UI4	ビットサイズ	

使用例

```
Dim vntAddr As Variant
vntAddr = caoCtrl.Execute("GetVarAddr",ARRAY("VAR://a","VAR://b"))
vntAddr = caoCtrl.GetVarAddr("VAR://a","VAR://b")
```

3.2.5. CaoController::Execute("ASyncReadMem") コマンド

指定されたアドレスに格納されている値を非同期モードで取得します. パラメータの<access>において，indeirect 指定をした場合において，同期用バッファがない場合には，direct アクセスを行います. <VarAddr>の<AddrList>と<Access>の要素数は同じ数にしてください.

書式　ASyncReadMem (<VarAddr>, <Access>, <SegmentCheck>)

< VarAddr > : [in] タグリビジョン＆アドレス情報 (VT_ARRAY |VT_VARIANT)
*詳細は GetVarAddr の戻り値を参照

< Access > : [in] 変数のアクセス方法リスト (VT_ARRAY |VT_UI4)

0	指定なし
1	indirect 変数の同期用バッファにアクセスする. 制御スレッドの途中でも常に最新の値が格納されている.
2	direct 変数の実体にアクセスする. 制御スレッドの終了後にのみ，最新の値が格納される.

< SegmentCheck > : [in] セグメントチェック (VT_UI4)
どのようなセグメントチェックを行うかを表す番号

0	チェックを行いません.
0 以外	実施されるチェック内容については, RM のコンポーネント I/F 仕様書を参照とする.

戻り値 : [out] 値のリスト
(VT_ARRAY | VT_VARIANT [VT_ARRAY | VT_UI1] : n 要素)
VarAddr で指定した要素数 n 分の戻り値が返ってきます. 変数の型が 2 バイト以上の場合, 1 バイトの配列として返ってきます. また, 指定した領域の開始位置, 又は終了位置がバイトの途中だった場合でも, ビットシフトなどの処理は行わず, 指定した領域を含むバイトデータを返します.

使用例

```
Dim vntAddr As Variant
Dim vntVal As Variant
Dim vntA,vntB As Variant
vntRetAddr = caoCtrl.GetVarAddr("VAR://a","VAR://b")
vntVal = caoCtrl.Execute("AsyncReadMem", Array(vntRetAddr, Array(0, 0), 0))
vntVal = caoCtrl.ASyncReadMem(vntRetAddr, Array(0, 0), 0)
vntA = vntVal(0)(0)
vntB = vntVal(1)(0)
```

3.2.6. CaoController::Execute("ASyncWriteMem") コマンド

指定されたアドレスに格納されている値を非同期モードで更新します. <access>において, indirect 指定をした場合において, 同期用のバッファがない場合には, direct アクセスを行います. <Value>と<AddrList>と<Access>の要素数は同じ数にしてください.

書式 ASyncWriteMem (<ValueList>, <VarAddr>, <Access>, <SegmentCheck>)

<ValueList> : [in] 値リスト
(VT_ARRAY | VT_VARIANT[VT_ARRAY | VT_UI1] :n 要素)

<VarAddr> : [in] タグリビジョン＆アドレス情報 (VT_ARRAY | VT_VARIANT)
*詳細は GetVarAddr の戻り値を参照

<Access> [in] 変数のアクセス方法リスト (VT_ARRAY |VT_UI4)

0	指定なし
1	indirect 変数の同期用バッファにアクセスする. 制御スレッドの途中でも常に最新の値が格納されている.
2	direct

	変数の実体にアクセスする. 制御スレッドの終了後にのみ, 最新の値が格納される.

<SegmentCheck>　[in] セグメントチェック (VT_UI4)

どのようなセグメントチェックを行うかを表す番号

0	チェックを行いません.
0 以外	実施されるチェック内容については, RM のコンポーネント I/F 仕様書を参照とする.

戻り値　:　[out] なし

使用例

```
Dim vntAddr As Variant
Dim vntVal As Variant
Dim vntA,vntB As Variant
vntRetAddr = caoCtrl.GetVarAddr("VAR://a","VAR://b")
vntVal = caoCtrl.Execute("ASyncWriteMem", _
                         Array(Array(Array(1), Array(1)), vntRetAddr, Array(0,0), 0))
vntVal = caoCtrl.ASyncWriteMem(Array(Array(1), Array(1)), vntRetAddr, Array(0,0), 0)
```

3.2.7. CaoController::Execute("ReadPDO") コマンド

軸変数を読み出します.

書式　ReadPDO (<NodeAddrList>)

<NodeAddrList>　:　[in] ノードアドレスリスト (VT_ARRAY | VT_VARIANT)

戻り値　:　[out] 値のリスト (VT_ARRAY | VT_BSTR)

表 3-5 ReadPDO 引数のデータ型

データ型			説明	値範囲
VT_ARRAY |VT_VARIANT :n 要素			NodeAddList	--
0 ~ n	VT_ARRAY | VT_VARIANT: 3 要素		ノードアドレス	--
	0	VT_UI1	ECAT スレーブのノードアドレス	1~192
	1	VT_UI2	PDO マップで定義されているインデックス番号	0x0000 ～ 0xFFFF
	2	VT_UI1	PDO マップで定義されているサブインデックス番号	0x00 ～ 0xFF

使用例

```
Dim vntVal As Variant
vntVal = caoCtrl.Execute("ReadPDO", Array(Array(1, 28672, 0), Array(1, 32768,0)))
vntVal = caoCtrl.ReadPDO(Array(1, 28672, 0), Array(1, 32768,0))
```

3.2.8. CaoController::Execute("WritePDO") コマンド

軸変数エリアにデータを書き込みます.

書式　WritePDO (<NodeAddrList>)

<NodeAddrList>　:　[in] ノードアドレスリスト (VT_ARRAY | VT_VARIANT)

戻り値　:　[out]なし

表 3-6 WritePDO 引数のデータ型

データ型			説明	値範囲
VT_ARRAY |VT_VARIANT :n 要素			NodeAddList	--
0 ~ n	VT_ARRAY | VT_VARIANT: 3 要素		ノードアドレス	--
	0	VT_UI1	ECAT スレーブのノードアドレス	1~192
	1	VT_UI2	PDO マップで定義されているインデックス番号	0x0000 ～ 0xFFFF
	2	VT_UI1	PDO マップで定義されているサブインデックス番号	0x00 ～ 0xFF
	3	VT_BSTR	書き込む値	

使用例

```
Dim vntVal As Variant
vntVal = caoCtrl.Excute("WritePDO",Array(1, 28672, 0, 1), Array(1, 32768, 0, 1))
vntVal = caoCtrl.WritePDO(Array(1, 28672, 0, 1), Array(1, 32768, 0, 1))
```

3.2.9. CaoController::Execute(“GetMode”) コマンド

CPU ユニットの動作モードを取得します.

書式　GetMode (<Section>)

<Section> : [in] 取得したい項目 (VT_BSTR)

“”	全ての現在の状態を取得します.
“mode”	WorkMode の値のみを取得します.
“state”	WorkState の値のみを取得します.
“error”	ErrorState の値のみを取得します.

戻り値 : [out] 取得した項目の状態(VT_BSTR)

取得する項目によって以下の状態を取得する.

ErrorState	NoError, HaltError, ContinuousError
WorkMode	RUN, PROGRAM
WorkState	InitializingWaitPLC, InitializingWaitIO, InitializingWaitControl, Initializing, ReadyWaitPLC, ReadyWaitIO, ReadyWawitControl, Ready, ProgramWaitPLC, ProgramWawitSubsystems, Program, RunWaitPLC, RunWaitSubsystems, Run, DownloadingWaitControl, DownloadinigWaitIO, DownloadingwaitPLC, Downloading, SubsystemsTerminate, Backup, Terminate

使用例

```
Dim vntRet As Variant
vntRet = caoCtrl.Execute(“GetMode”, “mode”)
vntRet = caoCtrl.GetMode(“mode”)
```

3.2.10. CaoController::Execute(“SetMode”) コマンド

CPU ユニットの動作モードを変更します.

書式　SetMode (<bstrSection >,<bstrState>)

< bstrSection > : [in] 変更対象 (VT_BSTR)

“mode”	動作モード
“error”	エラー状況

< bstrState > : [in] 状態 (VT_ BSTR)

指定できる状態は, <bstrSection>の指定で以下のように決められています.

“mode”	run,program
“error”	alert,error,warning,notice,clear

戻り値 : [out] なし

使用例

```
Dim vntRet As Variant
vntRet = caoCtrl.Execute(“SetMode”, Array(“mode”, “program”))
vntRet = caoCtrl.SetMode( “mode” ,” program” )
```

3.2.11. CaoController::Execute(“SetScanExec”) コマンド

Runtime シミュレータの実行状態を変化させます. Runtime シミュレータが運転モードの場合, SetScanExec(‘1’)コマンドを受信すると, 1 周期実行して待機します. 待機中の状態で, SetScanExec(‘0’)コマンドを受信する, または一定時間経過すると, 連続実行状態になります.

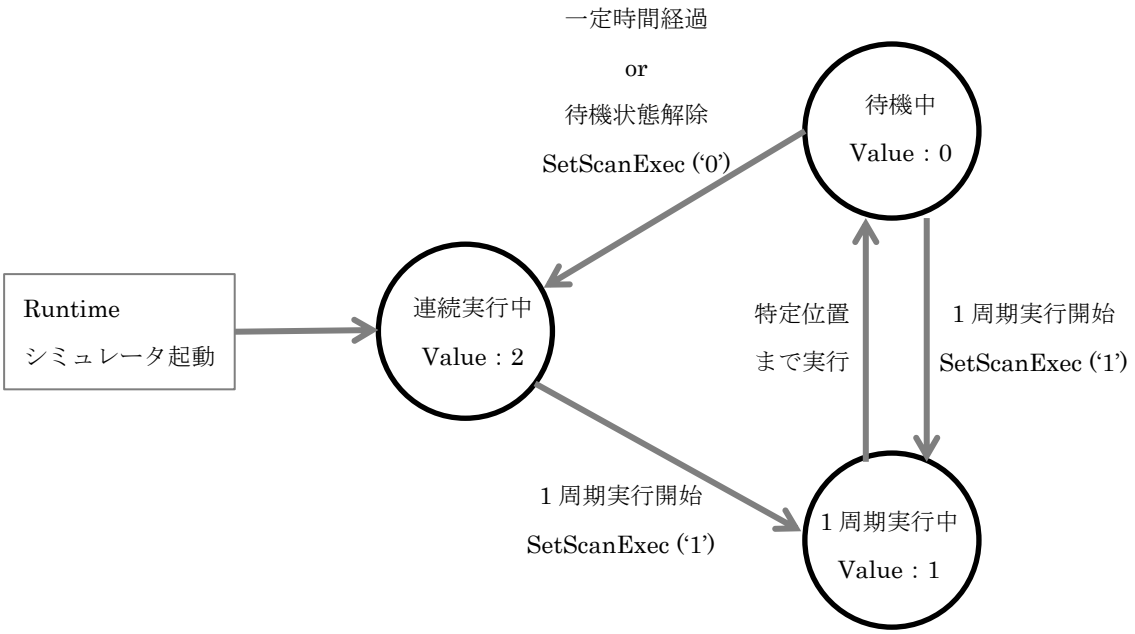


図 3-2 シミュレータの状態遷移図

書式　SetScanExec (<bstrValue>)

< bstrValue > : [in] 設定値 (VT_BSTR)

0	待機状態を解除します.
1	1 周期実行を開始し, 一定時間待機させます.

戻り値 : [out] なし

使用例

```
Dim vntRet As Variant
vntRet = caoCtrl.Execute("SetScanExec", 0)
vntRet = caoCtrl.SetScanExec(0)
```

3.2.12. CaoController::Execute("CheckScanState") コマンド

Runtime シミュレータの実行状態を取得します.

書式　CheckScanState ( )

引数 : [in] なし

戻り値 : [out] 実行状態 (VT_BSTR)

0	待機中
1	1 周期実行中
2	連続実行中

使用例

```
Dim vntRet As Variant
vntRet = caoCtrl.Execute("CheckScanState")
vntRet = caoCtrl.CheckScanState()
```

3.2.13. CaoVariable::get_Value プロパティ

3.2.2 で作成した変数の値を取得します. 取得する変数の詳細については, 3.3.1 を参照してください.

3.2.14. CaoVariable::put_Value プロパティ

3.2.2 で作成した変数の値を設定します. 取得する変数の詳細については, 3.3.1 を参照してください.

3.3. 変数一覧

3.3.1. コントローラクラス

表 3-7 コントローラクラス ユーザ変数一覧

変数名	データ型	説明	属性	
			get	put
<VAR 変数>	<Type>オプション | VT_ARRAY	メモリへの値の取得設定を行います. AddVariable 時に"Path"オプションを指定した場合は, この変数の動作になります. 配列のデータ型は, AddVariable 時に<Type>オプションで指定した型を使用します.	○	○
<PDO 変数>	VT_BSTR	PDO への値の取得・設定を行います. AddVariable 時に"PDO"オプションを指定した場合に, この変数の動作になります.	○	○

表 3-8 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
@MODE	VT_BSTR	WorkMode の値を取得する. 指定できる値は以下のものになります. "RUN", "PROGRAM"	○	○
@ERROR	VT_BSTR	ErrorState の値を取得する. "alert", "error", "warning", "notice", "clear"	○	○
@STATE	VT_BSTR	WorkState の値を取得する.	○	-

4. コマンドリファレンス

各コマンド一覧を示します.

表 4-1 CaoExtension::Execute コマンド一覧

コマンド	機能	ページ
GetVarAddr	変数のアドレスを取得する.	P. 11
ASyncReadMem	指定されたアドレスに格納されている値を取得する.	P. 12
ASyncWriteMem	指定されたアドレスに格納されている値を更新する.	P. 13
ReadPDO	軸変数を読み出す.	P. 14
WritePDO	軸変数エリアに値を書き込む.	P. 15
GetMode	CPU ユニットの動作モードを取得する	P. 16
SetMode	CPU ユニットの動作モードを変更する.	P. 17
SetScanExec	1 周期実行の開始を指示, あるいは指示を解除する.	P. 17
CheckScanState	Runtime シミュレータの実行状態を取得する.	P. 18

5. サンプルプログラム

List 5-1 SampleText.frm

```
Dim m_eng As CaoEngine
Dim m_ctrl As CaoController

Private Sub Form_Load()
    Set m_eng = New CaoEngine
    Set m_ctrl = m_eng.Workspaces(0).AddController("SYSMAC", "CaoProv.OMRON.Sysmac.Studio", "",
"server=127.0.0.1,port=7000")

    Dim vntVal As Variant
    Dim vntRetAddr1 As Variant
    Dim vntRetAddr2 As Variant
    vntRetAddr1 = m_ctrl.Execute("GetVarAddr", Array("VAR://TEST1"))
    vntRetAddr2 = m_ctrl.Execute("GetVarAddr", Array("VAR://TEST3"))

    vntVal = m_ctrl.Execute("ASyncWriteMem", Array(Array(Array(0)), vntRetAddr1))
    vntVal = m_ctrl.Execute("ASyncReadMem", Array(vntRetAddr1))

    vntVal = m_ctrl.Execute("ASyncWriteMem", Array(Array(Array(1)), vntRetAddr1))
    vntVal = m_ctrl.Execute("ASyncReadMem", Array(vntRetAddr2))

    vntVal = m_ctrl.Execute("ReadPDO", Array(Array(1, 24698, 0)))

    vntVal = m_ctrl.Execute("SetMode", Array("mode", "run"))
    vntVal = m_ctrl.Execute("GetMode", Array("Mode"))

    vntVal = m_ctrl.Execute("SetMode", Array("mode", "program"))
    vntVal = m_ctrl.Execute("GetMode", Array("Mode"))

    m_ctrl.Execute  "SetScanExec", 1

    vntVal = m_ctrl.Execute("CheckScanState")
    vntVal = m_ctrl.Execute("SetScanExec", 0)
    vntVal = m_ctrl.Execute("CheckScanState")

End Sub
```