Modbus.X プロバイダ

Modbus ASCII／RTU／TCP 通信

Version 1.0.7

ユーザーズ ガイド

May 24, 2021

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0.0	2015-1-10	初版
1.0.1.0	2015-4-22	サーバモード追加
1.0.2.0	2015-6-08	DI/DO 変数のアドレッシングを, データ幅に関わらず 1Bit に固定 保持/入力レジスタのアドレッシングを, データ幅に関わらず 16Bit に固定
1.0.2.1	2015-8-18	一部 補足および誤記修正
1.0.3.0	2015-8-21	CaoControlller::AddExtension 時のオプションに, “OffsetAddressZero”と“Endian” を追加
1. 0. 4. 0	2016-6-2	サンプル修正
1. 0. 5. 0	2016-9-6	CaoExtension:AddVariable 時のオプションに, ”VT”と”Elem” を追加
1. 0. 5.1	2016-9-19	用語定義追加. 旧 Modbus コマンド名称との対比追加.
1. 0. 5.2	2016-11-18	CaoExtension:AddVariable 時のオプションに, ”RcvPacketLen” と ”SndPacketLen” を追加
1. 0. 6. 0	2020-3-12	CaoWorkspace::AddController 時のオプション:PacketType に UDP モードを追加
1. 0. 7. 0	2021-5-24	Execute("ReceiveQuery")の型不一致を修正

【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 Modbus.X プロバイダのユーザーズガイドです.

この Modbus.X プロバイダを使用することで, CAO クライアントは Modbus プロトコル[1]の送受信を簡単に使用することができます.

本書は, この Modbus.X 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. 概要

1.1. 用語定義

本書では以降以下の用語の名称定義(統一)します.

- “コイル(Coil)” :“DO(Discrete Output) ”で統一.
- “入力ステータス”:“DI(Discrete Input)“ で統一.

[1] Modbus プロトコルは Modicon 社が 1979 年, 同社のプログラマブルロジックコントローラ (PLC) 向けに策定したシリアル通信である. 産業界におけるデ・ファクト標準の通信プロトコルとなり, 現在では産業用電子機器を接続する最も一般的手段となっている. (*wikipedia* より)
Modbus ASCII/RTU に関するプロトコル仕様は,「Modicon Modbus Protocol Reference Guide PI-MBUS-300 Rev. J」, Modbus TCP に関するプロトコル仕様は,「OPEN MODBUS/TCP SPECIFICATION Release 1.0, 29 March 1999」を参照してください.

2. プロバイダの概要

2.1. 概要

Modbus.X プロバイダは, Modbus プロトコルを送受信するプロバイダです.

通信デバイスが com(RS232C/RS485 等のシリアルデバイス(EIA-485))で通信モードがクライアントの時は Modbus マスタとして動作し, Modbus スレーブ機器に対してシリアル通信を行います. また, 通信モードがサーバの時は Modbus スレーブとして動作し, Modbus クライアント機器からのシリアル通信に応答します.

通信デバイスが eth(Ethernet)時で通信モードがクライアントの時は Modbus サーバ機器に対して TCP/IP 通信を行い, 通信モードがサーバの時は Modbus クライアント機器からの TCP/IP 通信に応答します.

後述の表記より, 通信デバイスに関わらず **マスタ＝クライアント**, **スレーブ＝サーバ** で名称統一します.

表 2-1 通信デバイスと通信モード

		通信モード	
Modbus 通信プロトコル	通信デバイス	クライアント	サーバ
ASCII, RTU	com	○	○
TCP	eth	○	○※

※ TCP プロトコル時の接続可能なクライアント数は 16 です.

Modbus.X 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 2-2 のようになっています.

表 2-2 Modbus.X プロバイダ

ファイル名	CaoProvModbusX.dll
ProgID	CaoProv.Modbus.X
レジストリ登録[2]	regsvr32 CaoProvModbusX.dll
レジストリ登録の抹消	regsvr32 /u CaoProvModbusX.dll

[2] 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. 実行モード

Modbus.X プロバイダでは実行モードとして同期モードと非同期モードの 2 つがあります. これは AddController の Sync オプションを指定することで切り替えることができます.

2.2.1. 非同期モード

2.2.1.1. クライアントモードの場合

CaoExtension::Execute() Modbus 機能対応コマンド(表 2-13 参照)の実行 又は CaoExtension::CaoVariable（ユーザ変数）のアクセス(表 2-15 参照)により, Modbus 要求（クエリ）メッセージを送信します. その後, サーバ機器から応答メッセージを受信した時に OnMessage イベントを発生させます.

2.2.1.2. サーバモードの場合

クライアント機器から要求（クエリ）メッセージを受信した時に, OnMessage イベントを発生させます. その後, OnMessage イベントで取得した, CaoMessage オブジェクトの CaoMessage::Reply()メソッドで, クライアント機器に対して応答メッセージを返信します.

2.2.2. 同期モード

2.2.2.1. クライアントモードの場合

CaoExtension::Execute() Modbus 機能対応コマンド(表 2-13 参照) 又は CaoExtension::CaoVariable（ユーザ変数）(表 2-15 参照)のアクセスにより Modbus 通信メッセージの送受信を行います.

2.2.2.2. サーバモードの場合

クライアント機器からの要求（クエリ）メッセージ受信は CaoController::Execute() ”ReceiveQuery”コマンド, クライアント機器への応答メッセージの送信は CaoController::Execute() ”SendReply”コマンドで行います.

2.3. メソッド・プロパティ

2.3.1. CaoWorkspace::AddController メソッド

Modbus.X プロバイダでは，AddController で接続パラメータを設定し通信の接続を行います．
このときオプションで通信形態，接続パラメータ，タイムアウトの設定等を指定します．
以下に AddController の引数仕様を示します．

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. Modbus.X".
bstrPcName : [in] プロバイダの実行マシン名
bstrOption : [in] 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．ここで，通信デバイスの欄に"-"が入っているオプションは，そのデバイスを指定したときに無視されます．

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

オプション	説明	サーバ		クライアント	
		eth	com	eth	com
Client [=True / False]	通信モード(クライアント／サーバ)の設定を行います． True: クライアント(デフォルト) False: サーバ 注: サーバモード時の接続可能クライアント数は 16 です．	○	○	○	○
ServerUnitAddress [=<サーバ機器アドレス>]	サーバ機器アドレス または ユニット識別子の設定を行います． com 時: サーバ機器アドレス(範囲:1-255) デフォルト:1 eth 時: ユニット識別子(範囲:-1, 0-255) デフォルト:-1 注： eth 時で"-1"はユニット識別子は任意となり，クライアント機器から送信される，全てのユニット識別子に対して受信通知します． 注: クライアントモード時，このオプションは無視されます．	○	○	-	-
Sync [=True / False]	同期モードの設定を行います． True: 同期モード(デフォルト) False: 非同期モード	○	○	○	○

Conn =<接続パラメータ>	必須. 通信形態と接続パラメータ. （参照 2.3.1.1）	○	○	○	○
PacketType [=<パケットパラメータ>]	Modbus 通信プロトコルデータタイプの設定をします. Conn オプションで指定した通信デバイスが"com"時: 0: RTU（デフォルト） 1: ASCII Conn オプションで指定した通信デバイスが"eth",かつクライアントモード時: 0: TCP（デフォルト） 1: UDP 注:サーバモード時 かつ Conn オプションで指定した通信デバイスが"eth"時は, このオプションは無視されます.	-	○	○	○
TcpConnectionTimeout [=<TCP 接続タイムアウト時間>]	TCP 接続タイムアウト時間[ms]の設定をします. 要求（クエリ）メッセージが設定時間内に受信されない場合, TCP 接続を切断します. 範囲:0～3600000（デフォルト:0） 0 はタイムアウト無効 注: クライアントモード時 または Conn オプションで指定した通信デバイスが"com"時は, このオプションは無視されます.	○	-	-	-
ReceiveQueryTimeout [=<クエリ受信タイムアウト時間>]	クエリ受信タイムアウト時間[ms]の設定をします. 範囲:0～100000（デフォルト:0） 注: クライアントモード時, このオプションは無視されます.	○	○	-	-
SendReplyTimeout [=<リプライ送信タイムアウト時間>]	リプライ送信タイムアウト時間[ms]の設定をします. 範囲:1～100000（デフォルト:1000） 注: クライアントモード時, このオプションは無視されます.	○	○	-	-
Timeout [=<送受信タイムアウト時間>]	送受信タイムアウト時間[ms]を設定します. 範囲:1～100000（デフォルト:1000） 注: サーバモード時, このオプションは無視されます.	-	-	○	○
Retry [=<リトライ回数>]	送受信時の通信リトライ回数の設定をします. 範囲:0～10（デフォルト:0） 注: サーバモード時, このオプションは無視されます.	-	-	○	○

OffsetAddressZero [=<True/False>]	各 Execute コマンドのパラメータで指定する, レジスタアドレス値 及び 各ユーザ変数名末尾で指定する?(レジスタアドレス値)のオフセット値を 0～ に設定します. True: レジスタアドレスのオフセット値は 0～ False: レジスタアドレスのオフセット値は 1～ (デフォルト: False) 注: サーバモード時, このオプションは無視されます. サーバ機器別に設定する場合は, CaoController::AddExtension 時の”OffsetAddressZero” オプション(表 2-4)を設定してください.	-	-	○	○
RtsTransmitDelayTime [=<送受信切換遅延時間>]	RTS 信号による送受信切換遅延時間[ms]の設定をします. 0:RTS 信号は常に ON(デフォルト) 1~100000:RTS 信号による送受信回路制御あり 送信直前に RTS ON → データ送信開始 → 送信完了※1 → 設定遅延時間経過後に RTS OFF 注: ・主に伝送モードが半二重の場合で, ソフトウエアによる送受信回路の切り替えが必要とされるハードウエア構成の場合に利用され, ”1”ms 以上に設定することにより, RTS 信号により送受信の切り替えを行います. ・本プロバイダ内部の上記※1(送信完了)の判定は, 通信デバイスドライバからの送信完了通知を起点にしているため, 実際の伝送路上の送信完了時機よりも早期になります. (実際の送信完了までの遅延時間の算出は, 通信ハードウエア側の FIFO を使用した場合, ベンダ依存となるため, FIFO 未使用で遅延時間を算出されることを推奨します) ・Conn オプションで指定したデバイスが, "eth"の時, このオプションは無視されます.	-	○	-	○
PollDelayTime [=<ポーリング遅延時間>]	ポーリング遅延時間の設定をします[ms]. 範囲:0～100000 (デフォルト:0) 注: サーバモード時, このオプションは無視されます.	-	-	○	○

Endian [=<Int>[:<Float>]]	32bit コマンド時のデータ転送順(エンディアン)を設定. <Int> 1: 32bit 整数/浮動小数点型ビックエンディアン (上位ワード→下位ワード) 0: 32bit 整数/浮動小数点型リトルエンディアン (下位ワード→上位ワード) (デフォルト) 但し, 浮動小数点型に関しては以下の <Float>指定がない場合のみ該当. <Float> 1: 32bit 浮動小数点型ビックエンディアン 0: 32bit 浮動小数点型リトルエンディアン 注: サーバモード時, このオプションは無視されます. サーバ機器別に設定する場合は, CaoController::AddExtension 時の”Endian”オプション(表 2-4)を設定してください.	-	-	○	○

2.3.1.1. Conn オプション

以下に Conn オプションの接続パラメータ文字列を示します. ここで角括弧(“[ ]”)内は省略可能を示します. また, 各パラメータの解説中の下線部はオプションを指定しなかったときのデフォルト値になります.

- **RS232C/RS485 デバイス**

“Conn=com:<COM Port>[:<BaudRate>[:<Parity>:<DataBits>:<StopBits>]]”

<COM Port>	:	COM ポート番号. ‘1’-COM1, ‘2’-COM2, …
<BaudRate>	:	通信速度. 1200, 4800, 9600, 19200, 38400, 57600, 115200, Max(UART ハードウェアに依存)
<Parity>	:	パリティ. ‘N’-NONE, ‘E’-EVEN, ‘O’-ODD.
<DataBits>	:	データビット数. ‘7’-7bit, ‘8’-8bit. 注: PacketType オプションが RTU モード時のデータビット数は 8 固定のため, '8'以外を指定した場合はエラーとなります.
<StopBits>	:	ストップビット数. ‘1’-1bit, ‘2’-2bit.

※フロー制御設定は”RtsTransmitDelayTime” オプションにより RTS 信号を利用するため, なし(NONE)固定となります.

- **Ethernet デバイス**

“Conn=eth:<IP Address>[:<Port No>]”

<IP Address>	:	<クライアントモード時> 接続先サーバ機器 IP アドレス <サーバモード時> 待受クライアント機器 IP アドレス 注:任意 IP アドレス (0.0.0.0)指定時の接続可能クライアント数は 16 です.
<Port No>	:	TCP 接続ポート番号. デフォルト:502

2.3.2. CaoController::Execute メソッド

使用できるコマンド名と詳細は 2.4.1 を参考にしてください.

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

bstrCommand	: [in] コマンド名
vntParam	: [in] パラメータ
pVal	: [out] 取得データ

2.3.3. CaoController::AddVariable メソッド

変数オブジェクトを作成します. 変数名には, 2.5.1 の変数のみ使用することができます.

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

bstrName : [in] 任意の名前

bstrOption : [in] オプション文字列(未使用)

2.3.4. CaoController::GetVariableNames プロパティ

2.5.1の変数名リストを取得します.

2.3.5. CaoController::AddExtension メソッド(クライアントモードのみ)

クライアントモード時, Modsbus サーバ機器と通信を行う CaoExtension を生成します.

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

bstrName : [in] 任意の名前

bstrOption : [in] オプション文字列

使用可能な “オプション文字列”を下表に記します.

表 2-4 CaoController: AddExtension のオプション文字列

オプション	意味
UnitAddress [=<機器アドレス>]	通信先のサーバ機器アドレス(com 時)又はユニット識別子(eth 時)を設定します. com 時: サーバ機器アドレス（範囲:0-255）デフォルト:1 eth 時: ユニット識別子 （範囲:0-255）デフォルト:0
OffsetAddressZero [=<True/False>]	サーバ機器アドレス別に, 各 Execute コマンドのパラメータで指定する, レジスタアドレス値 及び 各ユーザ変数名末尾で指定する?(レジスタアドレス値)のオフセット値を 0 ～ に設定します. 詳細は, 表 2-3 の” OffsetAddressZero”オプション参照. デフォルト: CaoWorkspace::AddController 時の” OffsetAddressZero”オプション設定値.
Endian [=<Int>[:<Float>]]	サーバ機器アドレス別に, 32bit コマンド時のデータ転送順(エンディアン)を設定します. 詳細は, 表 2-3 の”Endian”オプション参照. デフォルト: CaoWorkspace::AddController 時の”Endian”オプション設定値.

2.3.6. CaoExtension::GetID ソッド(クライアントモードのみ)

クライアントモード時, “UnitAddress”オプションで指定した, サーバの機器アドレスを取得します.

2.3.7. CaoExtension::Execute メソッド（クライアントモードのみ）

クライアントモード時，機能(Modbus ファンクション・コード)別に定義されたコマンド名により，サーバ機器との通信する Execute コマンドを実行します．

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

bstrCommand	: [in] コマンド名
vntParam	: [in] パラメータ
pVal	: [out] 取得データ

使用可能な “コマンド名” は表 2-13 を参照してください．

2.3.8. CaoExtension::AddVariable メソッド（クライアントモードのみ）

クライアントモード時，Modsbus サーバ機器と通信を行う変数オブジェクトを生成します．変数名には表 2-15 の変数のみ使用することができます．これら以外の変数名を指定したときは，このメソッドはエラーを返します．

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

bstrName	: [in] 任意の名前
bstrOption	: [in] 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 2-5 CaoExtension:AddVariable のオプション文字列

オプション	意味
UserVarWidth[=< ユーザ変数データ幅>]	ユーザ変数データ幅[bit]を設定します. [範囲] ・<bstrName> = DO? or DI? 時: 1(デフォルト), 8, 16, 32 [bit] ・<bstrName> = HRI? or IRI? 時: 16(デフォルト), 32 [bit] ・<bstrName> = HRF? or IRF? 時: 32(デフォルト)[bit] 固定 注:このオプションが指定されていない時で, "VT"オプションが指定された場合, このオプションは無視されます.
VT[=<変数型>]	データ型を指定します.(詳細は 表 2-6 参照) [対応 bstrName] <bstrName> = DO? or DI? or HRI? or IRI? 注:HRF? Or IRF?指定時は本オプションは無視され変数型は VT_R4 固定です. 省略時: 無効で"UserVarWidth"オプションが優先されます. 注:"UserVarWidth"オプションが指定されている時, このオプションは無視されます.
Elem[=<要素数>]	データ要素数を指定します. <bstrName>と"VT"オプション別の要素数の範囲は, 表 2-6 参照. "VT"オプションが BSTR 以外の変数型は配列型(VT_ARRAY)となり, 要素数を指定します. "VT"オプションが BSTR の変数型は VT_BSTR の文字列数(バイト単位)を指定します. (注:配列型(VT_ARRAY)の要素数ではありません) 要素数は 10 進数での指定の他, 16 進数での指定も可能です. 例) 0x0A, &h0A, 0AH [対応 bstrName] <bstrName> = DO? or DI? or HRI? or IRI? or HRF? or IRF? 省略時: 無効(変数型は非配列となります) ※VT オプション省略時のデータ型は以下なります. 各データ型の要素数範囲 および ?(アドレス)範囲は表 2-6 参照. [<bstrName> = DO? or DI? 時] UserVarWidth= 1: VT_ARRAY | VT_BOOL UserVarWidth= 8: VT_ARRAY | VT_UI1 UserVarWidth=16: VT_ARRAY | VT_UI2 UserVarWidth=32: VT_ARRAY | VT_UI4 [<bstrName> = HRI? or IRI? 時] UserVarWidth=16: VT_ARRAY | VT_UI2 UserVarWidth=32: VT_ARRAY | VT_UI4 ・・・ "Endian"オプション:有効 [<bstrName> = HRF? or IRFI? 時] VT_ARRAY | VT_R4 ・・・ "Endian"オプション:有効
RcvPacketLen[=< 受信パケット長>]	データ取得時の受信パケット長最大値を WORD 単位で指定します. 範囲: 4 ~ 125 省略 又は 範囲外: 125
SndPacketLen[=< 送信パケット長>]	データ設定時の送信パケット長最大値を WORD 単位で指定します. 範囲: 4 ~ 123 省略 又は 範囲外時: 123

表 2-6 VT オプションで指定可能なデータ型一覧

VT	データ型		説明
	“Elem“ 指定なし	“Elem“ 指定あり	
BIT	VT_UI1	VT_ARRAY | VT_UI1 [要素数範囲] “DO?”or“DI?”時:1~65536 “HRI?”or“IRI?”時:1~131072	データを 0/1 の 2 値に変換して書込み/読出します. Val==0:0, Val!=0:1 [?(アドレス)範囲] “OffsetAddressZero=False”時:1~65536 “OffsetAddressZero=True”時:0~65535
BOOL	VT_BOOL	VT_ARRAY | VT_ BOOL [要素数範囲] “DO?”or“DI?”時:1~65536 “HRI?”or“IRI?”時:1~65536	データを 0/-1 の 2 値に変換して書込み/読出します. Val==VARIANT_FALSE:0,Val!=VARIANT_FALSE:-1 [?(アドレス)範囲] “OffsetAddressZero=False”時:1~65536 “OffsetAddressZero=True”時:0~65535
BSTR	VT_BSTR	VT_BSTR [要素数範囲] “DO?”or“DI?”時:1~8192 “HRI?”or“IRI?”時:1~131072	BSTR を ASCII として書込み/読出します. “Elem“指定なし時: 1 文字(バイト) “Elem”指定あり時: 指定文字数(バイト単位) [?(アドレス)範囲] <“DO?”or“DI?”時> “OffsetAddressZero=False”時:1~65529 “OffsetAddressZero=True”時:0~65528 <“HRI?”or“IRI?”時> “OffsetAddressZero=False”時:1~65536 “OffsetAddressZero=True”時:0~65535
I1	VT_I1	VT_ARRAY | VT_I1 [要素数範囲] “DO?”or“DI?”時:1~8192 “HRI?”or“IRI?”時:1~131072	符号付 1 バイトデータとして書込み/読出します. [?(アドレス)範囲] <“DO?”or“DI?”時> “OffsetAddressZero=False”時:1~65529 “OffsetAddressZero=True”時:0~65528 <“HRI?”or“IRI?”時> “OffsetAddressZero=False”時:1~65536 “OffsetAddressZero=True”時:0~65535
I2	VT_I2	VT_ARRAY | VT_ I2 [要素数範囲] “DO?”or“DI?”時:1~4096 “HRI?”or“IRI?”時:1~65536	符号付 2 バイトデータとして書込み/読出します. [?(アドレス)範囲] <“DO?”or“DI?”時> “OffsetAddressZero=False”時:1~65521 “OffsetAddressZero=True”時:0~65520 <“HRI?”or“IRI?”時>

			“OffsetAddressZero=False”時:1~65536 “OffsetAddressZero=True”時:0~65535
I4	VT_I4	VT_ARRAY | VT_I4 [要素数範囲] “DO?”or“DI?”時:1~2048 “HRI?”or“IRI?”時:1~32768	符号付 4 バイトデータとして書込み/読出します. [?(アドレス)範囲] <“DO?”or“DI?”時 ・・・ ”Endian”オプション:無効> “OffsetAddressZero=False”時:1~65505 “OffsetAddressZero=True”時:0~65504 <“HRI?”or“IRI?”時 ・・・”Endian”オプション:有効> “OffsetAddressZero=False”時:1~65535 “OffsetAddressZero=True”時:0~65534
I8	VT_I8	VT_ARRAY | VT_I8 [要素数範囲] “DO?”or“DI?”時:1~1024 “HRI?”or“IRI?”時:1~16384	符号付 8 バイトデータとして書込み/読出します. [?(アドレス)範囲] <“DO?”or“DI?”時 ・・・ ”Endian”オプション:無効> “OffsetAddressZero=False”時:1~65473 “OffsetAddressZero=True”時:0~65472 <“HRI?”or“IRI?”時 ・・・”Endian”オプション:有効> “OffsetAddressZero=False”時:1~65533 “OffsetAddressZero=True”時:0~65532
UI1	VT_UI1	VT_ARRAY | VT_UI1 [要素数範囲] “DO?”or“DI?”時:1~8192 “HRI?”or“IRI?”時:1~131072	符号無 1 バイトデータとして書込み/読出します. [?(アドレス)範囲] <“DO?”or“DI?”時> “OffsetAddressZero=False”時:1~65529 “OffsetAddressZero=True”時:0~65528 <“HRI?”or“IRI?”時> “OffsetAddressZero=False”時:1~65536 “OffsetAddressZero=True”時:0~65535
UI2	VT_UI2	VT_ARRAY | VT_UI2 [要素数範囲] “DO?”or“DI?”時:1~4096 “HRI?”or“IRI?”時:1~65536	符号無 2 バイトデータとして書込み/読出します. [?(アドレス)範囲] <“DO?”or“DI?”時> “OffsetAddressZero=False”時:1~65521 “OffsetAddressZero=True”時:0~65520 <“HRI?”or“IRI?”時> “OffsetAddressZero=False”時:1~65536 “OffsetAddressZero=True”時:0~65535
UI4	VT_UI4	VT_ARRAY | VT_UI4 [要素数範囲] “DO?”or“DI?”時:1~2048 “HRI?”or“IRI?”時:1~32768	符号無 4 バイトデータとして書込み/読出します. [?(アドレス)範囲] <“DO?”or“DI?”時 ・・・ ”Endian”オプション:無効> “OffsetAddressZero=False”時:1~65505 “OffsetAddressZero=True”時:0~65504 <“HRI?”or“IRI?”時 ・・・”Endian”オプション:有効> “OffsetAddressZero=False”時:1~65535

			“OffsetAddressZero=True”時：0~65534
UI8	VT_UI8	VT_ARRAY | VT_UI8 [要素数範囲] “DO?”or“DI?”時：1~1024 “HRI?”or“IRI?”時：1~16384	符号無 8 バイトデータとして書込み/読出します. [?(アドレス)範囲] <“DO?”or“DI?”時 ・・・ ”Endian”オプション：無効> “OffsetAddressZero=False”時：1~65473 “OffsetAddressZero=True”時：0~65472 <“HRI?”or“IRI?”時 ・・・”Endian”オプション：有効> “OffsetAddressZero=False”時：1~65533 “OffsetAddressZero=True”時：0~65532
R4	VT_R4	VT_ARRAY | VT_R4 [要素数範囲] “DO?”or“DI?”時：1~2048 “HRI?”or“IRI?”時：1~32768	単精度浮動小数点(4 バイト)データとして書込み/読出します. [?(アドレス)範囲] <“DO?”or“DI?”時 ・・・ ”Endian”オプション：無効> “OffsetAddressZero=False”時：1~65505 “OffsetAddressZero=True”時：0~65504 <“HRI?”or“IRI?”時 ・・・”Endian”オプション：有効> “OffsetAddressZero=False”時：1~65535 “OffsetAddressZero=True”時：0~65534
R8	VT_R8	VT_ARRAY | VT_R8 [要素数範囲] “DO?”or“DI?”時：1~1024 “HRI?”or“IRI?”時：1~16384	倍精度浮動小数点(8 バイト)データとして書込み/読出します. [?(アドレス)範囲] <“DO?”or“DI?”時 ・・・ ”Endian”オプション：無効> “OffsetAddressZero=False”時：1~65473 “OffsetAddressZero=True”時：0~65472 <“HRI?”or“IRI?”時 ・・・”Endian”オプション：有効> “OffsetAddressZero=False”時：1~65533 “OffsetAddressZero=True”時：0~65532

2.3.9. CaoExtension::GetVariableNames プロパティ(クライアントモードのみ)

クライアントモード時, 表 2-15の変数名リストを取得します.

2.3.10. CaoVariable::get_Value プロパティ

変数に対応する情報を取得します. 各変数の実装状況および取得データについては, 2.5 を参照して下さい.

2.3.11. CaoVariable::put_Value プロパティ

変数に対応する情報を設定します. 各変数の実装状況および設定データについては, 2.5 を参照して下さい.

2.3.12. CaoController::OnMessage イベント

下表の OnMessage イベントが発生します.

表 2-7 CaoController::OnMessage イベント

No	Desciption	機能	各動作モード別対応可否								頁
			サーバ				クライアント				
			同期		非同期		同期		非同期		
			eth	com	eth	com	eth	com	eth	com	
1	REPLY_MSG	サーバ機器からの応答メッセージ受信時通知	－	－	－	－	－	－	○	○	P.20
2	QUERY_MSG	クライアント機器からの要求(クエリー)メッセージ受信時通知	－	－	○	○	－	－	－	－	P.21
3	IPINFO_MSG	TCP クライアント機器の接続/切断時通知	○	－	○	－	－	－	－	－	P.23

REPLY_MSG

発生条件 サーバ機器からの応答メッセージを受信したとき

Number 1

Description REPLY_MSG

Source
Destination CaoExtension 名 又は ”*ユーザ変数名*”

Value VT_ARRAY | VT_VARIANT

Array[0]: VT_BSTR
”*Modbus 機能対応 Execute コマンド名*”

Array[1]: VT_BSTR
送り元 IP アドレス 例) ”192.168.0.1” ・・・ com の時は未使用のため空文字(“”)

Array[2]: VT_I4
サーバ機器アドレス or ユニット識別子(範囲: 1-255)

Array[3]: VT_I4
ファンクションコード(範囲: 1-127)

Array[4]: VT_I4
ファンクションサブコード ・・・ 未使用時は 0 固定

Array[5]: VT_I4
実行結果(0 or 1: 正常, 0 未満: エラーコード)

Array[6]: VT_ARRAY | VT_**
各ファンクションコードの戻り値と同じ(ない場合は VT_EMPTY). 詳細は 2.4.4 参照.

説明 サーバ機器からの応答メッセージが受信されたことを通知します.
Value プロパティ内に要求(クエリ)機能に対する実行結果が格納されます.

注:このコマンドは, 各動作モード別で対応可否があります. 詳細は, 表 2-7 参照.

QUERY_MSG

発生条件 クライアント機器から要求(クエリー)メッセージを受信したとき

Number 2

Description QUERY_MSG

Source

Destination

Value VT_ARRAY | VT_VARIANT

Array[0]: VT_BSTR

送り元 IP アドレス　例) ”192.168.0.1”

Array[1]: VT_I4

サーバ機器アドレス or ユニット識別子（範囲: 0-255）

Array[2]: VT_I4

Modbus protocol Function Code. 詳細は下表(表 2-8)参照.

Array[3]: VT_ARRAY|VT_VARIANT

Modbus protocol Function Code 別パラメータ. 詳細は下表(表 2-8)参照.

表 2-8

機能	Array[2]	Array[3][0]	Array[3][1]	Array[3][2]
DO(Discrete Output) 複数読込	1 (0x01)	VT_I4: 開始アドレス	VT_I4: 読込み点数	VT_EMPTY: なし
DI(Discrete Input) 複数読込	2 (0x02)	VT_I4: 開始アドレス	VT_I4: 読込み点数	VT_EMPTY: なし
保持レジスタ(16bit) 複数読込	3 (0x03)	VT_I4: 開始アドレス	VT_I4: 読込みデータ数	VT_EMPTY: なし
入力レジスタ(16bit) 複数読込	4 (0x04)	VT_I4: 開始アドレス	VT_I4: 読込みデータ数	VT_EMPTY: なし
例外ステータス 状態読込	7 (0x07)	VT_EMPTY: なし		
DO(Discrete Output) 複数書出	15 (0x0F)	VT_I4: 開始アドレス	VT_I4: 書出し点数	VT_ARRAY｜VT_BOOL 書出しデータ
保持レジスタ(16bit) 複数書出	16 (0x10)	VT_I4: 開始アドレス	VT_I4: 書出しデータ数	VT_ARRAY｜VT_I2 書出しデータ

説明 クライアント機器から要求(クエリ)メッセージが受信されたことを通知します.

受信通知可能な Modbus protocol Function Code は, 表 2-9 となります.

通常 CAO クライアントは Value プロパティ内の要求機能に対する処理を実施後, Message::Reply()メソッドを使用して”SendReplyTimeout”オプションで設定された時間内に, 応答メッセージ(引数は CaoController::Execute "SendReply”コマンド(P.26)と同じ)を返信する必要があります.

”SendReplyTimeout”オプションで設定された時間経過後に Message::Reply()メソッドを

実行した場合，エラーが返ります．

注:このコマンドは，各動作モード別で対応可否があります．詳細は，表 2-7 参照．

表 2-9 受信通知可能 Modbus ファンクション・コード（サーバモードのみ）

機能	Modbus プロトコル上				通知メッセージ (QUERY_MSE) or 受信コマンド (ReceiveQuery)			自動応答	
	Broadcast		Function Code (HEX)	Sub Code	通知(受信)可否	通知 FunctionCode(HEX)	通知 Sub Code	正常応答※1	例外応答※2
	TCP	ASCII / RTU							
DO(Discrete Output)複数読込	×	×	1(0x01)	—	○	1(0x01)	—	—	○
DI(Discrete Input)複数読込	×	×	2(0x02)	—	○	2(0x02)	—	—	○
保持レジスタ(16bit)複数読込	×	×	3(0x03)	—	○	3(0x03)	—	—	○
入力レジスタ(16bit)複数読込	×	×	4(0x04)	—	○	4(0x04)	—	—	○
DO(Discrete Output)単一書出	×	○	5(0x05)	—	○	15(0x0F)※6	—	—	○
保持レジスタ(16bit)単一書出	×	○	6(0x06)	—	○	16(0x10)※3	—	—	○
例外ステータス状態読込	×	×	7(0x07)	—	○	7(0x07)	—	—	○
診断 クエリデータのエコーバック	×	×	8(0x08)	0	—	—	—	○	○
診断 その他	×	×		1~	—	—	—	—	○
DO(Discrete Output)複数書出	×	○	15(0x0F)	—	○	15(0x0F)	—	—	○
保持レジスタ(16bit)複数書出	×	○	16(0x10)	—	○	16(0x10))	—	—	○
保持レジスタ AND/OR マスク書出	×	×	22(0x16)	—	○	2 回通知 (3(0x03) の後 16(0x10))※4	—	—	○
保持レジスタ複数読込書出	×	×	23(0x17)	—	○	2 回通知 (16(0x10) の後 3(0x03))※5	—	—	○
その他			上記以外	—	—	—	—	—	○

※1: 通知せずに自動で正常応答(リプライ)メッセージを返信します．
※2: 通知せずに自動で例外応答(リプライ)メッセージを返信します．
※3: Modbus プロトコル上の FunctionCode は 6 ですが，通知 FunctionCode は 16 です．書出データ数(Count)は 1 固定です．
※4: Modbus プロトコル上の FunctionCode は 22 ですが，通知 FunctionCode は 2 回に分けられ，1 回目が 3 で読込値に対して AND/OR 演算をした結果(値)を，2 回目の 16 で書出通知します．書出データ数(Count)は 1 固定です．
※5: Modbus プロトコル上の FunctionCode は 23(0x17)ですが，通知 FunctionCode は 2 回に分けられ，1 回目が 16 で，2 回目が 3 です．
※6: Modbus プロトコル上の FunctionCode は 5 ですが，通知 FunctionCode は 10 です．書出データ数(Count)は 1 固定です．

IPINFO_MSG

発生条件 TCP クライアント機器が, 接続 または 切断したとき

Number 3

Description IPINFO_MSG

Source

Destination

Value VT_ARRAY | VT_VARIANT

Array[0]: VT_BOOL

接続ステータス: 接続時(VARIANT_TRUE), 切断時(VARIANT_FALSE)

Array[1]: VT_BSTR

接続 または 切断した, IP アドレス 例) ”192.168.0.1”

Array[2]: VT_ARRAY|VT_VARIANT

“@IpInfo”システム変数と同じ,

現在接続されているクライアント機器の, IPアドレスとポート番号を取得します.

詳細は, “@IpInfo”システム変数(表 2-14) 参照.

説明 TCP クライアント機器が接続 または 切断したことを通知します.

[接続条件]

TCP クライアント機器からの接続要求があった時 且つ

現在接続している TCP クライント機器数が 16 台以下の時

[切断条件]

・TCP クライアント機器から切断要求があった時.

・” TcpConnectionTimeout”オプションで, TCP 接続タイムアウト時間[ms]を, 無効(0)以外に設定した場合で, TCP クライアント機器が接続してから, 要求(クエリ)メッセージが, 設定した時間内に受信されない場合.

・要求(クエリ)メッセージの受信長が 6Byte 未満の時.

・返信(リプライ)メッセージの送信に失敗した時.

注:このコマンドは, 各動作モード別で対応可否があります. 詳細は, 表 2-7 参照.

2.4. コマンド一覧

2.4.1. CaoController クラス

表 2-10 CaoController::Execute コマンド一覧

コマンド名	機能	各動作モード別使用可否								頁
		サーバ				クライアント				
		同期		非同期		同期		非同期		
		eth	com	eth	com	eth	com	eth	com	
ProviderCancel	キャンセル状態に設定	○	○	○	○	○	○	○	○	P.24
ProviderClear	キャンセル状態の解除	○	○	○	○	○	○	○	○	P.24
ReceiveQuery	要求(クエリ)メッセージ受信	○	○	－	－	－	－	－	－	P.25
SendReply	応答メッセージ送信	○	○	－	－	－	－	－	－	P.26

2.4.2. CaoController::Execute コマンド詳細

ProviderCancel

全動作モード使用可能

構文 *object*.ProviderCancel()

引数 なし

戻り値 なし

説明 プロバイダをキャンセル状態に設定します.
キャンセル状態の間は, 送受信の実行を中断します.
キャンセル状態を解除する場合は, “ProviderClear”コマンドを実行してください.

ProviderClear

全動作モード使用可能

構文 *object*.ProviderClear()

引数 なし

戻り値 なし

説明 プロバイダのキャンセル状態を解除します.

ReceiveQuery

サーバモード時のみ使用可能

構文

object.ReceiveQuery()

引数

なし

戻り値

<Data> = VT_ARRAY | VT_VARIANT または VT_EMPTY

Array[0]: VT_BSTR

送り元 IP アドレス　例) ”192.168.0.1”

Array[1]: VT_I4

サーバ機器アドレス or ユニット識別子（範囲: 0-255）

Array[2]: VT_I4

Modbus protocol Function Code. 詳細は下表(表 2-11)参照.

Array[3]: VT_ARRAY|VT_VARIANT

Modbus protocol Function Code 別パラメータ. 詳細は下表(表 2-11)参照.

表 2-11

機能	Array[2]	Array[3][0]	Array[3][1]	Array[3][2]
DO(Discrete Output) 複数読込	1 (0x01)	VT_I4: 開始アドレス	VT_I4: 読込み点数	VT_EMPTY: なし
DI(Discrete Input) 複数読込	2 (0x02)	VT_I4: 開始アドレス	VT_I4: 読込み点数	VT_EMPTY: なし
保持レジスタ(16bit) 複数読込	3 (0x03)	VT_I4: 開始アドレス	VT_I4: 読込みデータ数	VT_EMPTY: なし
入力レジスタ(16bit) 複数読込	4 (0x04)	VT_I4: 開始アドレス	VT_I4: 読込みデータ数	VT_EMPTY: なし
例外ステータス 状態読込	7 (0x07)	VT_EMPTY: なし		
DO(Discrete Output) 複数書出	15 (0x0F)	VT_I4: 開始アドレス	VT_I4: 書出し点数	VT_ARRAY | VT_BOOL 書出しデータ
保持レジスタ(16bit) 複数書出	16 (0x10)	VT_I4: 開始アドレス	VT_I4: 書出しデータ数	VT_ARRAY | VT_I2 書出しデータ

説明

クライアント機器からの要求(クエリ)メッセージを受信します.

受信通知可能な Modbus protocol Function Code は, 表 2-9 となります.

要求(クエリ)メッセージの先頭バイトが, ” ReceiveQueryTimeout”オプションで設定した時間経過しても,受信されていない時は処理を返します. その際 <Data>は VT_EMPTY となります.

通常 CAO クライアントは<Data>内の要求機能に対する処理を実施した後, SendReply()コマンドを使用して”SendReplyTimeout”オプションで設定された時間内に, 応答メッセージ(詳細は SendReply (P.26)参照)を返信する必要があります.

注:このコマンドは, 各動作モード別で使用可否があります. 詳細は, 表 2-10 参照.

SendReply

サーバモード時のみ使用可能

構文

object.SendReply(<Data>)

引数

<Data> = VT_ARRAY | VT_VARIANT

Array[0]: VT_I4

通知 Modbus FunctionCode(表 2-11 参照)

Array[1]: VT_BOOL

処理結果(VARIANT_TRUE:正常, VARIANT_FALSE:異常)

Array[2]: 実行結果が正常の時, 通知 Modbus FunctionCode により, 下表のデータ型を設定. 実行結果が異常の時は VT_EMPTY を設定してください.

表 2-12

機能	通知 Function Code (HEX)	データ型	内容
DO(Discrete Output)複数読込	1(0x01)	VT_ARRAY | VT_BOOL	読み込みデータ
DI(Discrete Input)複数読込	2(0x02)	VT_ARRAY | VT_BOOL	読み込みデータ
保持レジスタ(16bit)複数読込	3(0x03)	VT_ARRAY | VT_I2	読み込みデータ
入力レジスタ(16bit)複数読込	4(0x04)	VT_ARRAY | VT_I2	読み込みデータ
例外ステータス状態読込	7(0x07)	VT_UI1	例外ステータス
DO(Discrete Output)複数書出	15(0x0F)	VT_EMPTY	データなし
保持レジスタ(16bit)複数書出	16(0x10)	VT_EMPTY	データなし

戻り値

なし

説明

クライアント機器へ応答メッセージを送信します.

通常 CAO クライアントは ReceiveQuery()コマンドで受信した要求機能に対する処理を実施後, " SendReplyTimeout"オプションで設定された時間内に, 本コマンドを使用して応答メッセージを返信する必要があります.

"SendReplyTimeout"オプションで設定された時間経過後に本コマンドを実行した場合, エラーが返ります.

通信デバイスが com 時, ReceiveQuery()コマンドで受信したサーバ機器アドレスが, ブロードキャスト指定(0)時は, 本コマンドを実行する必要はありません. 仮に実行してもクライアント機器への応答メッセージは送信されません. また, 処理結果(Array[1])を異常(VARIANT_FALSE)に設定した場合も同様に, 送信されません.

注: このコマンドは, 各動作モード別で使用可否があります. 詳細は, 表 2-10 参照.

2.4.3. CaoExtension クラス（クライアントモードのみ）

下表に，クライアントモード時の Modbus 機能対応コマンドを記します．

表 2-13 CaoExtension::Execute Modbus 機能対応コマンド一覧（クライアントモードのみ）

コマンド名 （旧 Modbus プロバイダ対応コマンド名）	Modbus protocol				機能	頁
	Broadcast		Function Code (HEX)	Sub Code		
	TCP	ASCII / RTU				
ReadMultipleDiscreteOutputs (ReadCoilStatus)	×	×	1(0x01)	-	DO(Discrete Output)複数読込	P.28
ReadMultipleDiscreteInputs (ReadInputStatus)	×	×	2(0x02)	-	DI(Discrete Input)複数読込	P.28
ReadMultipleHoldingRegisters (ReadHoldingRegister)	×	×	3(0x03)	-	保持レジスタ(16bit)複数読込	P.28
ReadMultipleHoldingRegistersLongInt （なし）					保持レジスタ(32bit 整数型)複数読込	P.29
ReadMultipleHoldingRegistersFloat （なし）					保持レジスタ(32bit 浮動小数点型)複数読込	P.29
ReadMultipleInputRegisters (ReadInputRegister)	×	×	4(0x04)	-	入力レジスタ(16bit)複数読込	P.30
RreadMultipleInputRegistersLongInt （なし）					入力レジスタ(32bit 整数型)複数読込	P.30
RreadMultipleInputRegistersFloat （なし）					入力レジスタ(32bit 浮動小数点型)複数読込	P.30
WriteSingleDiscreteOutput (ForceSingleCoil)	×	○	5(0x05)	-	DO(Discrete Output)単一書出	P.31
WriteSingleHoldingRegister (PresetSingleRegister)	×	○	6(0x06)	-	保持レジスタ(16bit)単一書出	P.31
ReadExceptionStatus (同じ)	×	×	7(0x07)	-	例外ステータス状態読込	P.31
DiagnosticsReturnQueryData (同じ)	×	×	8(0x08)	0	診断：クエリデータのエコーバック	P.32
DiagnosticsRestartCommunicationsOption (同じ)				1	診断：通信ポートの初期化	P.32
WriteMultipleDiscreteOutputs (ForceMultipleCoils)	×	○	15(0x0F)	-	DO(Discrete Output)複数書出	P.32
WriteMultipleHoldingRegisters (PresetMultipleRegisters)	×	○	16(0x10)	-	保持レジスタ(16bit)複数書出	P.33
WriteMultipleHoldingRegistersLongInt （なし）					保持レジスタ(32bit 整数型)複数書出	P.33
WriteMultipleHoldingRegistersFloat （なし）					保持レジスタ(32bit 浮動小数点型)複数書出	P.34
MaskWriteHoldingRegister (MaskWrite4XRegister)	×	×	22(0x16)	-	保持レジスタ AND/OR マスク書出	P.34
ReadWriteMultipleHoldingRegisters (ReadWrite4XRegisters)	×	×	23(0x17)	-	保持レジスタ複数読込書出	P.34
AnotherFunctionCode （なし）	×	Function Code に依存	1-127 (0x01-0x7F)	-	その他ファンクションコード	P.35

2.4.4. CaoExtension::Execute Modbus 機能対応コマンド詳細（クライアントモードのみ）

ReadMultipleDiscreteOutputs

Function Code	Broad cast
1(0x01)	×

構文 *object*.ReadMultipleDiscreteOutputs(<Address>, <Count>)

引数 <Address> = VT_I4: 開始アドレス
範囲: 1-65536 "OffsetAddressZero=False"時(デフォルト)
0-65535 "OffsetAddressZero=True" 時
<Count> = VT_I4: 読み込み点数　範囲: 1-2000

戻り値 <Data> = VT_ARRAY | VT_BOOL: 読み込みデータ
VARIANT_TRUE: DO（Discrete Output） ON 状態
VARIANT_FALSE: DO（Discrete Output） OFF 状態

説明 開始アドレスから連続した **DO（Discrete Output）**の ON / OFF 状態を読込みます.

ReadMultipleDiscreteInputs

Function Code	Broad cast
2(0x02)	×

構文 *object*.ReadMultipleDiscreteInputs(<Address>, <Count>)

引数 <Address> = VT_I4: 開始アドレス
範囲: 1-65536 "OffsetAddressZero=False" 時(デフォルト)
0-65535 "OffsetAddressZero=True" 時
<Count> = VT_I4: 読み込み点数 範囲: 1-2000

戻り値 <Data> = VT_ARRAY | VT_BOOL: 読み込みデータ
VARIANT_TRUE: DI（Discrete Input） ON 状態
VARIANT_FALSE: DI（Discrete Input） OFF 状態

説明 開始アドレスから連続した **DI（Discrete Input）**の ON / OFF 状態を読み込みます.

ReadMultipleHoldingRegisters

Function Code	Broad cast
3(0x03)	×

構文 *object*.ReadMultipleHoldingRegisters(<Address>, <Count>)

引数 <Address> = VT_I4: 開始アドレス
範囲: 1-65536 ”OffsetAddressZero=False” 時(デフォルト)
0-65535 ”OffsetAddressZero=True” 時
<Count> = VT_I4: 読み込みデータ数 範囲: 1-125

戻り値 <Data> = VT_ARRAY | VT_UI2: 読み込みデータ

説明 開始アドレスから連続した**保持レジスタ(16bit)**の内容を読み込みます.

ReadMultipleHoldingRegistersLongInt

Function Code	Broad cast
3(0x03)	×

構文 *object*.ReadMultipleHoldingRegistersLongInt(<Address>, <Count>)

引数 <Address> = VT_I4: 開始アドレス
範囲: 1-65536 ”OffsetAddressZero=False” 時(デフォルト)
0-65535 ”OffsetAddressZero=True” 時
<Count> = VT_I4: 読み込みデータ数 範囲: 1-62

戻り値 <Data> = VT_ARRAY | VT_I4: 読み込みデータ

説明 開始アドレスから連続した**保持レジスタ(32bit 整数型)**の内容を読み込みます.

ReadMultipleHoldingRegistersFloat

Function Code	Broad cast
3(0x03)	×

構文 *object*.ReadMultipleHoldingRegistersFloat(<Address>, <Count>)

引数 <Address> = VT_I4: 開始アドレス
範囲: 1-65536 ”OffsetAddressZero=False” 時(デフォルト)
0-65535 ”OffsetAddressZero=True” 時
<Count> = VT_I4: 読み込みデータ数 範囲: 1-62

戻り値 <Data> = VT_ARRAY | VT_R4: 読み込みデータ

説明 開始アドレスから連続した**保持レジスタ(32bit 浮動小数点型)**の内容を読み込みます.

ReadMultipleInputRegisters

Function Code	Broad cast
4(0x04)	×

構文 *object*.ReadMultipleInputRegisters(<Address>, <Count>)

引数 <Address> = VT_I4: 開始アドレス
範囲: 1-65536 "OffsetAddressZero=False" 時(デフォルト)
0-65535 "OffsetAddressZero=True" 時
<Count> = VT_I4: 読み込みデータ数（範囲: 1-125）

戻り値 <Data> = VT_ARRAY | VT_UI2: 読み込みデータ

説明 開始アドレスから連続した**入力レジスタ(16bit)**の内容を読み込みます.

ReadMultipleInputRegistersLongInt

Function Code	Broad cast
4(0x04)	×

構文 *object*.ReadMultipleInputRegistersLongInt(<Address>, <Count>)

引数 <Address> = VT_I4: 開始アドレス
範囲: 1-65536 "OffsetAddressZero=False" 時(デフォルト)
0-65535 "OffsetAddressZero=True" 時
<Count> = VT_I4: 読み込みデータ数（範囲: 1-62）

戻り値 <Data> = VT_ARRAY | VT_I4: 読み込みデータ

説明 開始アドレスから連続した**入力レジスタ(32bit 整数型)**の内容を読み込みます.

ReadMultipleInputRegistersFloat

Function Code	Broad cast
4(0x04)	×

構文 *object*.ReadMultipleInputRegistersFloat(<Address>, <Count>)

引数 <Address> = VT_I4: 開始アドレス
範囲: 1-65536 "OffsetAddressZero=False" 時(デフォルト)
0-65535 "OffsetAddressZero=True" 時
<Count> = VT_I4: 読み込みデータ数（範囲: 1-62）

戻り値 <Data> = VT_ARRAY | VT_R4: 読み込みデータ

説明 開始アドレスから連続した**入力レジスタ(32bit 浮動小数点型)**の内容を読み込みます.

WriteSingleDiscreteOutput

Function Code	Broadcast	
	eth	com
5(0x05)	×	○

構文 *object*.WriteSingleDiscreteOutput(<Address>, <Data>)

引数 <Address> = VT_I4: アドレス
範囲: 1-65536 "OffsetAddressZero=False" 時(デフォルト)
0-65535 "OffsetAddressZero=True" 時
<Data> = VT_BOOL: 書き出しデータ
ON: VARIANT_TRUE, OFF: VARIANT_FALSE

戻り値 なし

説明 指定アドレスの **DO(Discrete Output)** の内容を更新します.

WriteSingleHoldingRegister

Function Code	Broadcast	
	eth	com
6(0x06)	×	○

構文 *object*.WriteSingleHoldingRegister(<Address>, <Data>)

引数 <Address> = VT_I4: アドレス
範囲: 1-65536 "OffsetAddressZero=False" 時(デフォルト)
0-65535 "OffsetAddressZero=True" 時
<Data> = VT_UI2: 書き出しデータ

戻り値 なし

説明 指定アドレスの**保持レジスタ(16bit)** の内容を更新します.

ReadExceptionStatus

Function Code	Broad cast
7(0x07)	×

構文 *object*.ReadExceptionStatus( )

引数 なし

戻り値 <Data> = VT_UI1: 例外ステータス

説明 例外ステータスの状態を読み込みます.

読み込んだ例外ステータスは，下位ビットから1ビットずつ割り当てられています．

DiagnosticsReturnQueryData

Function Code-Sub	Broad cast
8(0x08)-0	×

構文 *object*.DiagnosticsReturnQueryData(<Count>, <Query Data>)

引数 <Count> = VT_I4: クエリデータ数（範囲: 1-250）
<Query Data > = ARRAY|VT_UI1: クエリデータ

戻り値 < Echo Data > = ARRAY|VT_UI1: エコーデータ

説明 クライアントから送信したクエリデータを，サーバはそのままエコーバックして返信することにより通信回線の診断を行います．
成功した場合，戻り値の<Echo Data>は引数で指定した<Query Data>と同じ値になります．

DiagnosticsRestartCommunicationsOption

Function Code-Sub	Broad cast
8(0x08)-1	×

構文 *object*.DiagnosticsRestartCommunicationsOption(<ClearEventLog>)

引数 <ClearEventLog> = VT_BOOL: イベントログクリア指定
イベントログクリアする: VARIANT_TRUE
イベントログ残す: VARIANT_FALSE

戻り値 なし

説明 サーバ側の通信ポートを初期化する．
また，イベントログをクリアするか残すかを指定する．

WriteMultipleDiscreteOutputs

Function Code	Broadcast	
	eth	com
15(0x0F)	×	○

構文 *object*.WriteMultipleDiscreteOutputs(<Address>, <Count>, <Data>)

引数 <Address> = VT_I4: 開始アドレス
範囲: 1-65536 ”OffsetAddressZero=False” 時(デフォルト)
0-65535 ”OffsetAddressZero=True” 時

<Count> = VT_I4: 書き出し点数（範囲: 1-1968）
<Data> = VT_ARRAY | VT_BOOL: ON/OFF データ
ON: VARIANT_TRUE, OFF: VARIANT_FALSE

戻り値 なし

説明 開始アドレスから連続した点数分の **DO(Discrete Output)**の内容を更新します.

WriteMultipleHoldingRegisters

Function Code	Broadcast	
	eth	com
16(0x10)	×	○

構文 *object*. WriteMultipleHoldingRegisters(<Address>, <Data>)

引数 <Address> = VT_I4: 開始アドレス
範囲: 1-65536 "OffsetAddressZero=False" 時(デフォルト)
0-65535 "OffsetAddressZero=True" 時
<Count> = VT_I4: 書き出しデータ数（範囲: 1-123）
<Data> = VT_ARRAY | VT_UI2: 書き出しデータ

戻り値 なし

説明 開始アドレスから連続した**保持レジスタ(16bit)**の内容を更新します.

WriteMultipleHoldingRegistersLongInt

Function Code	Broadcast	
	eth	com
16(0x10)	×	○

構文 *object*.WriteMultipleHoldingRegistersLongInt(<Address>, <Data>)

引数 <Address> = VT_I4: 開始アドレス
範囲: 1-65536 "OffsetAddressZero=False" 時(デフォルト)
0-65535 "OffsetAddressZero=True" 時
<Count> = VT_I4: 書き出しデータ数（範囲: 1-61）
<Data> = VT_ARRAY | VT_I4: 書き出しデータ

戻り値 なし

説明 開始アドレスから連続した**保持レジスタ(32bit 整数型)**の内容を更新します.

WriteMultipleHoldingRegistersFloat

Function Code	Broadcast	
	eth	com
16(0x10)	×	○

構文 *object*. WriteMultipleHoldingRegistersFloat(<Address>, <Count>, <Data>)

引数 <Address> = VT_I4: 開始アドレス
範囲: 1-65536 ”OffsetAddressZero=False” 時(デフォルト)
0-65535 ”OffsetAddressZero=True” 時
<Count> = VT_I4: 書き出しデータ数（範囲: 1-61）
<Data> = VT_ARRAY | VT_R4: 書き出しデータ

戻り値 なし

説明 開始アドレスから連続した**保持レジスタ(32bit 浮動小数点型)**の内容を更新します.

MaskWriteHoldingRegister

Function Code	Broad cast
22(0x16)	×

構文 *object*.MaskWriteHoldingRegister(<Address>, <AND_Mask>, <OR_Mask>)

引数 <Address> = VT_I4: アドレス
範囲: 1-65536 ”OffsetAddressZero=False” 時(デフォルト)
0-65535 ”OffsetAddressZero=True” 時
<AND_Mask> = VT_UI2: AND マスク値
<OR_Mask> = VT_UI2: OR マスク値

戻り値 なし

説明 指定アドレスの**保持レジスタ(16bit)**の現在値とAND マスク値, OR マスク値を組み合わせてた内容で更新します.
(参考) 更新する値は, 通常以下の式でサーバ機器側で処理(演算)されます.
[更新値] = ([現在値]AND<AND_Mask>) OR (<OR_Mask> AND <NOT AND_Mask>)

ReadWriteMultipleHoldingRegisters

Function Code	Broad cast
23(0x17)	×

構文 *object*.ReadWriteMultipleHoldingRegisters(<ReadAddress>, <ReadCount>,

<WriteAddress>, <WriteCount>, <WriteData>)

引数 <ReadAddress> = VT_I4: 読み込み開始アドレス
範囲: 1-65536 "OffsetAddressZero=False" 時(デフォルト)
0-65535 "OffsetAddressZero=True" 時
<ReadCount> = VT_I4: 読み込み数（範囲: 1-125）
<WriteAddress> = VT_I4: 書き出し開始アドレス
範囲: 1-65536 "OffsetAddressZero=False" 時(デフォルト)
0-65535 "OffsetAddressZero=True" 時
<WriteCount> = VT_I4: 書き出し数（範囲: 1-121）
<WriteData> = VT_ARRAY | VT_UI2: 書き出しデータ

戻り値 <Data> = VT_ARRAY | VT_UI2: 読み込みデータ

説明 保持レジスタ(16bit)の読み込みと書き出しを実行します.
<WriteAddress>で指定したアドレスに<WriteCount>数分<WriteData>を書き込み，<ReadAddress>で指定したアドレスから<ReadCount>の数分のデータを読み込みます.

AnotherFunctionCode

Function Code	Broadcast	
	eth	com
1-127 (0x01-0x7F)	×	FunctionCode に依存

構文 *object*.AnotherFunctionCode(<FunctionCode>, <RequestCount>, <RequestData>)

引数 <FunctionCode> = VT_I4: ファンクションコード（範囲: 1-127）
<RequestCount> = VT_I4: データ部送信データ数（範囲: 0-252）
<RequestData> = VT_ARRAY | VT_UI1: データ部送信データ
<ResponseCount> = VT_I4: データ部応答データ数（範囲: 0-252）

戻り値 <ResponseData> = VT_ARRAY | VT_UI1: データ部応答データ

説明 その他(CaoController::Execute コマンドとして定義されていない)の Modbus ファンクションコードを実行します.
<RequestData>に<FunctionCode>に対応した，Modbus クエリメッセージ内データ部に相当する情報のみバイナリで<RequestCount>数分指定します(電文チェックコードは自動付加されます). <ResponseData>に Modbus 応答メッセージ内のデータ部に相当する情報のみバイナリで<ResponseCount>数分読み込まれます(電文チェックコードは自動削除されます).

[注意事項]

1)com 時の機器アドレスが Broadcast アドレス(UnitAddress=0)の場合，サーバ機器からの応答ﾃﾞｰﾀはないため，<ResponseCount>は無視され<ResponseData>は VT_EMPTY となります.

2)<ResponseCount>に 0 を設定した場合，<ResponseData>は VT_EMPTY となります.

3)上記 1),2)の場合を除き，<ResponseCount> <> 実際の<ResponseData>ﾃﾞｰﾀ数 の場合通信デバイスにより，以下の実行結果となります.

com の時: エラー終了.

eth の時: 正常終了はするが,
戻り値が <ResponseCount> <> 実際の<ResponseData>ﾃﾞｰﾀ数

2.5. 変数一覧

2.5.1. CaoController クラス

表 2-14 CaoController 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	バージョン情報	○	―
@Error	VT_I4	最後に発生したエラーコードを取得します.	○	―
@IpInfo	VT_ARRAY| VT_VARIANT 又は VT_EMPTY	現在接続されているクライアント機器の, IPアドレスとポート番号を取得します. Array[0]: VT_I4 接続クライアント数(0～16) Array[1]: 1 台目接続時: VT_ARRAY|VT_VARIANT 1 台目未接続時: VT_EMPTY Array[1][0]: VT_BSTR 1 台目 接続 IP アドレス Array[1][1]: VT_I4 1 台目 接続ポート番号 ・ ・ ・ Array[17]: 16 台目接続時: VT_ARRAY|VT_VARIANT 16 台目未接続時: VT_EMPTY Array[17][0]: VT_BSTR 16 台目 接続 IP アドレス Array[17][1]: VT_I4 16 台目 接続ポート番号 注: クライアントモード時 または シリアルモード時は VT_EMPTY となります.	○	―

2.5.2. CaoExtension クラス（クライアントモードのみ）

下表に，クライアントモード時の Modbus 機能対応ユーザ変数を記します．

表 2-15 CaoExtension クラス Modbus 機能対応ユーザ変数一覧

<table>
<tr><th rowspan="2">変数名</th><th rowspan="2">データ型</th><th rowspan="2">説明</th><th colspan="2">属性</th><th rowspan="2">備考</th></tr>
<tr><th>get</th><th>put</th></tr>
<tr><td>DO?※1</td><td>VT_I4※2</td><td>?番目の DO(DiscreteOutput)の状態(値)を設定/取得します．
変数名末尾の?は，アドレスとして論理番号を指定します．
例）“DO1”
- オフセットアドレス(”OffsetAddressZero=False”) 1～の場合 - サーバ機器側の DO アドレス“0”に相当
- オフセットアドレス(”OffsetAddressZero=True”) 0～の場合 - サーバ機器側の DO アドレス“1”に相当

・論理番号の範囲は，AddController の接続パラメータ オフセットアドレス(“OffsetAddressZero”)オプション と AddVariable 時のデータ幅(“UserVarWidth”)オプション との組み合わせにより下表となります．

<table>
<tr><th rowspan="2">オフセットアドレス
（OffsetAddressZero）</th><th colspan="4">データ幅（UserVarWidth）</th></tr>
<tr><th>1</th><th>8</th><th>16</th><th>32</th></tr>
<tr><td>True</td><td>0 ～ 65535</td><td>0 ～ 65528</td><td>0 ～ 65520</td><td>0 ～ 65504</td></tr>
<tr><td>False
(デフォルト)</td><td>1 ～ 65536</td><td>1 ～ 65529</td><td>1 ～ 65521</td><td>1 ～ 65505</td></tr>
</table>
・設定値／取得値の範囲は，データ幅(“UserVarWidth” オプション)により下表となります．また，データ幅が”1”以外時のデータ表記は，MSB(Most Significant Bit)となります．

<table>
<tr><th rowspan="2"></th><th colspan="4">データ幅（UserVarWidth）</th></tr>
<tr><th>1</th><th>8</th><th>16</th><th>32</th></tr>
<tr><td>設定値/取得値の範囲</td><td>0 ～ 1</td><td>0 ～ 255</td><td>0 ～ 65535</td><td>-2147483648 ～ +2147483647</td></tr>
</table>
（参考）
Modbus 通信プロトコル上送受信される FunctionCode は，設定/取得 と データ幅(“UserVarWidth”)オプションの組み合わせにより，下表となります．

<table>
<tr><th rowspan="2"></th><th colspan="4">データ幅（UserVarWidth）</th></tr>
<tr><th>1</th><th>8</th><th>16</th><th>32</th></tr>
<tr><td>取得（get）</td><td>1(0x01)</td><td>1(0x01)</td><td>1(0x01)</td><td>1(0x01)</td></tr>
<tr><td>設定（put）</td><td>5(0x05)</td><td>15(0x0F)</td><td>15(0x0F)</td><td>15(0x0F)</td></tr>
</table>
</td><td>○</td><td>○</td><td></td></tr>
</table>

<table>
<tr>
<td>DI?※1</td>
<td>VT_I4※2</td>
<td>
?番目の DI(DiscreteInput)の状態(値)を設定/取得します.
変数名末尾の?は, アドレスとして論理番号を指定します.
例）“DI1”
- オフセットアドレス(”OffsetAddressZero=False”) 1～の場合 -
サーバ機器側の DI アドレス“0”に相当
- オフセットアドレス(”OffsetAddressZero=True”) 0～の場合 -
サーバ機器側の DI アドレス“1”に相当

・論理番号の範囲は, AddController の接続パラメータ
オフセットアドレス(“OffsetAddressZero”)オプション と
AddVariable 時のデータ幅(“UserVarWidth”)オプション
との組み合わせにより下表となります.

<table>
<tr><td>オフセットアドレス
（OffsetAddressZero）</td><td colspan="4">データ幅（UserVarWidth）</td></tr>
<tr><td></td><td>1</td><td>8</td><td>16</td><td>32</td></tr>
<tr><td>True</td><td>0 ～
65535</td><td>0 ～
65528</td><td>0 ～
65520</td><td>0 ～
65504</td></tr>
<tr><td>False
(デフォルト)</td><td>1 ～
65536</td><td>1 ～
65529</td><td>1 ～
65521</td><td>1 ～
65505</td></tr>
</table>
・設定値／取得値の範囲は, データ幅(“UserVarWidth”
オプション)により下表となります. また, データ幅が”1”以外時の
データ表記は, MSB(Most Significant Bit)となります.

<table>
<tr><td></td><td colspan="4">データ幅（UserVarWidth）</td></tr>
<tr><td></td><td>1</td><td>8</td><td>16</td><td>32</td></tr>
<tr><td>設定値/取得
値の範囲</td><td>0 ～
1</td><td>0 ～
255</td><td>0 ～
65535</td><td>-2147483648 ～
+2147483647</td></tr>
</table>
（参考）
Modbus 通信プロトコル上送受信される FunctionCode は,
設定/取得 と データ幅(“UserVarWidth”)オプションの
組み合わせにより, 下表となります.

<table>
<tr><td></td><td colspan="4">データ幅（UserVarWidth）</td></tr>
<tr><td></td><td>1</td><td>8</td><td>16</td><td>32</td></tr>
<tr><td>取得（get）</td><td>2(0x02)</td><td>2(0x02)</td><td>2(0x02)</td><td>2(0x02)</td></tr>
<tr><td>設定（put）</td><td>－</td><td>－</td><td>－</td><td>－</td></tr>
</table>
</td>
<td>○</td>
<td>-</td>
<td></td>
</tr>
</table>

<table>
<tr>
<td>HRI?※1</td>
<td>VT_I4※2</td>
<td>?番目の保持レジスタの値を整数型で設定/取得します.
変数名末尾の?は, レジスタアドレスとして論理番号を指定します.
例）“HRI1”
- オフセットアドレス(”OffsetAddressZero=False”) 1～の場合 -
サーバ機器側の保持レジスタアドレス“0”に相当
- オフセットアドレス(”OffsetAddressZero=True”) 0～の場合 -
サーバ機器側の保持レジスタアドレス“1”に相当

・論理番号の範囲は, AddController の接続パラメータ
オフセットアドレス(“OffsetAddressZero”)オプション と
AddVariable 時のデータ幅(“UserVarWidth”)オプション
との組み合わせにより下表となります.

<table>
<tr><th rowspan="2">オフセットアドレス
（OffsetAddressZero）</th><th colspan="2">データ幅（UserVarWidth）</th></tr>
<tr><th>16</th><th>32</th></tr>
<tr><td>True</td><td>0 ～ 65535</td><td>0 ～65534</td></tr>
<tr><td>False (デフォルト)</td><td>1 ～ 65536</td><td>1 ～65535</td></tr>
</table>

・設定値／取得値の範囲は, データ幅(“UserVarWidth”)
オプションにより下表となります.

<table>
<tr><th rowspan="2"></th><th colspan="2">データ幅（UserVarWidth）</th></tr>
<tr><th>16</th><th>32</th></tr>
<tr><td>設定値/取得値の範囲</td><td>0 ～ 65535</td><td>-2147483648 ～ +2147483647</td></tr>
</table>

（参考）
Modbus 通信プロトコル上送受信される FunctionCode は,
設定/取得 と データ幅(“UserVarWidth”)オプションの
組み合わせにより, 下表となります.

<table>
<tr><th rowspan="2"></th><th colspan="2">データ幅（UserVarWidth）</th></tr>
<tr><th>16</th><th>32</th></tr>
<tr><td>取得（get）</td><td>3(0x03)</td><td>3(0x03)</td></tr>
<tr><td>設定（put）</td><td>6(0x06)</td><td>16(0x10)</td></tr>
</table>
</td>
<td>○</td>
<td>○</td>
<td></td>
</tr>
</table>

<table>
<tr>
<td>HRF?</td>
<td>VT_R4</td>
<td>?番目の保持レジスタの値を32bit 浮動小数点型で設定/取得します.
変数名末尾の?は, レジスタアドレスとして論理番号を指定します.
例）“HRF1”
- オフセットアドレス(”OffsetAddressZero=False”) 1～の場合 -
サーバ機器側の保持レジスタアドレス“0”に相当
- オフセットアドレス(”OffsetAddressZero=True”) 0～の場合 -
サーバ機器側の保持レジスタアドレス“1”に相当

・論理番号の範囲は, AddController の接続パラメータ
オフセットアドレス(“OffsetAddressZero”)オプション と
AddVariable 時のデータ幅(“UserVarWidth”)オプション
との組み合わせにより下表となります.

<table>
<tr><th rowspan="2">オフセットアドレス
（OffsetAddressZero）</th><th>データ幅（UserVarWidth）</th></tr>
<tr><th>32 固定</th></tr>
<tr><td>True</td><td>0 ～ 65534</td></tr>
<tr><td>False (デフォルト)</td><td>1 ～ 65535</td></tr>
</table>

（参考）
Modbus 通信プロトコル上送受信される FunctionCode は,
設定/取得 と データ幅(“UserVarWidth”)オプションの
組み合わせにより, 下表となります.

<table>
<tr><th rowspan="2"></th><th>データ幅（UserVarWidth）</th></tr>
<tr><th>32 固定</th></tr>
<tr><td>取得（get）</td><td>3(0x03)</td></tr>
<tr><td>設定（put）</td><td>16(0x10)</td></tr>
</table>
</td>
<td>○</td>
<td>○</td>
<td></td>
</tr>
</table>

<table>
<tr>
<td>IRI?※1</td>
<td>VT_I4※2</td>
<td>?番目の入力レジスタの値を整数型で取得します.
変数名末尾の?は, レジスタアドレスとして論理番号を指定します.
例）“IRI1”
- オフセットアドレス(”OffsetAddressZero=False”) 1～の場合 -
サーバ機器側の入力レジスタアドレス“0”に相当
- オフセットアドレス(”OffsetAddressZero=True”) 0～の場合 -
サーバ機器側の入力レジスタアドレス“1”に相当

・論理番号の範囲は, AddController の接続パラメータ
オフセットアドレス(“OffsetAddressZero”)オプション と
AddVariable 時のデータ幅(“UserVarWidth”)オプション
との組み合わせにより下表となります.

<table>
<tr><th rowspan="2">オフセットアドレス
（OffsetAddressZero）</th><th colspan="2">データ幅（UserVarWidth）</th></tr>
<tr><th>16</th><th>32</th></tr>
<tr><td>True</td><td>0 ～ 65535</td><td>0 ～ 65534</td></tr>
<tr><td>False (デフォルト)</td><td>1 ～ 65536</td><td>1 ～ 65535</td></tr>
</table>

・設定値／取得値の範囲は, データ幅(“UserVarWidth”)
オプションにより下表となります.

<table>
<tr><th rowspan="2"></th><th colspan="2">データ幅（UserVarWidth）</th></tr>
<tr><th>16</th><th>32</th></tr>
<tr><td>設定値/取得
値の範囲</td><td>0 ～ 65535</td><td>-2147483648 ～ +2147483647</td></tr>
</table>

（参考）
Modbus 通信プロトコル上送受信される FunctionCode は,
設定/取得 と データ幅(“UserVarWidth”)オプションの
組み合わせにより, 下表となります.

<table>
<tr><th rowspan="2"></th><th colspan="2">データ幅（UserVarWidth）</th></tr>
<tr><th>16</th><th>32</th></tr>
<tr><td>取得（get）</td><td>4(0x04)</td><td>4(0x04)</td></tr>
<tr><td>設定（put）</td><td>－</td><td>－</td></tr>
</table>
</td>
<td>○</td>
<td>-</td>
<td></td>
</tr>
</table>

<table>
<tr>
<td>IRF?</td>
<td>VT_R4</td>
<td>?番目の入力レジスタの値を 32bit 浮動小数点型で取得します.
変数名末尾の?は, レジスタアドレスとして論理番号を指定します.
例）“ IRF1”
- オフセットアドレス(”OffsetAddressZero=False”) 1～の場合 -
サーバ機器側の入力レジスタアドレス“0”に相当
- オフセットアドレス(”OffsetAddressZero=True”) 0～の場合 -
サーバ機器側の入力レジスタアドレス“1”に相当

・論理番号の範囲は, AddController の接続パラメータ
オフセットアドレス(“OffsetAddressZero”)オプション と
AddVariable 時のデータ幅(“UserVarWidth”)オプション
との組み合わせにより下表となります.

<table>
<tr><th>オフセットアドレス
（OffsetAddressZero）</th><th>データ幅（UserVarWidth）
32 固定</th></tr>
<tr><td>True</td><td>0 ～ 65534</td></tr>
<tr><td>False (デフォルト)</td><td>1 ～ 65535</td></tr>
</table>

（参考）
Modbus 通信プロトコル上送受信される FunctionCode は,
設定/取得 と データ幅(“UserVarWidth”)オプションの
組み合わせにより, 下表となります.

<table>
<tr><th></th><th>データ幅（UserVarWidth）
32 固定</th></tr>
<tr><td>取得（get）</td><td>4(0x04)</td></tr>
<tr><td>設定（put）</td><td>－</td></tr>
</table>
</td>
<td>○</td>
<td>-</td>
<td></td>
</tr>
</table>

※1: “VT”オプション有効時, 論理番号の範囲は変数型により異なります. 詳細は表 2-6 参照.
※2: “VT” 又は “Elem”オプション有効時, 変数型は異なります. 詳細は表 2-5, 表 2-6 参照.

2.6. エラーコード

Modbus.X プロバイダでは，以下の固有エラーコードが定義されています．

表 2-16 固有エラーコード一覧

エラー名	エラー番号	説明
E_CAOP_SYNC_ONLY	0x80100001	同期モードでのみ，実行することができます．
E_CAOP_ASYNC_ONLY	0x80100002	非同期モードでのみ，実行することができます．
E_CAOP_CLIENT_ONLY	0x80100003	クライアントモードでのみ，実行することができます．
E_CAOP_SERVER_ONLY	0x80100004	サーバモードでのみ，実行することができます．
E_CAOP_SERVER_SEND_REPLY_TIMEOUT	0x80100005	サーバモード時，要求（クエリ）メッセージを受信した後，”SendReplyTimeout”オプションで指定した時間経過以降に応答(リプライ)送信した場合に発生します．
E_CAOP_ILLEGAL_ARGUMENT	0x80100F01	引数エラー． このエラーコードを返すコマンドに渡されたパラメータが無効か，範囲外です．
E_CAOP_ILLEGAL_STATE	0x80100F02	状態エラー． 関数が間違った状態で呼ばれています．プロトコルがまだ正常にオープンされていない場合，この戻りコードはすべての関数によって返されます．
E_CAOP_ILLEGAL_SLAVE_ADDRESS	0x80100F05	不正サーバ機器アドレス． ブロードキャストに対応していない機能でアドレス 0 が使用されました．
E_CAOP_OPEN	0x80100F42	ポートまたはソケットオープンエラー． TCP/IP ソケットまたはシリアルポートを開けませんで

		した. シリアルポートの場合には, シリアルポートがシステムに存在していない可能性があります.
E_CAOP_FTALK_PORT_ALREADY_OPEN	0x80100F43	シリアル･ポートは既に開いています. オープン操作用に定義されたシリアルポートが既に他のアプリケーションで開かれます.
E_CAOP_FTALK_TCPIP_CONNECT	0x80100F44	TCP/IP 接続エラー. TCP/IP 接続を確立できませんでした. ホストがネットワーク または IP アドレス上に存在するか, 名前が間違っているホストの場合, 通常このエラーが発生します. リモートホストは, 適切な Port 番号を Listen する必要があります.
E_CAOP_CONNECTION_WAS_CLOSED	0x80100F45	リモートピアは, TCP/IP 接続を閉じました. TCP / IP 接続がリモートピアによって閉じたり壊れていたことを通知します.
E_CAOP_SOCKET_LIB	0x80100F46	ソケットライブラリエラー. TCP/IP ソケットライブラリー（例えば WINSOCK）がロードできませんでした. DLL が見つからないか, インストールされていない可能性があります.
E_CAOP_PORT_ALREADY_BOUND	0x80100F47	TCP ポートは既にバインドしています. 指定された TCP ポートをバインドすることができないことを示します. Port が既に別のアプリケーションによって取らされたり, 再使用のための TCP / IP スタックによってまだリリースされていない可能性があります.
E_CAOP_LISTEN_FAILED	0x80100F48	Lisen に失敗しました. 指定された TCP ポート Listen に失敗しました.
E_CAOP_FILEDES_EXCEEDED	0x80100F49	ファイル記述子を超えました. 使用可能なファイル記述

		子の最大数を超えました.
E_CAOP_PORT_NO_ACCESS	0x80100F4A	シリアルポートまたは TCP ポートにアクセスする権限がありません. シリアルポートの場合, アクセス権を変更します. TCP/ IP の場合, TCP ポート番号が IPPORT_RESERVED 範囲外です.
E_CAOP_PORT_NOT_AVAIL	0x80100F4B	TCP ポートは使用できません. 指定された TCP ポートは, この動作環境では利用できません.
E_CAOP_LINE_BUSY	0x80100F4C	シリアルラインがビジーです. シリアル回線は, トラフィックがあってはならない状態であるにもかかわらず, ノイズ等を受信しています.
E_CAOP_CHECKSUM	0x80100F81	チェックサムエラー. 受信したフレームのチェックサムが無効です.
E_CAOP_INVALID_FRAME	0x80100F82	無効なフレームエラー. 受信したフレームが通信プロトコルのいずれかの構造または内容によって対応していないか, 以前に送信されたクエリのフレームと一致しないことを通知します.
E_CAOP_INVALID_REPLY	0x80100F83	無効な応答エラー. 受信した応答フレームが通信プロトコルに対応していないことを通知します.
E_CAOP_REPLY_TIMEOUT	0x80100F84	タイムアウトエラー. サーバ機器が時間内に応答しない, またはまったく応答しない場合に発生することがあります. 間違ったサーバ機器アドレスは, このエラーを誘発します.
E_CAOP_SEND_TIMEOUT	0x80100F85	送信タイムアウトエラー. データ送信がタイムアウトしたことを通知します. ハンドシェーク･ラインが正し

		く設定されていない場合に発生する可能性があります.
E_CAOP_INVALID_MBAP_ID	0x80100F86	無効な識別子. プロトコルまたはトランザクション識別子ですが間違っています. TCPサーバデバイスは, TCP クライアントから受信した識別子を返す必要があります.
E_CAOP_MBUS_EXCEPTION_RESPONSE	0x80100FA0	Modbus 例外応答メッセージを受信したことを通知します
E_CAOP_MBUS_ILLEGAL_FUNCTION_RESPONSE	0x80100FA1	Modbus 無効な関数例外応答(コード 01)を受信したことを通知します.
E_CAOP_MBUS_MBUS_ILLEGAL_ADDRESS_RESPONSE	0x80100FA2	Modbus 不正データアドレス例外応答(コード 02)を受信したことを通知します.
E_CAOP_MBUS_ILLEGAL_VALUE_RESPONSE	0x80100FA3	Modbus 不正な値例外応答(コード 03)でを受信したことを通知します.
E_CAOP__MBUS_SLAVE_FAILURE_RESPONSE	0x80100FA4	Modbus スレーブ失敗例外応答(コード 04)でを受信したことを通知します.

・ Windows のシステムエラーが発生した場合は, エラー番号を“0x8010000”でマスクした値を返します.

例) Windows のシステムエラー:2(0x0002) → CAO API のエラー:0x80100002

ORiN2 共通エラーについては,「ORiN2 プログラミングガイド」のエラーコードの章を参照してください.

3. サンプルプログラム

3.1. クライアントモード

以下に, RS232C/RS485 デバイス:COM1 で接続後,

- サーバ機器アドレス=1 のデバイスに対して, ユーザ変数“DO1”を定義しDO(DiscreateOutput)のアドレス 1 に ON 又は OFF を出力(設定).
- サーバ機器アドレス=2 のデバイスに対して, ユーザ変数“DI1”を 8bit 幅で定義し, DI(DiscreateInput)のアドレス 1-8 の状態を, MSB(Most Significant Bit)でバイトデータとして取得.

するサンプルを示します.

List 3-1 Sample31.frm

```
Private caoEng As CaoEngine
Private caoCntl As CaoController
Private caoExt As CaoExtension
Private caoVarS1DO1 As CaoVariable
Private caoVarS2DI1 As CaoVariable

Private Sub Form_Load()

    Set caoEng = New CaoEngine
    Set caoCntl = caoEng.Workspaces(0).AddController("", "CaoProv.Modbus.X", "", "Conn=COM1")
    Set caoExt1 = caoCntl.AddExtension("MbSlave1")
    Set caoExt2 = caoCntl.AddExtension("MbSlave2", "UnitAddress=2" )

    Set caoVarS1DO1 = caoExt1.AddVariable("DO1", "")
    Set caoVarS2DI1 = caoExt2.AddVariable("DI1", " UserVarWidth=8")

End Sub

Private Sub cmdS1DO1 _ON_Click ()

    caoVarS1DO1.Value = True

End Sub

Private Sub cmdS1DO1_OFF_Click ()

    caoVarS1DO1.Value = False

End Sub

Private Sub cmdS2DI1_In_Click ()

    Ret = caoVarS2DI1.Value

    Text1.Text = Ret

End Sub
```

3.2. サーバモード

3.2.1. 同期モードサンプル

以下に, サーバモード, 同期モード, TCP 通信モード, IP アドレス"192.168.0.1"で接続後,

Timer イベント(Timer1)を使用して, 100ms 周期でクライアント機器からの要求(クエリ)メッセージを CaoController::Execute" ReceiveQuery"コマンドで受信し, "SendReply"コマンドで返信するサンプルを示します.

List 3-2-1 Sample321.frm

```
Private m_caoEng As CaoEngine
Private m_caoCntl As CaoController

' Modbus メモリマップ
Private Const MEM_ARRAY_SIZE As Long = (65536)
Private m_bDO(MEM_ARRAY_SIZE - 1) As Boolean   ' DO(Discrete Output)
Private m_bDI(MEM_ARRAY_SIZE - 1) As Boolean   ' DI(Discrete Input)
Private m_iHR(MEM_ARRAY_SIZE - 1) As Integer   ' 保持レジスタ(16bit)
Private m_iIR(MEM_ARRAY_SIZE - 1) As Integer   ' 保持レジスタ(16bit)
Private m_byExpSts As Byte

Private Sub Form_Load()

    Set m_caoEng = New CaoEngine
    Set m_caoCntl = m_caoEng.Workspaces(0).AddController("",  "CaoProv.Modbus.X",  "", _
                                          "Client=False, Sync=True, eth:192.168.0.1")
    Timer1.Interval = 100
    Timer1.Enabled = True

End Sub

Private Sub Sub Timer1_Timer()

    Dim vntArg As Variant
    Dim vntQueryData As Variant

    ' "ReceiveQuery"コマンド実行
    vntQueryData = m_caoCtrl.Execute("ReceiveQuery", vntArg)

    If IsEmpty(vntQueryData) Then
        Exit Sub
    End If

    ' Query データ解析&Reply データ生成
    vntArg = AnalyzeQueryDataToCreateReplyData(vntQueryData)

    If VarType(vntArg) <> vbEmpty Then
        ' "SendReply"コマンド実行
        M_caoCtrl.Execute "SendReply", vntArg
    End If

End Sub

Private Function AnalyzeQueryDataToCreateReplyData(vntQueryData As Variant) As Variant

    ' Query データ処理
    Dim i As Long
    Dim lSrvAddress As Long
    Dim lFuncCode As Long
    Dim lAddress As Long
    Dim lCount As Long
    Dim bArray() As Boolean
```

```
Dim iArray() As Integer
Dim bResult As Boolean
Dim vntData As Variant
Dim lArrSize As Long

lSrvAddress = CLng(vntQueryData(1))
lFuncCode = CLng(vntQueryData(2))

Select Case lFuncCode
' DO(Discrete Output)複数読込(1)
' DI(Discrete Input)複数読込(2)
Case 1, 2
    If (m_bTCP Or ((Not m_bTCP And lSrvAddress <> 0) And (lSrvAddress = m_lUnitAddr))) Then
        lAddress = CLng(vntQueryData(3)(0))
        lCount = CLng(vntQueryData(3)(1))
        If 0 < lCount Then
            If lAddress + lCount <= MEM_ARRAY_SIZE Then
                ReDim bArray(lCount - 1)
                If lFuncCode = 1 Then
                    For i = 0 To lCount - 1
                       bArray(i) = m_bDO(lAddress + i)
                    Next
                Else
                    For i = 0 To lCount - 1
                       bArray(i) = m_bDI(lAddress + i)
                    Next
                End If
                vntData = bArray
                bResult = True
            Else
                m_byExpSts = 3      ' 例外ステータス 3: アドレス範囲異常
                bResult = False
            End If
        Else
            m_byExpSts = 2      ' 例外ステータス 2: 点数異常
            bResult = False
        End If
    Else
        Exit Function
    End If

' 保持レジスタ(16bit)複数読込(3)
' 入力レジスタ(16bit)複数読込(4)
Case 3, 4
    If (m_bTCP Or ((Not m_bTCP And lSrvAddress <> 0) And (lSrvAddress = m_lUnitAddr))) Then
        lAddress = CLng(vntQueryData(3)(0))
        lCount = CLng(vntQueryData(3)(1))
        If 0 < lCount Then
            If lAddress + lCount <= MEM_ARRAY_SIZE Then
                ReDim iArray(lCount - 1)
                If lFuncCode = 3 Then
                    For i = 0 To lCount - 1
                       iArray(i) = m_iHR(lAddress + i)
                    Next
                Else
                    For i = 0 To lCount - 1
                       iArray(i) = m_iIR(lAddress + i)
                    Next
                End If
                vntData = iArray
                bResult = True
            Else
                m_byExpSts = 3      ' 例外ステータス 3: アドレス範囲異常
                bResult = False
            End If
        Else
```

```
                m_byExpSts = 2      ' 例外ステータス 2: 点数異常
                bResult = False
            End If
        Else
            Exit Function
        End If

    ' 例外ステータス状態読込(7)
    Case 7
        If (m_bTCP Or ((Not m_bTCP And lSrvAddress <> 0) And (lSrvAddress = m_lUnitAddr))) Then
            vntData = m_byExpSts
            bResult = True
        Else
            Exit Function
        End If

    ' DO(Discrete Output)複数書出(15)
    ' 保持レジスタ(16bit)複数書出(16)
    Case 15, 16
        lAddress = CLng(vntQueryData(3)(0))
        lCount = CLng(vntQueryData(3)(1))
        If 0 < lCount Then
            If lAddress + lCount <= MEM_ARRAY_SIZE Then
                Dim vt As Variant
                vt = VarType(vntQueryData(3)(2))
                If ((lFuncCode = 15) And vt = (vbArray Or vbBoolean)) Or _
                   ((lFuncCode = 16) And vt = (vbArray Or vbInteger)) Then
                    If lCount = UBound(vntQueryData(3)(2)) + 1 Then
                        If lFuncCode = 15 Then
                            For i = 0 To lCount - 1
                                m_bDO(lAddress + i) = vntQueryData(3)(2)(i)
                            Next
                        Else
                            For i = 0 To lCount - 1
                                m_iHR(lAddress + i) = CInt(vntQueryData(3)(2)(i))
                            Next
                        End If
                        bResult = True
                    Else
                        m_byExpSts = 5      ' 例外ステータス 5: 書出点数アンマッチ異常
                        bResult = False
                    End If
                Else
                    m_byExpSts = 4      ' 例外ステータス 4: 書出データ型アンマッチ異常
                    bResult = False
                End If
            Else
                m_byExpSts = 3      ' 例外ステータス 3: アドレス範囲異常
                bResult = False
            End If
        Else
            m_byExpSts = 2      ' 例外ステータス 2: 点数異常
            bResult = False
        End If
    Case Else
            m_byExpSts = 1      ' 例外ステータス 1: 通知ファンクション・コード異常
            bResult = False
    End Select

    AnalyzeQueryDataToCreateReplyData = Array(lFuncCode, bResult, vntData)

End Function
```

3.2.2. 非同期モードサンプル

以下に, サーバモード, 非同期モード, TCP 通信モード, IP アドレス"192.168.0.1"で接続後, クライアント機器からの要求(クエリ)メッセージを CaoController::OnMessage" QUERY_MSG メッセージ"で受信し, Message::Reply()メソッドで返信するサンプルを示します.

List 3-2-2 Sample322.frm

```
Private m_caoEng As CaoEngine
Private WithEvents m_caoCntl As CaoController

' Modbus メモリマップ
Private Const MEM_ARRAY_SIZE As Long = (65536)
Private m_bDO(MEM_ARRAY_SIZE - 1) As Boolean   ' DO(Discrete Output)
Private m_bDI(MEM_ARRAY_SIZE - 1) As Boolean   ' DI(Discrete Input)
Private m_iHR(MEM_ARRAY_SIZE - 1) As Integer   ' 保持レジスタ(16bit)
Private m_iIR(MEM_ARRAY_SIZE - 1) As Integer   ' 保持レジスタ(16bit)
Private m_byExpSts As Byte

Private Sub Form_Load()

    Set m_caoEng = New CaoEngine
    Set m_caoCntl = m_caoEng.Workspaces(0).AddController("",  "CaoProv.Modbus.X",  "", _
                                        "Client=False, Sync=False, eth:192.168.0.1")

End Sub

Private Sub m_caoCtrl_OnMessage(ByVal pICaoMess As CAOLib.ICaoMessage)

    Select Case pICaoMess.Number
    Case MSG_ID_QUERY_MSG

        ' Query データ解析&Reply データ生成
        Dim vntReply As Variant
        vntReply = AnalyzeQueryDataToCreateReplyData(pICaoMess.Value)
        PutLogQueryMsg pICaoMess

        If VarType(vntReply) <> vbEmpty Then
            ' Reply(返信)処理
            pICaoMess.Reply vntReply
            PutLogReplyData vntReply
        End If

    End Select

    Exit Sub

End Sub
```

4. 付録

4.1. 旧 Modbus コマンド名称との対比

下表に，旧 Modbus コマンド名称と新コマンド名との対比表を記します.

表 17 旧 Modbus コマンド名称との対比表

旧コマンド名称	新コマンド名称	Function Code (HEX)	頁
ReadCoilStatus	ReadMultipleDiscreteOutputs	1(0x01)	P.28
ReadInputStatus	ReadMultipleDiscreteInputs	2(0x02)	P.28
ReadHoldingRegister	ReadMultipleHoldingRegisters	3(0x03)	P.28
ReadInputRegister	ReadMultipleInputRegisters	4(0x04)	P.30
ForceSingleCoil	WriteSingleDiscreteOutput	5(0x05)	P.31
PresetSingleRegister	WriteSingleHoldingRegister	6(0x06)	P.31
ReadExceptionStatus	旧コマンド名と同じ	7(0x07)	P.31
DiagnosticsReturnQueryData	↑	8(0x08) - 0	P.32
DiagnosticsRestartCommunicationsOption	↑	8(0x08) - 1	P.32
ForceMultipleCoils	WriteMultipleDiscreteOutputs	15(0x0F)	P.32
PresetMultipleRegisters	WriteMultipleHoldingRegisters	16(0x10)	P.33
MaskWrite4XRegister	MaskWriteHoldingRegister	22(0x16)	P.34
ReadWrite4XRegisters)	ReadWriteMultipleHoldingRegisters	23(0x17)	P.34
FetchCommEventCounter	AnotherFunctionCode で代用	11(0x0B)	P.35
FetchCommEventLog	↑	12(0x0C)	P.35
ReportSlaveID	↑	17(0x11)	P.35
Read General Reference	↑	20(0x14)	P.35
Write General Reference	↑	21(0x15)	P.35
ReadFIFOQueue	↑	22(0x16)	P.35