JTEKT 製 PLC 用

CMP-LINK プロバイダ

Version 1.1.1

ユーザーズ ガイド

April 14, 2023

備考:

【改版履歴】

バージョン	日付	内容
1.0.0	2018-6-1	初版
1.0.1	2018-11-5	AddController 時のメモリリーク修正
	2020-5-7	5.アプリケーション開発のための環境セットアップ　追記 Ethernet 接続動作確認時の使用ユニットを追記
1.1.0	2021-8-27	PC10 で追加された拡張アドレスに対応 Ethnet 接続時の Bit 型デバイスの Bit アドレス指定時に PrgNo1 以外の指定を禁止 パケット分割時にデータが読み取れないバグを修正 アドレスの範囲チェックを追加. シリアル通信時に 1 桁の StationNumber を指定できないバグを修正
1.1.1	2021-12-21	エラー誤検知の修正
	2023-04-14	マニュアル修正

【対応機種】

機種	バージョン	注意事項
PC10G シリーズ		
PC3J シリーズ		PC10 拡張領域にはアクセスできません.
TOYOPUC-Nano		

【動作確認機種】

機種	バージョン	注意事項
PC10G-CPU (型式:TUC-6353)		
TOYOPUC-Nano (型式:TUC-6941)		

目次

1. はじめに

本書は，JTEKT 製 PLC に対しデータの書込み/読出しを行う CAO プロバイダのユーザーズガイドです．本書で扱う CAO プロバイダ(CaoProvJTEKTCMP-LINK.dll)を CMP-LINK プロバイダと呼びます．

第 2 章に CMP-LINK プロバイダの概要，変数やコマンドの詳細を記載しています．

CMP-LINK プロバイダで実装している通信コマンドの対応状況及びデータ列については，通信先となる PLC に依存します．通信の詳細については各種 PLC の取扱説明書の「コンピュータリンク機能」を参照ください．

図 1-1 構成図

また，本プロバイダ及びデバイスそれぞれの対応を図 1-2 に表します．

(※一例です．全てを表しているわけではありません．)

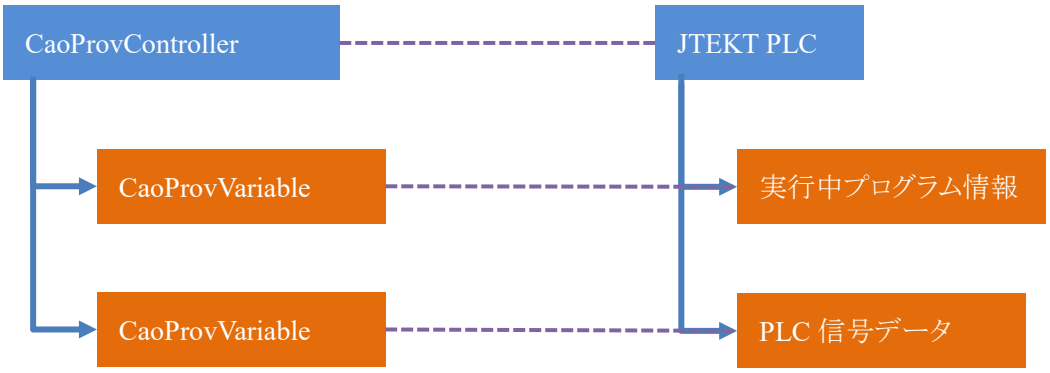


図 1-2 プロバイダの構成とデバイス情報との対応図

2. プロバイダの概要

2.1. 概要

CMP-LINK プロバイダは, JTEKT 製 PLC に対して TCP/IP またはシリアル接続し, コンピュータリンク機能を用いてデータの書込み/読出しを行う CAO プロバイダです.

そのファイル形式は DLL(Dynamic Link Library)であり, CAO エンジンから使用時に動的ロードされます. CMP-LINK プロバイダを使用するにあたっては ORiN2SDK をインストールするか, 下表を参照して手作業でレジストリ登録を行う必要があります.

表 2-1 CMP-LINK プロバイダ

ファイル名	CaoProvJTEKTCMP-LINK.dll
ProgID	CaoProv.JTEKT.CMP-LINK
レジストリ登録[1]	regsvr32 CaoProvJTEKTCMP-LINK.dll
レジストリ登録の抹消	regsvr32 /u CaoProvJTEKTCMP-LINK.dll

2.2. メソッド・プロパティ

2.2.1. CaoWorkspace::AddController メソッド

CMP-LINK プロバイダは AddController 時に通信用の接続パラメータを参照し, 通信の接続を行います.

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.JTEKT.CMP-LINK"
<bstrPcName> : [in] プロバイダの実行マシン名(未使用)
<bstrOption> : [in] オプション文字列

使用例

```
using(CCaoEngine engine = new CCaoEngine())
{
    CCaoController ctrl = engine.Workspaces[0].AddController(
        "PLC-Link",
        "CaoProv.JTEKT.CMP-LINK",
        null,
        "Conn=TCP:192.168.0.1:5001,Timeout=3000,ConnTimeout=3000");
}
```

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
Conn=<接続パラメータ>	必須．通信形態とその接続パラメータを設定します. (参照 2.2.1.1)
SumCheck[=<TRUE/FALSE>]	シリアル接続時のサムチェックの有効/無効を指定します. (デフォルト：TRUE)
StationNumber[=<局番>]	シリアル接続時の局番指定．8 進数 2 文字 00～37 を指定します. (参照 2.2.1.2)
ConnTimeout[=<タイムアウト時間>]	接続時のタイムアウト時間(ミリ秒)を指定します. （デフォルト：3000）
Timeout[=<タイムアウト時間>]	TCP 通信におけるタイムアウト時間(ミリ秒)を指定します. （デフォルト：3000）

2.2.1.1. 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を示します.

【シリアルデバイス】

“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.

300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.

<Parity> : パリティ．‘E’-EVEN.

<DataBits> : データビット数．‘7’-7bit, ‘8’-8bit.

<StopBits> : ストップビット数．‘1’-1bit, ‘2’-2bit.

【イーサネットデバイス】

TCP/IP

“Conn=TCP:<Dest IP Address>:<Dest Port No>[:<Src IP Address>:<Src Port No>]”

< Dest IP Address > : 接続先の IP アドレス.

< Dest Port No > : 接続先のポート番号.

< Src IP Address > : 自 IP アドレス.

(PLC 側の通信設定が TCP 相手特定パッシブの場合に使用)

< Src Port No > : 自ポート番号.
(PLC 側の通信設定が TCP 相手特定パッシブの場合に使用)

2.2.1.2. StationNumber オプション

StationNumber オプションを指定することでシリアル接続時に局番を指定することが可能です. 未指定時は 00 となります (TCP/IP 接続時には使用しません).

指定された局番はシリアル接続時の通信コマンドに付与されます. 通信コマンドの詳細は各種 PLC の取扱説明書を参照してください.

2.2.2. CaoController::GetVariableNames プロパティ

表 3-1 に示している変数名の一覧を取得します.

2.2.3. CaoController::AddVariable メソッド

表 3-1 に示している変数オブジェクトを作成します. 詳細は各変数の章を参照してください.

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

< bstrVariableName > : [in] 変数名
<bstrOption> : [in] オプション文字列

使用例

```
CCaoVariable variable = ctrl.AddVariable("VAR1", "DeviceType=D,PrgNo=1,Address=0,VT=UI2");
```

2.2.4. CaoController::Execute メソッド

CaoController クラスの Execute メソッドは, コマンドを実行するためのメソッドです. 各コマンドの詳細はコマンドリファレンスを参照してください.

書式 Execute (<bstrCommandName:VT_BSTR>,[<vntParam : VT_VARIANT>])

bstrCommandName: [in] コマンド名
vntParam : [in] パラメータ

2.2.5. CaoVariable::get_Value プロパティ

アクセス対象のデバイスからオプションで指定されたサイズになるように読出しを行うコマンド(I/O レジスタバイト読出し)を送出します. 読み出した結果をオプションで指定されたデータ型に変換して返します.

使用例

```
object value = variable.Value;
```

2.2.6. CaoVariable::put_Value プロパティ

引数で渡された値をオプション指定に従い変換した後，アクセス対象のデバイスに対し書込みを行うコマンド(I/O レジスタバイト書込み)を送出します．

システムで使用されているアドレスに対して書込みを行った場合，put_Value は成功しますが値が変化しない場合があります．

使用例

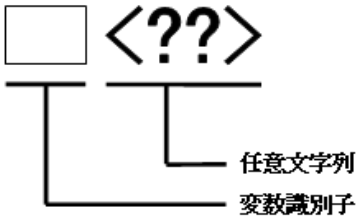
```
variable.Value = 100;
```

3. 変数一覧

各クラスで使用可能な変数一覧を定義します．変数は，CaoVariable クラスのオブジェクトを指します．複数変数を登録(オプションのみ変更したい場合等に有用)するために任意の文字列を付与することが可能です．

変数名に任意文字列を付与するための書式を以下に示します．

複数変数共通指定書式



3.1. CaoController クラス

表 3-1 CaoController クラス 変数一覧

変数名	説明	属性		参照
		get	put	
@MAKER_NAME	メーカー名("JTEKT")を取得します.	○	-	P.10
@VERSION	プロバイダバージョンを取得します.	○	-	P.11
@PLC_MODE	CPU ステータスを取得します.	○	-	P.11
<??>	PLC 内のデバイスへアクセスします.	○	○	P.13

3.1.1. @MAKER_NAME

メーカー名を取得します.

オプション

存在しません.

データ型

データ型	説明
VT_BSTR	"JTEKT"が格納されます.

3.1.2. @VERSION

プロバイダのバージョンを取得します.

オプション

存在しません.

データ型

データ型	説明
VT_BSTR	プロバイダバージョンが格納されます.

3.1.3. @PLC_MODE

CPU ステータスを取得します.

オプション

存在しません.

データ型

データ型		説明
VT_I2 | VT_ARRAY		取得結果を 0 / 1(0: OFF, 1: ON)の配列で格納します. 0~11 はシリアル接続, イーサネット接続共通です. 12 以降はイーサネット接続時のみ存在します.
0		PC3 モード
1		I/O モニタユーザモード
2		デバッグモード中
3		疑似停止中
4		停止要求継続中
5		停止中
6		RUN
7		メモリカードあり
8		I/O 割付パラメータ変更あり
9		アラーム
10		軽故障
11		重故障
以降はイーサネット接続時のみ存在します.		
12		プログラム及び付帯情報書込禁止
13		メモリカード運転
14		システム I/O 書込禁止
15		システム I/O 読出禁止

16		システムメモリ書込禁止
17		システムメモリ読出禁止
18		1 命令ステップ
19		1 プロックスステップ
20		1 スキャンステップ
21		トリガ検出
22		イネーブル検出
23		定周期サンプリングトレース
24		スキャンサンプリングトレース
25		トレース
26		ステータスラッチ設定
27		リモート RUN 設定
28		I/O オフライン
29		RUN 中書込み中
30		RUN 中書込み異常
31		プログラム及び付帯情報書込権限定
32		プログラム 1 運転中
33		プログラム 2 運転中
34		プログラム 3 運転中

3.1.1. <??> (PLC デバイス)

PLC 内のデバイスへアクセスします. データ型は指定により異なります. 以下にプロバイダがアクセス可能な PLC 内のアドレス一覧を定義します。

表 3-2 PLC 内のアドレス一覧

名称	領域	サイズ	アドレス範囲	プログラム番号[1]	デバイス種別[2]	アドレス(ビット)[3]	アドレス(バイト)[4]	アドレス(ワード)[5]
エッジ検出	基本ビット領域	Bit	P0000~P01FF	**1**, 2, 3	P	0~1FF	0~3F	0~1F
PC10 用エッジ検出	PC10 用基本ビット領域	Bit	P1000~P17FF	**1**, 2, 3	P1	-	0~FF	0~7F
キープリレー	基本ビット領域	Bit	K0000~K02FF	**1**, 2, 3	K	0~2FF	0~5F	0~2F
特殊リレー	基本ビット領域	Bit	V0000~V00FF	**1**, 2, 3	V	0~0FF	0~1F	0~F
PC10 用特殊リレー	PC10 用基本ビット領域	Bit	V1000~V17FF	**1**, 2, 3	V1	-	0~FF	0~7F
タイマ/カウンタ	基本ビット領域	Bit	T/C0000~T/C01FF	**1**, 2, 3	T / C	0~1FF	0~3F	0~1F
PC10 用タイマ/カウンタ	PC10 用基本ビット領域	Bit	T/C1000~T/C17FF	**1**, 2, 3	T1 / C1	-	0~FF	0~7F
リンクリレー	基本ビット領域	Bit	L0000~L07FF	**1**, 2, 3	L	0~7FF	0~FF	0~7F
PC10 用リンクリレー	PC10 用基本ビット領域	Bit	L1000~L1FFF	**1**, 2, 3	L1	-	0~1FF	0~FF
PC10 用リンクリレー	PC10 用基本ビット領域	Bit	L2000~L2FFF	**1**, 2, 3	L2	-	0~1FF	0~FF
入力/出力	基本ビット領域	Bit	X/Y0000~X/Y07FF	**1**, 2, 3	X / Y	0~7FF	0~FF	0~7F
内部リレー	基本ビット領域	Bit	M0000~M07FF	**1**, 2, 3	M	0~7FF	0~FF	0~7F

名称	領域	サイズ	アドレス範囲	プログラム番号[1]	デバイス種別[2]	アドレス(ビット)[3]	アドレス(バイト)[4]	アドレス(ワード)[5]
PC10 用内部リレー	PC10 用基本ビット領域	Bit	M1000~M17FF	**1**, 2, 3	M1	-	0~FF	0~7F
特殊レジスタ	基本ワード領域	Word	S0000~S03FF	**1**, 2, 3	S	-	-	0~3FF
PC10 用特殊レジスタ	PC10 用基本ワード領域	Word	S1000~S13FF	**1**, 2, 3	S1	-	-	0~3FF
現在値レジスタ	基本ワード領域	Word	N0000~N01FF	**1**, 2, 3	N	-	-	0~1FF
PC10 用現在値レジスタ	PC10 用基本ワード領域	Word	N1000~N17FF	**1**, 2, 3	N1	-	-	0~7FF
リンクレジスタ	基本ワード領域	Word	R0000~R07FF	**1**, 2, 3	R	-	-	0~7FF
データレジスタ	基本ワード領域	Word	D0000~D2FFF	**1**, 2, 3	D	-	-	0~FFF
データレジスタ	基本ワード領域	Word	D1000~D1FFF	**1**, 2, 3	D1	-	-	0~FFF
データレジスタ	基本ワード領域	Word	D2000~D2FFF	**1**, 2, 3	D2	-	-	0~FFF
拡張エッジ	拡張ビット領域 1	Bit	EP0000~EP0FFF	**0**	EP	-	0~1FF	0~FF
拡張キープリレー	拡張ビット領域 1	Bit	EK0000~EK0FFF	**0**	EK	-	0~1FF	0~FF
拡張特殊リレー	拡張ビット領域 1	Bit	EV0000~EV0FFF	**0**	EV	-	0~1FF	0~FF
拡張タイマー/カウンタ	拡張ビット領域 1	Bit	ET/EC0000~ET/EC07FF	**0**	ET/EC	-	0~FF	0~7F
拡張リンクリレー	拡張ビット領域 1	Bit	EL0000~EL1FFF	**0**	EL	-	0~3FF	0~1FF
拡張入力	拡張ビット領域 1	Bit	EX/EY0000~EX/EY07FF	**0**	EX/EY	-	0~FF	0~7F
拡張内部リレー	拡張ビット領域 1	Bit	EM0000~EM1FFF	**0**	EM	-	0~3FF	0~1FF

名称	領域	サイズ	アドレス範囲	プログラム番号[1]	デバイス種別[2]	アドレス(ビット)[3]	アドレス(バイト)[4]	アドレス(ワード)[5]
拡張特殊レジスタ	拡張ワード領域 1	Word	ES0000~ES07FF	**0**	ES	-	-	0~7FF
拡張現在値レジスタ	拡張ワード領域 1	Word	EN0000~EN07FF	**0**	EN	-	-	0~7FF
拡張設定値レジスタ	拡張ワード領域 1	Word	H0000~H07FF	**0**	H	-	-	0~7FF
拡張入力/出力	拡張ビット領域 2	Bit	GX/GY0000~GX/GYFFFF	**7**	GX / GY	-	0~1FFF	0~FFF
拡張内部リレー	拡張ビット領域 2	Bit	GM0000~GMFFFF	**7**	GM	-	0~1FFF	0~FFF
拡張データレジスタ	拡張ワード領域 2	Word	U000000~U007FFF	**8**	U	-	-	0~7FFF
			U000000~U01FFFF	*	U	-	-	0~1FFFF
拡張ファイルレジスタ	拡張ワード領域 3	Word	EB00000~EB07FFF	**9**	EB	-	-	0~7FFF
			EB08000~EB0FFFF	A	EB	-	-	0~7FFF
			EB10000~EB17FFF	B	EB	-	-	0~7FFF
			EB18000~EB1FFFF	C	EB	-	-	0~7FFF
			EB00000~EB3FFFF	*	EB	-	-	0~3FFFF
フラッシュレジスタ	拡張ワード領域 4	Word	FR000000~FR1FFFFF	*****	FR	-	-	0~1FFFFF

[1] PrgNo オプションに対応します。

[2] DeviceType オプションに対応します。

[3] ビットアドレスにビットアクセスする際に指定する Address オプションに対応します。

[4] イーサネット接続時にビットアドレスにバイトアクセスする際に指定する Address オプションに対応します。

[5] ワードアドレスにアクセスする際、または、シリアル接続時にビットアドレスにバイトアクセスする際に指定する Address オプションに対応します。

3.1.1.1. シリアル接続時

オプション

以下にシリアル接続時、オプション文字列へ指定するリストを示します.

オプション	必須	意味
DeviceType	○	デバイス種別をコードで指定します. 指定可能なデバイス種別は表 3-2 を参照してください.
PrgNo	-	プログラム番号を 1 桁の 16 進文字列で指定します. (参照 3.1.1.1.1)
Address	-	デバイス番号を 16 進文字列で指定します. (参照 3.1.1.1.2)
VT	-	Put/Get するデータ型を指定します. (参照 3.1.1.1.3)
Elem	-	Put/Get するデータの要素数を 10 進数で指定します. （デフォルト:1）
Interval	-	PLC がコマンドを受信してからレスポンスを返すまでの時間を 1 桁の 16 進文字列で指定します. シリアル接続時のみ使用します. (参照 3.1.1.1.4)
Array	-	1 要素の読み込み時に配列型にするかどうかを指定します. （デフォルト:FALSE）

3.1.1.1.1. PrgNo オプション

どのプログラム領域のデバイスにアクセスするか指定します. 指定可能なプログラム番号は表 3-2 のプルグラム番号列を参照してください. PrgNo を省略した場合はデバイスに対応したデフォルト値を使用します. デフォルト値は表 3-2 のプルグラム番号列で太字とアンダーバーで示されます. 「**1**, 2, 3」であればデフォルト値は「1」であることを示しています.

表 3-2 のプルグラム番号列で「*」と表示されている領域は, PC10 で追加されたプログラム番号です. PC10 以降の機種と接続する場合に, プログラム番号に「*」を指定することで, PC10 に追加された領域にアクセスすることが可能となります.

3.1.1.1.2. Address オプション

読み書きする先頭アドレスを16進文字列で指定します. 省略時は0として扱います. アドレス値は, ビット / ワードの 2 種類存在します. どのアドレスを指定したかは, デバイス種別(DeviceType), プログラム番号(PrgNo), 要素数(Elem), データ型(VT)の指定内容から, プロバイダが自動で決定します. アドレス種類ごとに指定可能な範囲は表 3-2 に示します.

また, ワードアドレス指定時, アドレスの末尾に'L','H'を付加することができます. 'L'は下位バイト, 'H'は上位バイトに対してアクセスすることを表します. 省略した場合は指定アドレスの下位バイトを先頭としてア

クセスします. アドレスの表記方法は以下の図を参照してください.

	ビットアドレス	ワードアドレス (シリアル接続時)	
ビットアドレス領域	X000	(LSB)	下位バイト
	X001	↑	
	X002	|	
	X003	X00L	
	X004	|	
	X005	|	
	X006	↓	
	X007	(MSB)	
	X008	(LSB)	上位バイト
	X009	↑	
	X00A	|	
	X00B	X00H	
	X00C	|	
	X00D	|	
	X00E	↓	
	X00F	(MSB)	

	ビットアドレス	ワードアドレス (シリアル接続時)	
ワードアドレス領域	D0000-0	(LSB)	下位バイト
	D0000-1	↑	
	D0000-2	|	
	D0000-3	D0000L	
	D0000-4	|	
	D0000-5	|	
	D0000-6	↓	
	D0000-7	(MSB)	
	D0000-8	(LSB)	上位バイト
	D0000-9	↑	
	D0000-A	|	
	D0000-B	D0000H	
	D0000-C	|	
	D0000-D	|	
	D0000-E	↓	
	D0000-F	(MSB)	

<基本ビット領域(P, K, V, T, C, L, X, Y, M)のアドレス>

プログラム番号, 要素数, データ型の組み合わせにより, 下図の通り判断します. X/Y は同一領域です. ワードアドレスと判断した場合, バイト単位でのアクセスを行い, ビット位置の指定はできません.

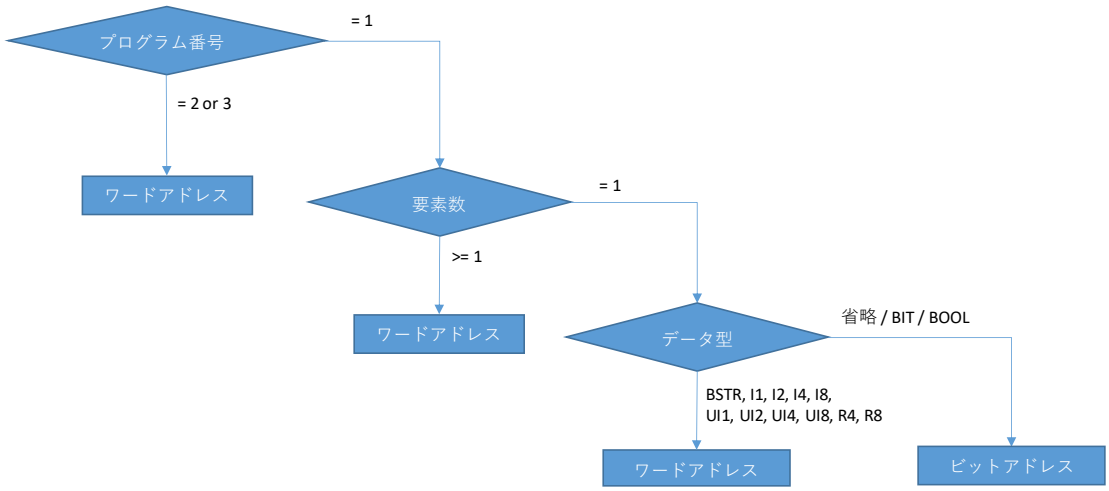


図 3-1 基本ビットデバイスでのアドレス指定

(例) DeviceType=X,Address=0001 (ビットデバイスに単一ビットアクセス)

ビットアドレス X001 に対して 1 ビット読み書きし, アドレスが 1 増えると 1 ビット分移動します.

Address	P1-X00W															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0

Get

Value = { true }

Set

Address	P1-X00W															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0

図 3-2 基本ビットデバイスにビット単位でアクセス

(例) DeviceType=X,Address=0001, Elem=2 (基本ビットデバイスに複数ビットアクセス)

読出し時はワードアドレス X01W を先頭に 2 ビット分読み出します. 書き込み時はワードアドレス X01W を先頭に 1 バイト単位で書き込みます. 0~1 ビットは指定されたデータを書込み、2~7 ビットは 0 埋めします.

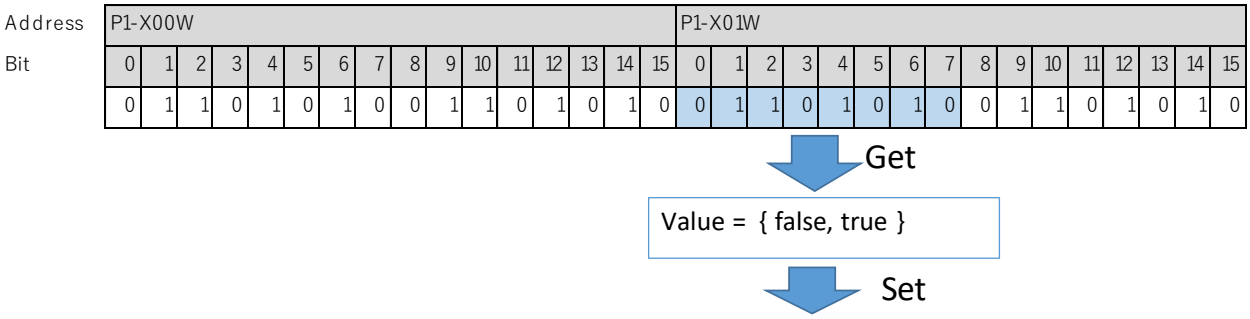


図 3-3 基本ビットデバイスに複数ビットアクセス

(例) DeviceType=X,Address=0001H,VT=UI2,Elem=2(基本ビットデバイスにバイト単位アクセス)
ワードアドレス X01W の上位バイトアドレスを先頭に 2 バイト読み書きし，アドレスが 1 増えると 1 ワード分移動します.

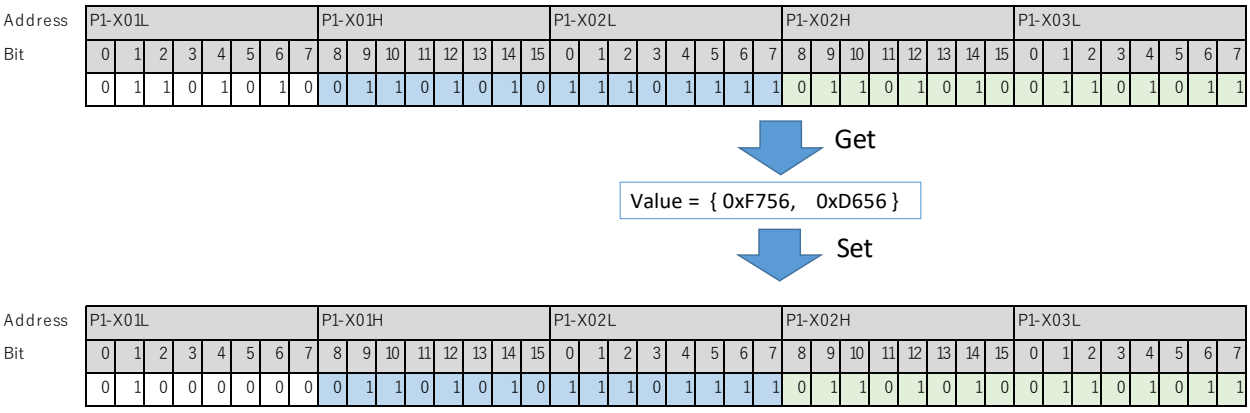


図 3-4 基本ビットデバイスにバイト単位アクセス

<PC10 用基本ビット領域(P1, V1, T1, C1, L1, L2, M1)のアドレス>

常にワードアドレスと判断し，バイト単位でのアクセスを行います. なお，アドレスの最上位 1 桁は DeviceType で指定し，Address には下 3 桁のビットアドレスをもとに計算したワードアドレスを指定してください.

(例) DeviceType=P1, Address=1F, Elem=1(PC10 用基本ビットデバイスに単一バイトアクセス)
基本ビット領域とは動作が異なります. ワードアドレス P11FW を先頭に 1 バイト分読み書きします.

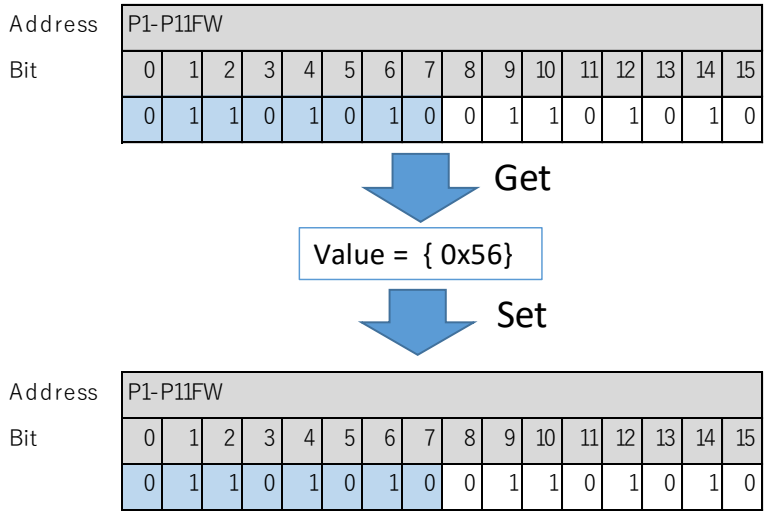


図 3-5 PC10 用基本ビットデバイスに単一バイトアクセス

(例) DeviceType=P1, Address=1F, Elem=2(PC10 用基本ビットデバイスに複数バイトアクセス)

基本ビット領域とは動作が異なります. ワードアドレス P11FW を先頭に 2 バイト分読み書きします.

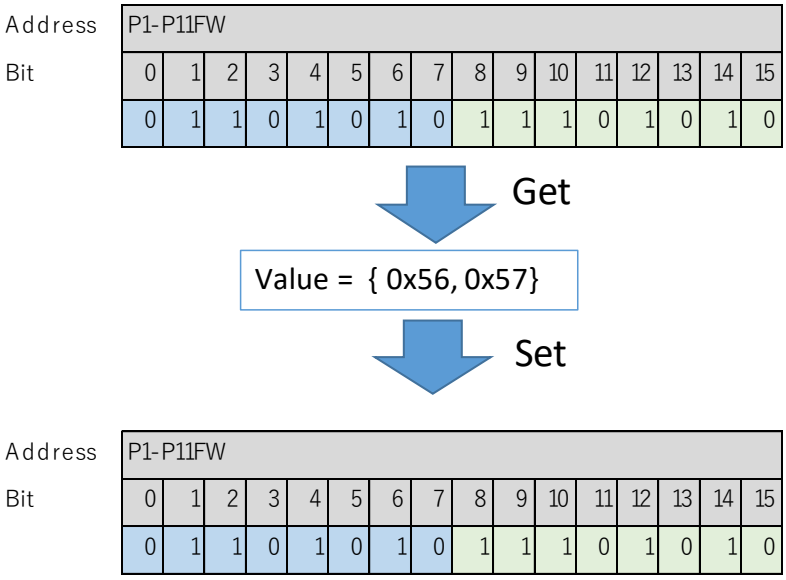


図 3-6 PC10 用基本ビットデバイスに複数バイトアクセス

<基本ワード領域(S, N, R, D)のアドレス>

常にワードアドレスと判断します.

(例) DeviceType=D,Address=0001, Elem=2(ワードアドレスにバイト単位でのアクセス)

ワードアドレス D0001 を先頭に 2 バイト読み書きし, アドレスが 1 増えると 1 ワード分移動します.

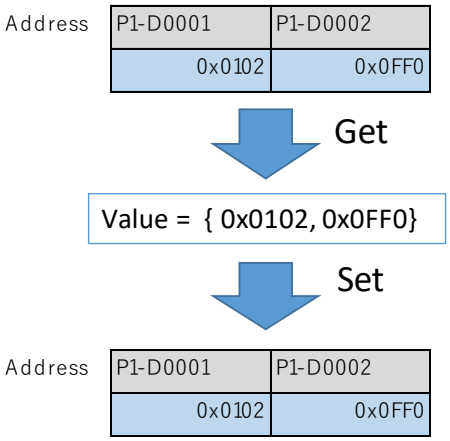


図 3-7 基本ワードデバイスに複数点アクセス

<PC10 用基本ワード領域(S1, N1, D1, D2)のアドレス>

常にワードアドレスと判断します. アドレスの最上位 1 桁は DeviceType で指定し, Address には下 3 桁のワードアドレスを指定してください.

(例) DeviceType=D1,Address=0001,VT=I2(ワードアドレスにバイト単位でのアクセス)

ワードアドレス D1001 を先頭に 2 バイト読み書きし, アドレスが 1 増えると 1 ワード分移動します.

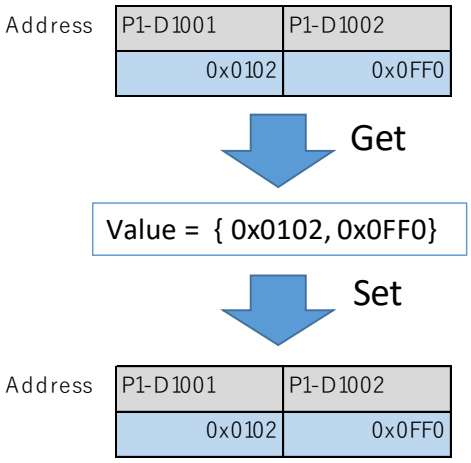


図 3-8 PC10 用基本ワードデバイスに複数点アクセス

<拡張ビット領域 1, 2(EP, EK, EV, ET, EC, EL, EX, EY, EM, GX, GY, GM)のアドレス>

常にワードアドレスと判断し, バイト単位でのアクセスを行います. なお, EX/EY, GX/GY は同一領域です.

(例) DeviceType=EK, Address=01F, Elem=2 (ビットデバイスにバイト単位でのアクセス)

基本ビット領域とは動作が異なります. 読出し時はワードアドレス EK01FW を先頭に 2 バイト分読み書きします.

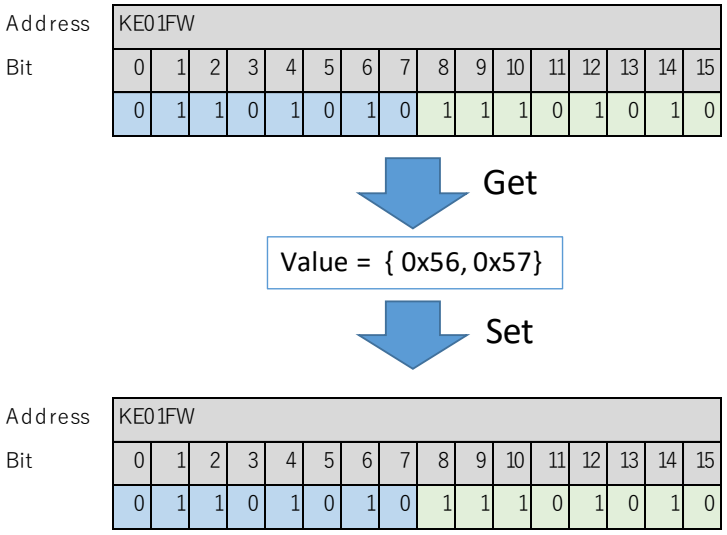


図 3-9 拡張ビットデバイスに複数バイトアクセス

<拡張ワードアドレス領域 1, 2 (ES, EN, H, U)のアドレス>

常にワードアドレスと判断します. なお, PC10 以降に拡張されたアドレス(U008000 以上のアドレス)にアクセスする場合はプログラム番号に"*"を指定してください.

(例) DeviceType=U, Address=0F10, Elem=2 (拡張ワードデバイスにバイト単位でのアクセス)

ワードアドレス U0F10 を先頭に 2 バイト読み書きし, アドレスが 1 増えると 1 ワード分移動します.

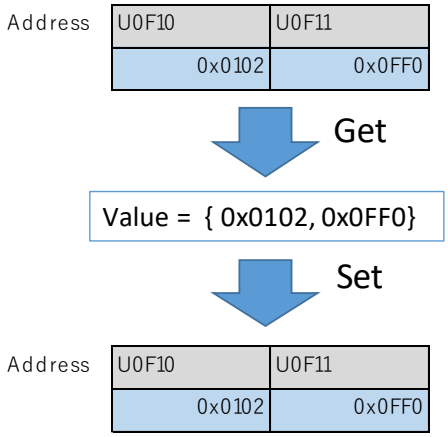


図 3-10 拡張ワードデバイスに複数点アクセス

<拡張ワードアドレス領域 3(EB)のアドレス>

常にワードアドレスで指定します. ただしプログラム番号の指定値が「9」, 「A」, 「B」, 「C」の場合は相対アドレス, 「*」の場合は絶対アドレスの指定となります.

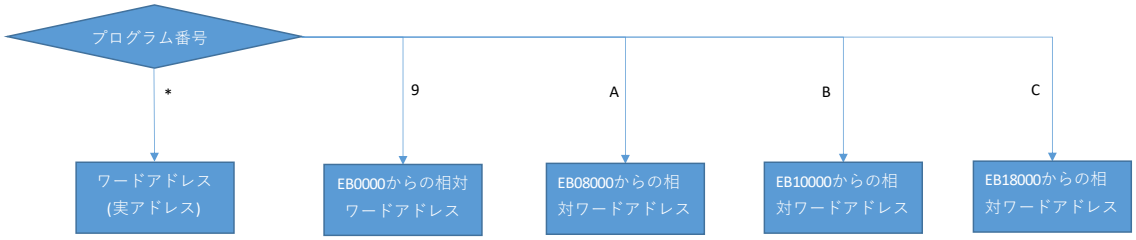


図 3-11 拡張ワードデバイス 3 のアドレス指定

（例）DeviceType=EB, PrgNo=C, Address=001F, Elem=2 (相対アドレスを指定して EB1801F にアクセス)
全機種が対応している指定方法です. PrgNo=C の場合, Address には EB18000 を基準とした相対アドレスを指定します. その他のプログラム番号と基準アドレスの対応は表 3-2 を参照してください. ワードアドレス EB1801F を先頭に 2 バイト分読み書きし, アドレスが 1 増えると 1 ワード分移動します.

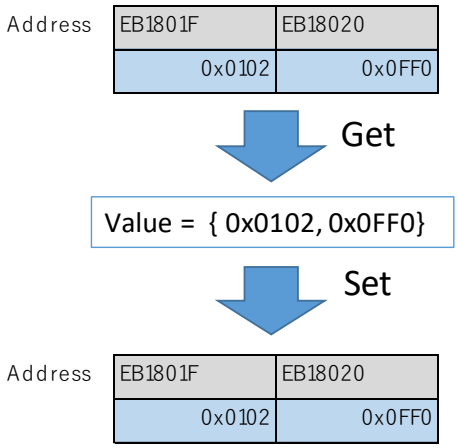


図 3-12 拡張ワードデバイス 3 に相対アドレスでアクセス

（例）DeviceType=EB, PrgNo=*, Address=1801F, Elem=2 (絶対アドレスを指定して EB1801F にアクセス)
PC10 以降の機種が対応している指定方法です. ワードアドレス EB1801F を先頭に 2 バイト分読み書きし, アドレスが 1 増えると 1 ワード分移動します.

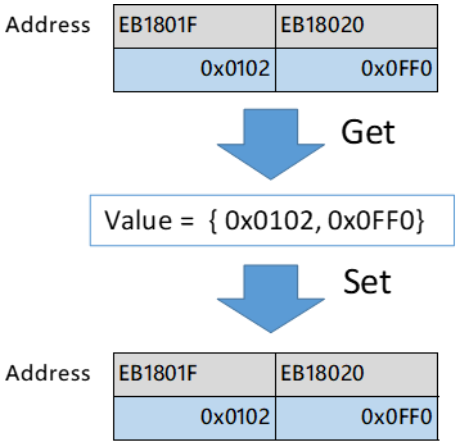


図 3-13 拡張ワードデバイス 3 に絶対アドレスでアクセス

（例）DeviceType=EB, PrgNo=*, Address=2FFFF (絶対アドレスを指定して EB2FFFF にアクセス)

PC10 で拡張されたアドレスには相対アドレスを指定したアクセスはできません．ワードアドレス EB2FFFF を先頭に 2 バイト分読み書きし，アドレスが 1 増えると 1 ワード分移動します．

<拡張ワードアドレス領域 4(FR)のアドレス>

常にワードアドレスと判断します．

3.1.1.1.3. VT オプション

読み書きするデータ型を指定します．指定は VT 文字列または対応する VARTYPE の値(10 進数数値)のいずれかで行います．

VT オプション省略時は，デバイスのアドレス種別に合わせたデータ型を使用します．

使用可能なデータ型の一覧を以下に示します．

DeviceType オプションに基本ビット領域(P, K, V, T, C, L, X, Y, M)以外を指定した場合は VT に BIT および BOOL は指定することができません．

表 3-3 データ型一覧

VT	データ型	説明
BIT	VT_UI1	データを 0/1 の 2 値に変換して書込み/読出し
BOOL	VT_BOOL	データを 0/1 の 2 値に変換して書込み/読出し
BSTR	VT_BSTR	Elem バイトの ASCII として書込み/読出し
I1	VT_I1	1 バイトデータ(符号あり)として書込み/読出し
I2	VT_I2	2 バイトデータ(符号あり)として書込み/読出し
I4	VT_I4	4 バイトデータ(符号あり)として書込み/読出し

I8	VT_I8	8 バイトデータ(符号あり)として書込み/読出し
UI1	VT_UI1	1 バイトデータ(符号なし)として書込み/読出し
UI2	VT_UI2	2 バイトデータ(符号なし)として書込み/読出し
UI4	VT_UI4	4 バイトデータ(符号なし)として書込み/読出し
UI8	VT_UI8	8 バイトデータ(符号なし)として書込み/読出し
R4	VT_R4	4 バイトデータ(浮動小数)として書込み/読出し
R8	VT_R8	8 バイトデータ(倍精度浮動小数)として書込み/読出し

表 3-4 未指定時のデータ型

アドレス種別	VT	データ型
基本ビット領域	BIT	VT_UI1
PC10 用基本ビット領域	UI1	VT_UI1
拡張ビット領域 1, 2	UI1	VT_UI1
基本ワード領域	UI2	VT_UI2
PC10 用基本ワード領域	UI2	VT_UI2
拡張ワード領域 1, 2, 3, 4	UI2	VT_UI2

3.1.1.1.4. Interval オプション

PLC がコマンドを受信してからレスポンスを送出するまでの時間を 16 進文字列(0～F)で指定します.
指定文字列と送出時間の対応は以下の通りです. 省略時は 0 を使用します.

表 3-5 レスポンス送出時間

Interval	応答時間(ms)	Interval	応答時間(ms)
0	0	8	80
1	10	9	90
2	20	A	100
3	30	B	200
4	40	C	300
5	50	D	400
6	60	E	500
7	70	F	600

(例) レスポンス時間を 600ms に指定する場合

DeviceType=D,PrgNo=1,Address=FFF,Interval=F

3.1.1.2. イーサネット接続時

以下にイーサネット接続時、オプション文字列へ指定するリストを示します.

オプション

以下にシリアル接続時、オプション文字列へ指定するリストを示します.

オプション	必須	意味
DeviceType	○	デバイス種別をコードで指定します. 指定可能なデバイス種別は表 3-2 を参照してください.
PrgNo	-	プログラム番号を 1 桁の 16 進文字列で指定します. (参照 3.1.1.2.1)
Address	-	デバイス番号を 16 進文字列で指定します. (参照 3.1.1.2.2)
VT	-	Put/Get するデータ型を指定します. (参照 3.1.1.2.3)
Elem	-	Put/Get するデータの要素数を 10 進数で指定します. （デフォルト:1）
Array	-	1 要素の読み込み時に配列型にするかどうかを指定します. （デフォルト:FALSE）

3.1.1.2.1. PrgNo オプション

どのプログラム領域のデバイスにアクセスするか指定します. 指定可能なプログラム番号は表 3-2 のプルグラム番号列を参照してください. PrgNo を省略した場合はデバイスに対応したデフォルト値を使用します. デフォルト値は表 3-2 のプルグラム番号列で太字とアンダーバーで示されます.「**1**, 2, 3」であればデフォルト値は「1」であることを示しています.

表 3-2 のプルグラム番号列で「*」と表示されている領域は, PC10 で追加されたプログラム番号です. PC10 以降の機種と接続する場合に, プログラム番号に「*」を指定することで, PC10 に追加された領域にアクセスすることが可能となります.

3.1.1.2.2. Address オプション

読み書きする先頭アドレスを16進文字列で指定します. 省略時は0として扱います. アドレス値は, ビット / バイト / ワードの 3 種類存在します. どのアドレスを指定したかは, デバイス種別(DeviceType), プログラム番号(PrgNo), 要素数(Elem), データ型(VT)の指定内容から, プロバイダが自動で決定します. アドレス種別ごとに指定可能な範囲は表 3-2 に示します.

<基本ビット領域(P, K, V, T, C, L, X, Y, M)のアドレス>

プログラム番号, 要素数, データ型の組み合わせにより, 下図の通り判断します. X/Y は同一領域です. バイトアドレスと判断した場合, バイト単位でのアクセスを行い, ビット位置の指定はできません. 通信に使用

するコマンドの制約上，ビットアドレス指定でアクセスする場合，プログラム番号に 1 以外を指定するとエラーになります．

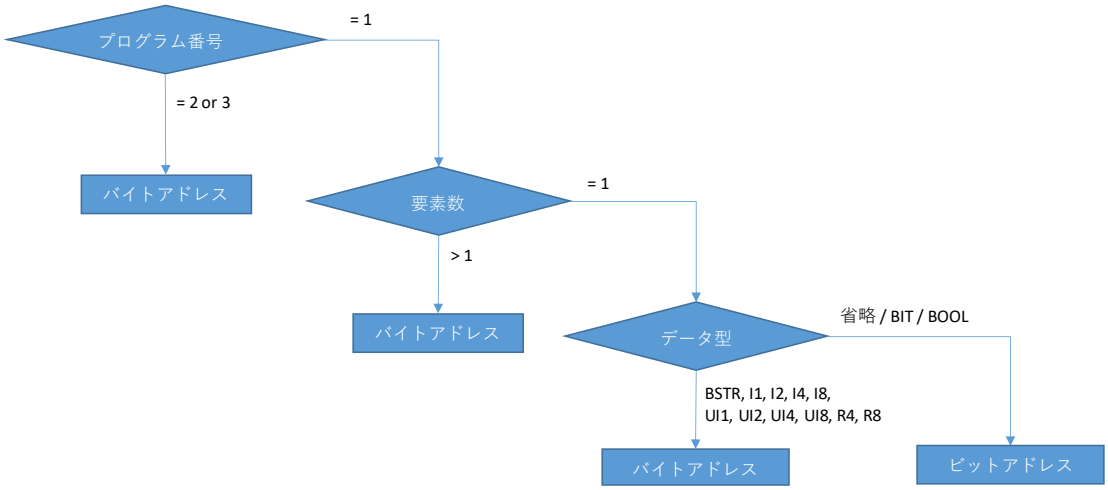


図 3-14 基本ビットデバイスでのアドレス指定

（例）DeviceType=X,Address=0011（ビットデバイスに単一ビットアクセス）

ビットアドレス X0011 に対して 1 ビット読み書きし，アドレスが 1 増えると 1 ビット分移動します．

Address	P1-X01W															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0

Get

Value = { true }

Set

Address	P1-X01W															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0

図 3-15 基本ビットデバイスにビット単位でアクセス

（例）DeviceType=X,Address=0011, Elem=2（基本ビットデバイスに複数ビットアクセス）

バイトアドレスの指定となります．読出し時はワードアドレス X08W の上位バイトを先頭に 2 ビット分読み出します．書き込み時はワードアドレス X08W の上位バイトを先頭に 1 バイト単位で書き込みます．0~1 ビットは指定されたデータを書込み、2~7 ビットは 0 埋めします．

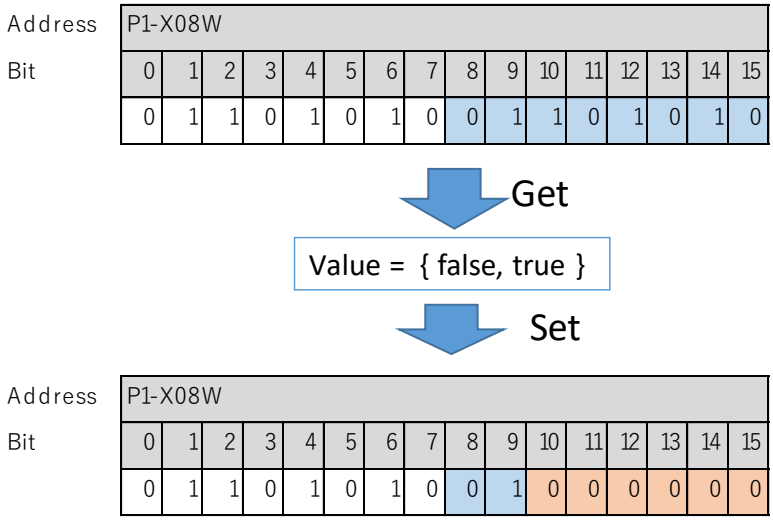


図 3-16 基本ビットデバイスに複数ビットアクセス

(例) DeviceType=X,Address=0001,VT=UI1,Elem=2(基本ビットデバイスにバイト単位アクセス)
ワードアドレス X00W の上位バイトを先頭に 2 バイト読み書きし，アドレスが 1 増えると 1 バイト分移動します.

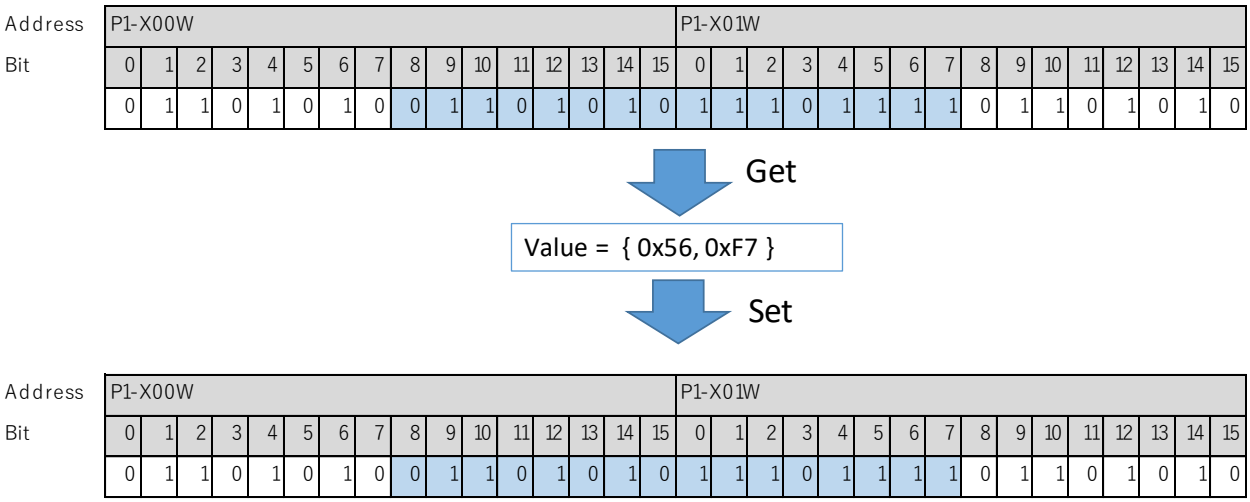


図 3-17 基本ビットデバイスにバイト単位アクセス

<PC10 用基本ビット領域(P1, V1, T1, C1, L1, L2, M1)のアドレス>

常にバイトアドレスと判断し，バイト単位でのアクセスを行います. なお，アドレスの最上位 1 桁は DeviceType で指定し，Address には下 3 桁のビットアドレスをもとに計算したバイトアドレスを指定してください.

(例) DeviceType=P1, Address=3E, Elem=1(PC10 用基本ビットデバイスに単一バイトアクセス)

基本ビット領域とは動作が異なります. ワードアドレスP11FWの下位バイトを先頭に1バイト分読み書きします.

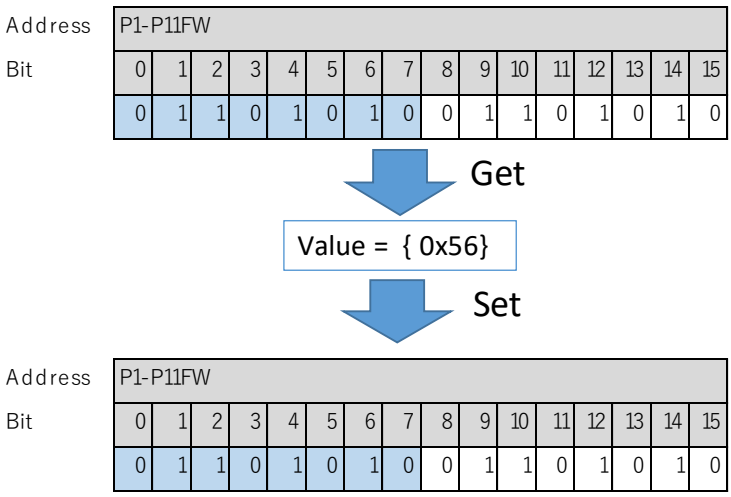


図 3-18 PC10 用基本ビットデバイスに単一バイトアクセス

(例) DeviceType=P1, Address=3E, Elem=2(PC10 用基本ビットデバイスに複数バイトアクセス)

基本ビット領域とは動作が異なります. ワードアドレスP11FWの下位バイトを先頭に2バイト分読み書きします.

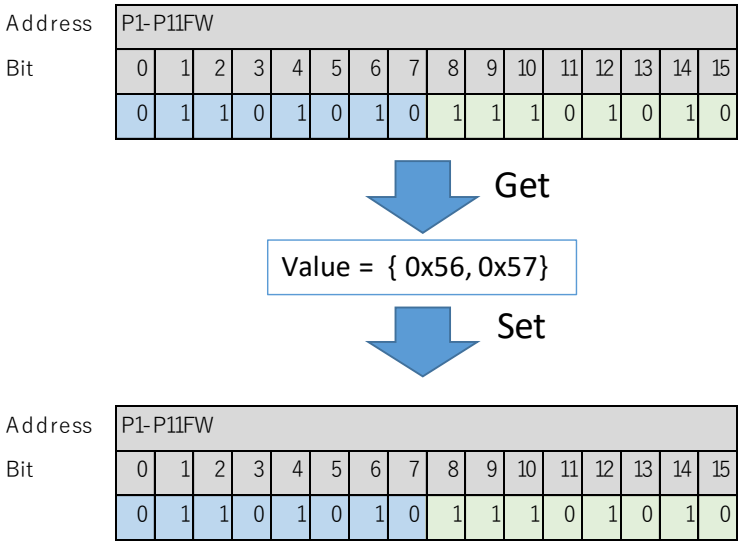


図 3-19 PC10 用基本ビットデバイスに複数バイトアクセス

<基本ワード領域(S, N, R, D)のアドレス>

常にワードアドレスと判断します.

(例) DeviceType=D,Address=0001, Elem=2(ワードアドレスにバイト単位でのアクセス)

ワードアドレス D0001 を先頭に 2 バイト読み書きし，アドレスが 1 増えると 1 ワード分移動します．

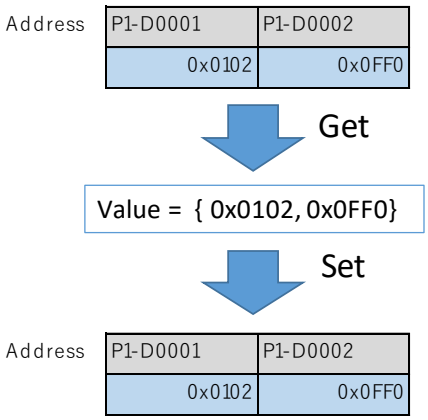


図 3-20 基本ワードデバイスに複数点アクセス

<PC10 用基本ワード領域(S1, N1, D1, D2)のアドレス>

常にワードアドレスと判断します．アドレスの最上位 1 桁は DeviceType で指定し，Address には下 3 桁のワードアドレスを指定してください．

（例）DeviceType=D1,Address=0001,VT=I2(ワードアドレスにバイト単位でのアクセス)

ワードアドレス D1001 を先頭に 2 バイト読み書きし，アドレスが 1 増えると 1 ワード分移動します．

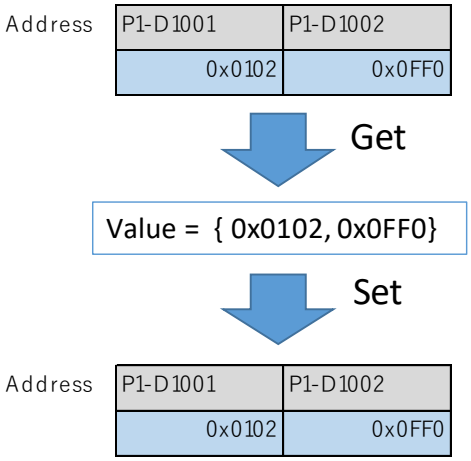


図 3-21 PC10 用基本ワードデバイスに複数点アクセス

<拡張ビット領域 1, 2(EP, EK, EV, ET, EC, EL, EX, EY, EM, GX, GY, GM)のアドレス>

常にバイトアドレスと判断し，バイト単位でのアクセスを行います．なお，EX/EY，GX/GY は同一領域です．

（例）DeviceType=EK, Address=01F, Elem=2（ビットデバイスにバイト単位でのアクセス）

基本ビット領域とは動作が異なります．読出し時はワードアドレス EK0FW の上位バイトを先頭に 2 バ

イト分読み書きします.

Address	KE00FW																KE01FW													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	1	1	0	1	1	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	1	1	1	1	1	1	0	1	0	1	0

Get

Value = { 0x56, 0xF6}

Set

Address	KE00FW																KE01FW													
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	1	1	0	1	1	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	1	1	1	1	1	1	0	1	0	1	0

図 3-22 拡張ビットデバイスに複数バイトアクセス

<拡張ワードアドレス領域 1, 2 (ES, EN, H, U)のアドレス>

常にワードアドレスと判断します. なお, PC10 以降に拡張されたアドレス(U008000 以上のアドレス)にアクセスする場合はプログラム番号に"*"を指定してください.

(例) DeviceType=U, Address=0F10, Elem=2(拡張ワードデバイスにバイト単位でのアクセス)

ワードアドレス U0F10 を先頭に 2 バイト読み書きし, アドレスが 1 増えると 1 ワード分移動します.

Address	U0F10	U0F11
	0x0102	0x0FF0

Get

Value = { 0x0102, 0x0FF0}

Set

Address	U0F10	U0F11
	0x0102	0x0FF0

図 3-23 拡張ワードデバイスに複数点アクセス

<拡張ワードアドレス領域 3(EB)のアドレス>

常にワードアドレスで指定します. ただしプログラム番号の指定値が「9」,「A」,「B」,「C」の場合は相対アドレス,「*」の場合は絶対アドレスの指定となります.

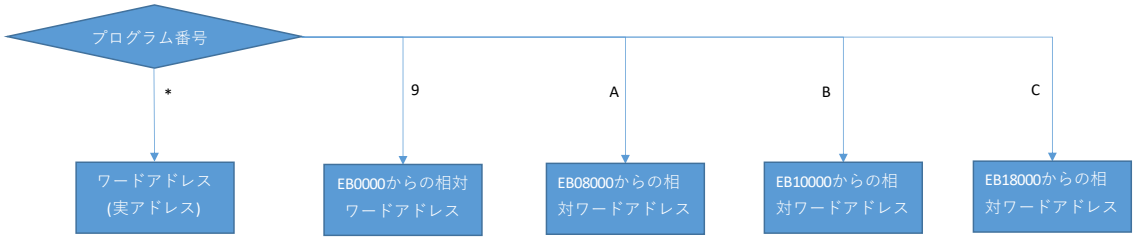


図 3-24 拡張ワードデバイス 3 のアドレス指定

（例）DeviceType=EB, PrgNo=C, Address=001F, Elem=2 (相対アドレスを指定して EB1801F にアクセス)
全機種が対応している指定方法です. PrgNo=C の場合, Address には EB18000 を基準とした相対アドレスを指定します. その他のプログラム番号と基準アドレスの対応は表 3-2 を参照してください. ワードアドレス EB1801F を先頭に 2 バイト分読み書きし, アドレスが 1 増えると 1 ワード分移動します.

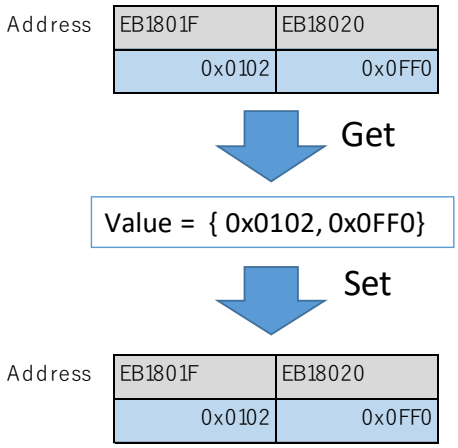


図 3-25 拡張ワードデバイス 3 に相対アドレスでアクセス

（例）DeviceType=EB, PrgNo=*, Address=1801F, Elem=2 (絶対アドレスを指定して EB1801F にアクセス)
PC10 以降の機種が対応している指定方法です. ワードアドレス EB1801F を先頭に 2 バイト分読み書きし, アドレスが 1 増えると 1 ワード分移動します.

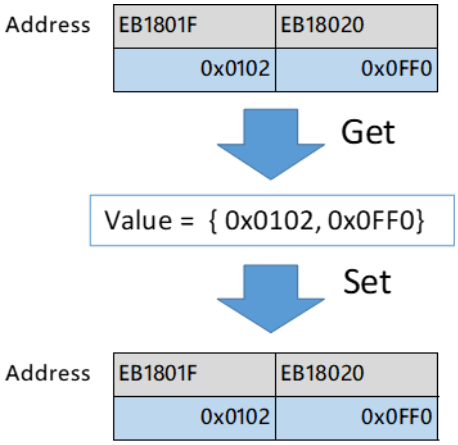


図 3-26 拡張ワードデバイス 3 に絶対アドレスでアクセス

（例）DeviceType=EB, PrgNo=*, Address=2FFFF (絶対アドレスを指定して EB2FFFF にアクセス)

PC10 で拡張されたアドレスには相対アドレスを指定したアクセスはできません．ワードアドレス EB2FFFF を先頭に 2 バイト分読み書きし，アドレスが 1 増えると 1 ワード分移動します．

<拡張ワードアドレス領域 4(FR)のアドレス>

常にワードアドレスと判断します．

3.1.1.2.3. VT オプション

読み書きするデータ型を指定します．指定は VT 文字列または対応する VARTYPE の値(10 進数数値)のいずれかで行います．

VT オプション省略時は，デバイスのアドレス種別に合わせたデータ型を使用します．

使用可能なデータ型の一覧を以下に示します．

DeviceType オプションに P，K，V，T，C，L，X，Y，M 以外を指定した場合は VT に BIT および BOOL は指定することができません．

表 3-6 データ型一覧

VT	データ型	説明
BIT	VT_UI1	データを 0/1 の 2 値に変換して書込み/読出し
BOOL	VT_BOOL	データを 0/1 の 2 値に変換して書込み/読出し
BSTR	VT_BSTR	Elem バイトの ASCII として書込み/読出し
I1	VT_I1	1 バイトデータ(符号あり)として書込み/読出し
I2	VT_I2	2 バイトデータ(符号あり)として書込み/読出し
I4	VT_I4	4 バイトデータ(符号あり)として書込み/読出し

I8	VT_I8	8 バイトデータ(符号あり)として書込み/読出し
UI1	VT_UI1	1 バイトデータ(符号なし)として書込み/読出し
UI2	VT_UI2	2 バイトデータ(符号なし)として書込み/読出し
UI4	VT_UI4	4 バイトデータ(符号なし)として書込み/読出し
UI8	VT_UI8	8 バイトデータ(符号なし)として書込み/読出し
R4	VT_R4	4 バイトデータ(浮動小数)として書込み/読出し
R8	VT_R8	8 バイトデータ(倍精度浮動小数)として書込み/読出し

表 3-7 未指定時のデータ型

アドレス種別	VT	データ型
基本ビット領域	BIT	VT_UI1
PC10 用基本ビット領域	UI1	VT_UI1
拡張ビット領域 1, 2	UI1	VT_UI1
基本ワード領域	UI2	VT_UI2
PC10 用基本ワード領域	UI2	VT_UI2
拡張ワード領域 1, 2, 3, 4	UI2	VT_UI2

4. コマンドリファレンス

4.1. CaoController クラス

表 4-1 CaoController クラス コマンド一覧

コマンド	機能	ページ
ステータス取得		
GetStatus	CPU ステータス読出しコマンドを送信します.	35
CPU 操作		
StopScan	スキャン停止コマンドを送出します.	35
AwakeStopScan	スキャン停止解除コマンドを送出します.	36
StartScan	スキャン再開コマンドを送出します. [2]	36
Reset	リセットコマンドを送出します.	37

4.1.1. CaoController::Execute ("GetStatus") コマンド

PLC に CPU ステータス読出しコマンドを送出します. システム変数「@PLC_MODE」の get_Value と同様の結果を返します.

書式 GetStatus(<Interval>)

Interval : [in] コマンドを受信してからレスポンスを送出するまでの時間.
16 進文字列(0～F)で指定. シリアル接続時のみ使用. (VT_BSTR)

戻り値 : [out] CPU ステータスの一覧(VT_ARRAY | VT_I2)
配列内の各データは表 3-1 の「@PLC_MODE」を参照.

使用例

```
object status = ctrl.Execute("GetStatus", "0");
```

4.1.2. CaoController::Execute ("StopScan") コマンド

PLC にスキャン停止コマンドを送出します.

書式 StopScan(<PrgNo>, <Interval>)

PrgNo : [in] スキャンを停止するプログラム番号. シリアル接続時のみ使用.
(VT_BSTR)
0:プログラム全体, 2:プログラム 2, 3:プログラム 3

[2] スキャンを再開するには, スキャン停止解除が実行されている必要があります. 詳細は各 PLC の取扱説明書を参照してください.

Interval [in] コマンドを受信してからレスポンスを送出するまでの時間.
16 進文字列(0～F)で指定. シリアル接続時のみ使用. (VT_BSTR)
戻り値 : なし

使用例

```
ctrl.Execute("StopScan", new object[] { "0", "0" });
```

4.1.3. CaoController::Execute ("AwakeStopScan ") コマンド

PLC にスキャン停止解除コマンドを送出します.

書式 AwakeStopScan (<PrgNo>, <Interval>)

PrgNo : [in] スキャン停止解除をするプログラム番号.
シリアル接続時のみ使用. (VT_BSTR)
0:プログラム全体, 2:プログラム 2, 3:プログラム 3
Interval [in] コマンドを受信してからレスポンスを送出するまでの時間.
16 進文字列(0～F)で指定. シリアル接続時のみ使用. (VT_BSTR)
戻り値 : なし

使用例

```
ctrl.Execute("AwakeStopScan", new object[] { "0", "0" });
```

4.1.4. CaoController::Execute ("StartScan") コマンド

PLC にスキャン再開コマンドを送出します.

書式 StartScan (<PrgNo >, <Interval>)

PrgNo : [in] スキャンを再開するプログラム番号. シリアル接続時のみ使用.
(VT_BSTR)
0:プログラム全体, 2:プログラム 2, 3:プログラム 3
Interval [in] コマンドを受信してからレスポンスを送出するまでの時間.
16 進文字列(0～F)で指定. シリアル接続時のみ使用. (VT_BSTR)
戻り値 : なし

使用例

```
ctrl.Execute("StartScan", new object[] { "0", "0" });
```

4.1.5. CaoController::Execute ("Reset") コマンド

PLC にリセットコマンドを送出します.

書式 Reset(<Interval>)

Interval		[in] コマンドを受信してからレスポンスを送出するまでの時間. 16 進文字列(0～F)で指定. シリアル接続時のみ使用. (VT_BSTR)
戻り値	:	なし

使用例

```
ctrl.Execute("Reset", "0");
```

5. サンプルプログラム

以下にアドレス"D0000"から I2 の値設定，取得をする場合の C#のサンプルを示します．

List 5-1

```
…(略)…
using ORiN2.ManagedCAO;

namespace CMP_LINK_Sample
{
    public partial class Sample : Form
    {
        private CCaoEngine engine;
        private CCaoWorkspaces wss;
        private CCaoWorkspace ws;
        private CCaoControllers ctrls;
        private CCaoController ctrl;
        private CCaoVariable value;

        public Sample()
        {
            InitializeComponent();
        }

        private void Sample_Load(object sender, EventArgs e)
        {
            // Caoエンジンの生成
            this.engine = new CCaoEngine();

            this.wss = this.engine.Workspaces;
            this.ws = this.wss[0];
            this.ctrls = this.ws.Controllers;

            // 接続オプション(TCP)
            string option = "Conn=TCP:192.168.0.100:6000";

            // シリアル接続の場合は以下のようにする
            // string option = "Conn=COM:1:115200:E:8:1,StationNumber=37";

            // CMP-LINKへの接続
            this.ctrl = this.ws.AddController("Sample",
                                              "CaoProv.JTEKT.CMP-LINK",
                                              "",
                                              option);

            // アクセス用変数追加
            this.value = ctrl.AddVariable("Item", "DeviceType=D,Address=0000,VT=I2");
        }

        private void btnPut_Click(object sender, EventArgs e)
        {
            // 値の設定
            this.value.Value = Int16.Parse(txtItemValue.Text);
        }

        private void btnGet_Click(object sender, EventArgs e)
        {
            // 値の取得
            Int16 val = (Int16)this.value.Value;
            txtItemValue.Text = val.ToString();
        }
```

```
        private void Sample_FormClosing(object sender, FormClosingEventArgs e)
        {
            // Cao関連オブジェクトの開放
            if (this.engine != null)
            {
                this.engine.Dispose();
                this.engine = null;
            }
        }
    }
}
```

6. エラーコード

CMP-LINK プロバイダでは, 以下の固有のエラーコードを返します.

表 6-1 固有エラーコード

エラー名	エラー番号	説明
応答なし	0x80100000	PLC からデータが受信できない場合に返します. PLC の通信設定と接続オプションが一致しているか確認してください.
受信データ異常	0x80100001	チェックサムが一致しない場合や, 想定外のデータを受信した場合に返します.
受信データ欠損	0x80100002	受信データに欠落(最小データサイズ未満等)が見られた場合に返します.
データ変換エラー	0x80100003	エラー応答のコードや受信データの型変換に失敗した場合に返します.
書込みモード変更エラー	0x80100004	シリアル接続時のデータ書込みで, 書込みモードへの切り替えに失敗した場合に返します.
エラー応答	0x801001XX	コマンドの応答結果としてエラーを受信した場合に返します. PLC から受信した 16 進数のエラーコード([3])が XX に挿入されます. 例) 21 → 0x80100121 3F → 0x8010013F

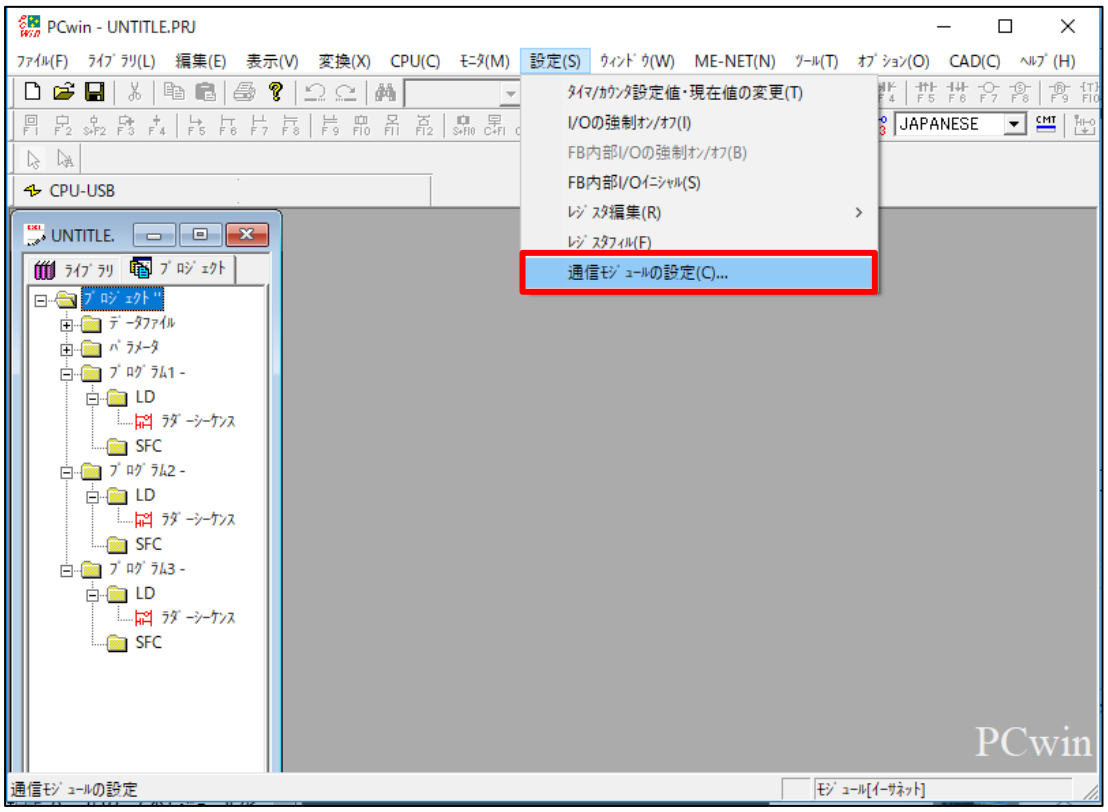
[3] エラーコードの詳細に関しては各 PLC の取扱説明書を参照してください.

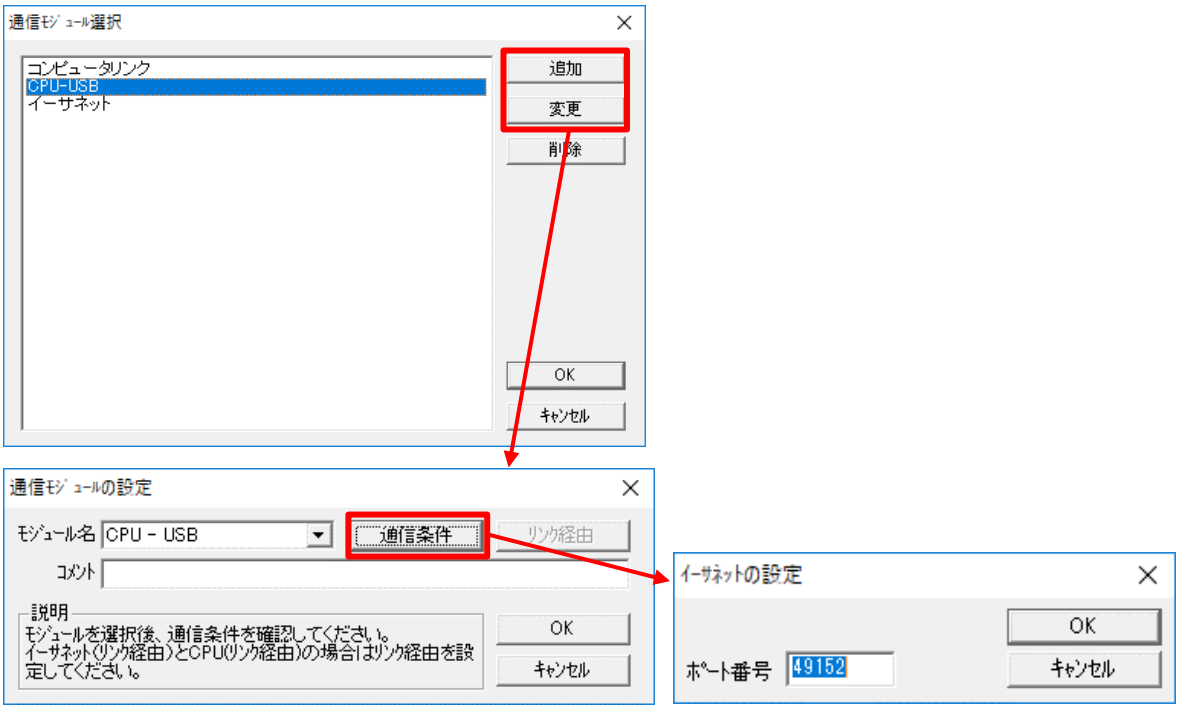
7. アプリケーション開発のための環境セットアップ

本章では，クライアント PC から CMP-LINK プロバイダを用いて PC10G シリーズに接続するために必要な PC10G の通信設定について説明します．

7.1. 事前準備

通信設定には PC10G 付属の設定ツール PCwin を用います． 通信設定を行う前に PCwin のインストールと現在の設定の読み出しを行います．PCwin の詳細については，PCwin 付属のマニュアルを参照してください．

1	クライアント PC に PCwin をインストールし，クライアント PC と PC10G の CPU を USB ケーブルで接続します．
2	PCwin を起動し，USB 接続で CPU と通信するための設定を行います． メニューの[設定]-[通信モジュールの設定]を選択します． 
3	通信モジュール選択画面で[追加]を押します． (既に一覧に CPU-USB が表示されている場合は，[変更]を押します．)



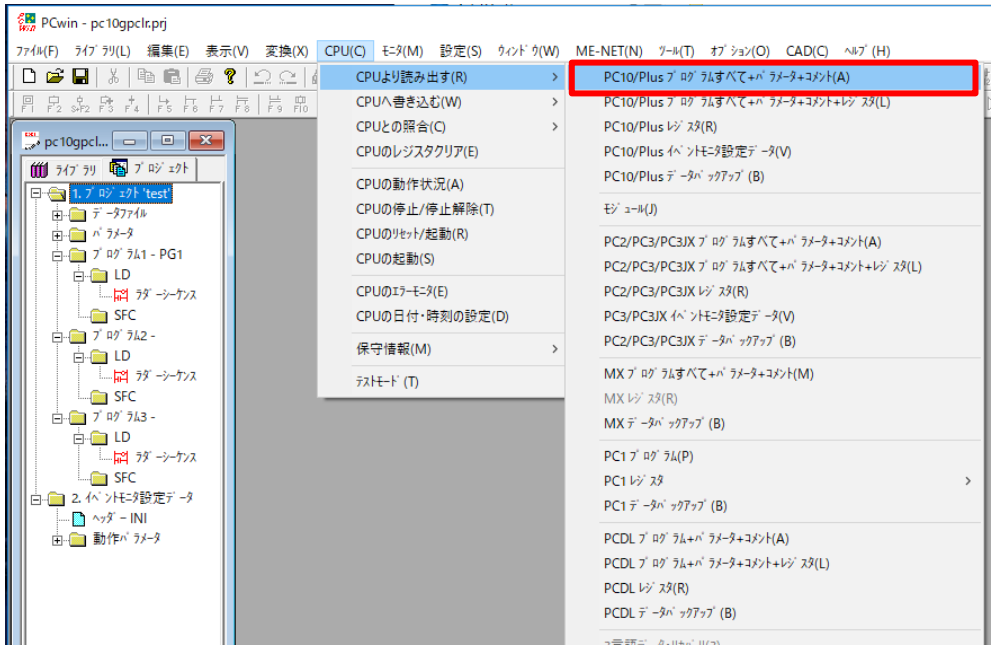
通信モジュールの設定画面でモジュール名を[CPU-USB]に設定します.

ポートを指定する場合は[通信条件]を押して, ポート番号を入力します.

4 通信モジュール選択画面に戻り, [CPU-USB]の行を選択した状態で[OK]ボタンを押し, 設定を終了します.

5 CPU から現在の設定を読み出します.

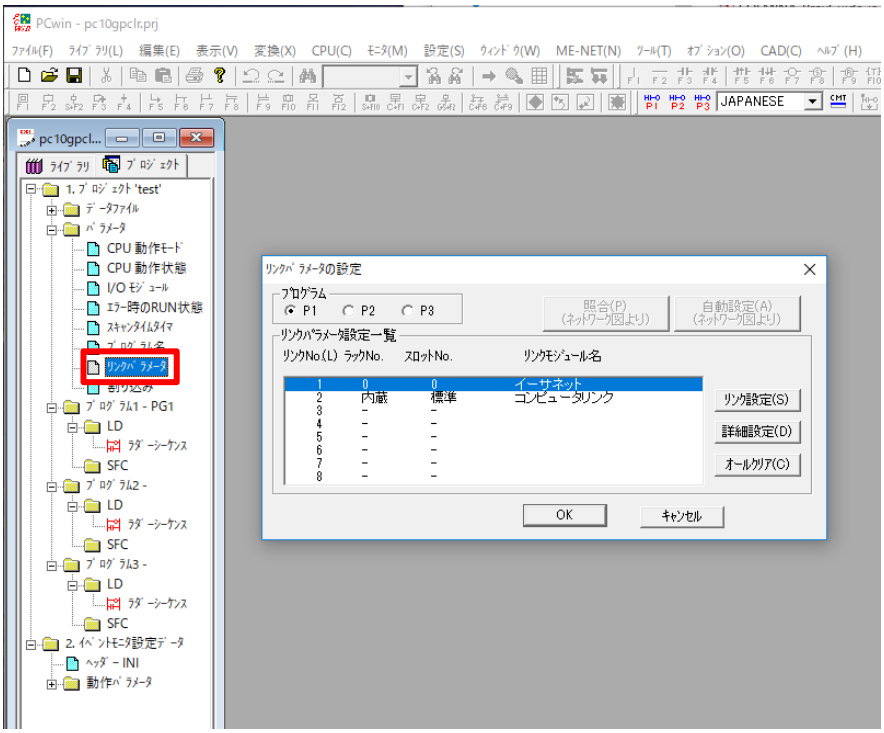
メニューの[CPU]-[CPU より読み出す]-[PC10/Plus プログラム全て+パラメータ+コメント]を選択します.



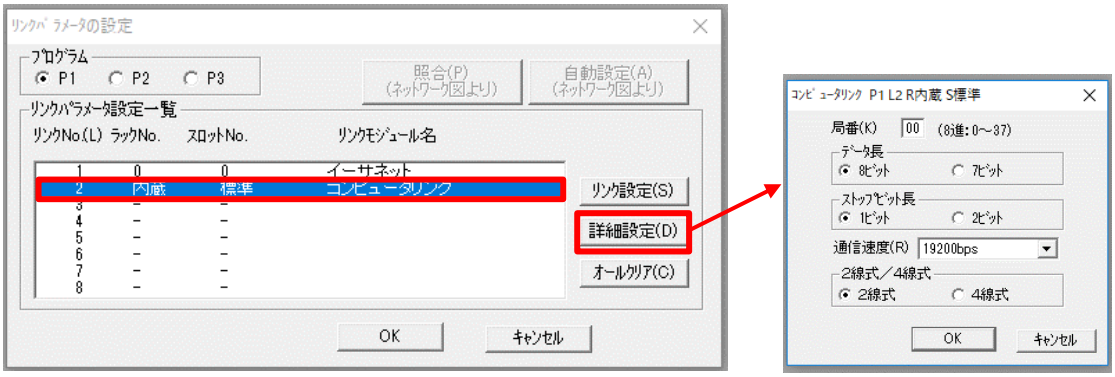
7.2. シリアル通信の設定

シリアル通信(コンピュータリンク)を使用する場合の設定を説明します. 事前に 7.1 を参照し事前準備を行ってください.

1 PCwin のプロジェクトメニューから[リンクパラメータ]を選択します.
リンクパラメータ設定画面で接続したいプログラム(P1～P3)に対してコンピュータリンクの設定を行います.

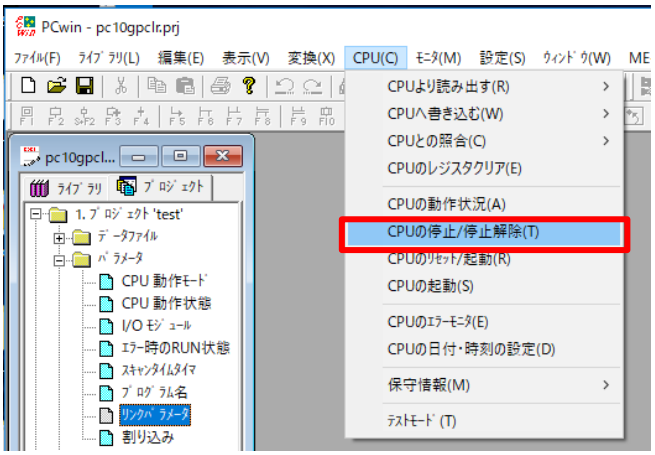


2 ラック No に[内臓], スロット No に[標準], リンクモジュール名に[コンピュータリンク]を設定します.
詳細設定ボタンを押し, シリアル通信のパラメータを設定します.

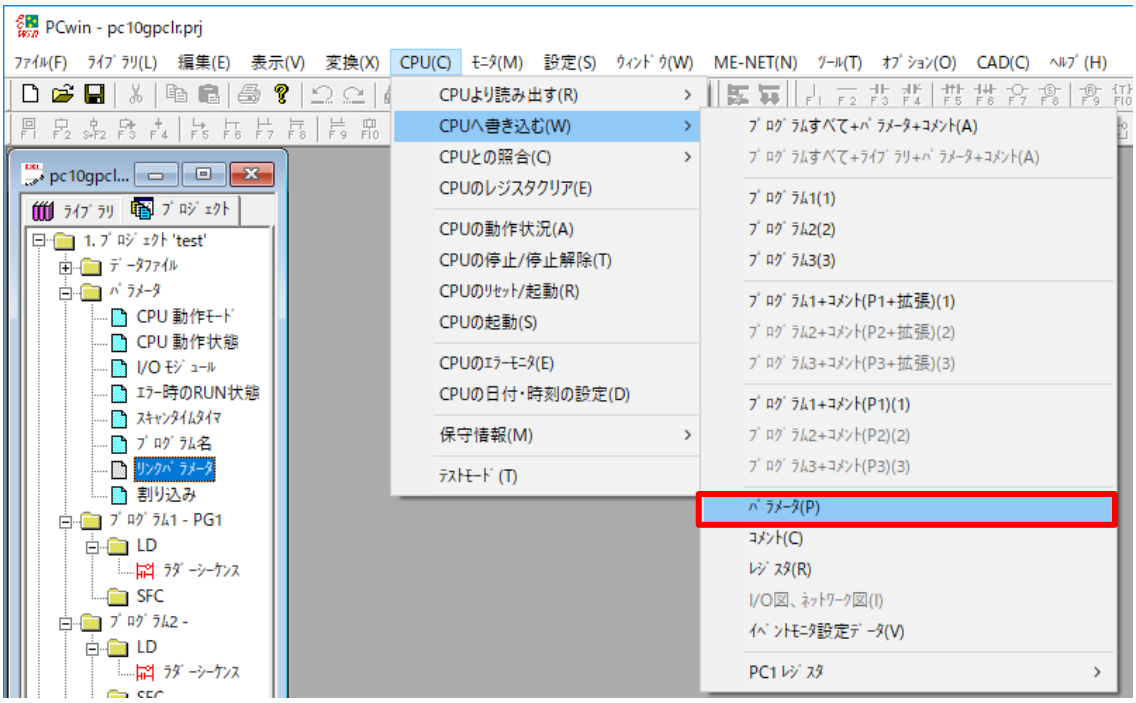


※画面では 2 線式/4 線式が選択できますが, PC10G は 2 線式のみです.

3 CPU にパラメータを書き込むには CPU が停止している必要があります.
メニューの[CPU]-[CPU の停止/停止解除]を選択し, CPU を停止します.



4 メニューの[CPU]-[CPU へ書き込む]を選択し, 設定したパラメータを CPU に反映します.



5 書き込み完了後, メニューの[CPU]-[CPU の停止/停止解除]を選択し CPU の停止状態を解除します.
その後実機のスイッチでリセット+スタートするか, PCwin のメニュー[CPU]-[CPU のリセット/起動]を選択して CPU を起動します.

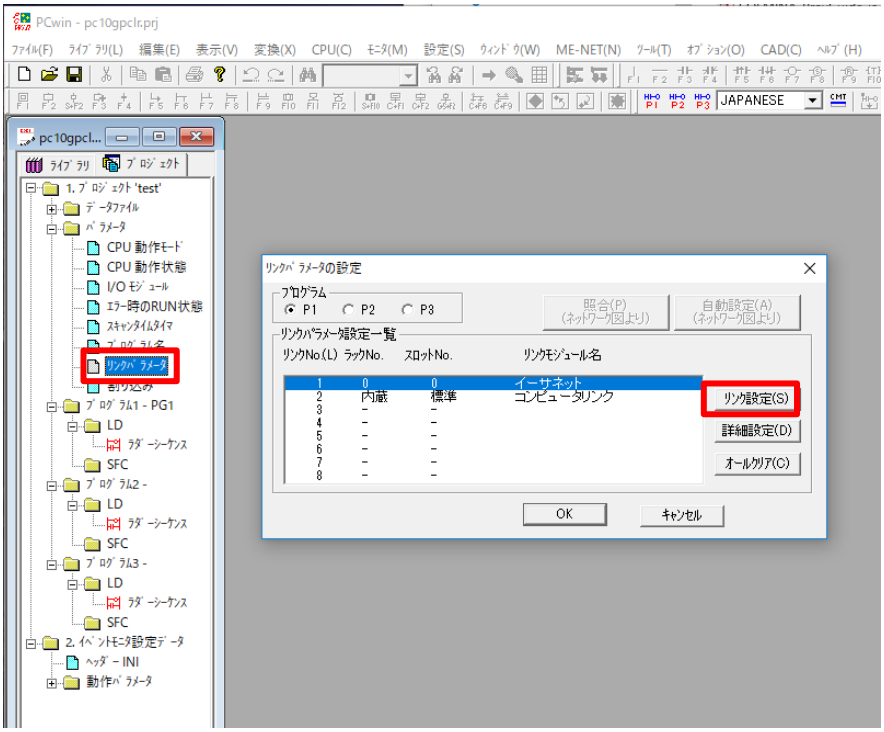
以上でシリアル通信の設定は完了です.

7.3. イーサネット通信の設定

イーサネット通信を使用する場合の設定を説明します．事前に 7.1 を参照し事前準備を行ってください．PC10G の CPU とイーサネットで通信するには CPU 内臓のコンピュータリンク用ポート(L1)を使用する方法と，イーサネットユニットを追加する方法があります[4]．イーサネットユニットを使用して通信場合はイーサネットユニットのマニュアルも併せてご参照ください．

1 PCwin のプロジェクトメニューから[リンクパラメータ]を選択します．

リンクパラメータ設定画面で接続したいプログラム(P1～P3)に対してイーサネットの設定を行います．



2 ラック No とスロット No，およびリンクモジュール名を設定します．

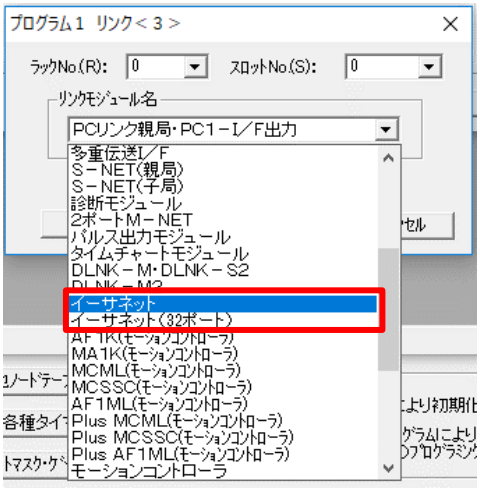
- CPU 内臓のポート(L1)で接続する場合，

ラック No	[内臓]
スロット No	[L1]

- イーサネットユニットを使用して接続する場合，

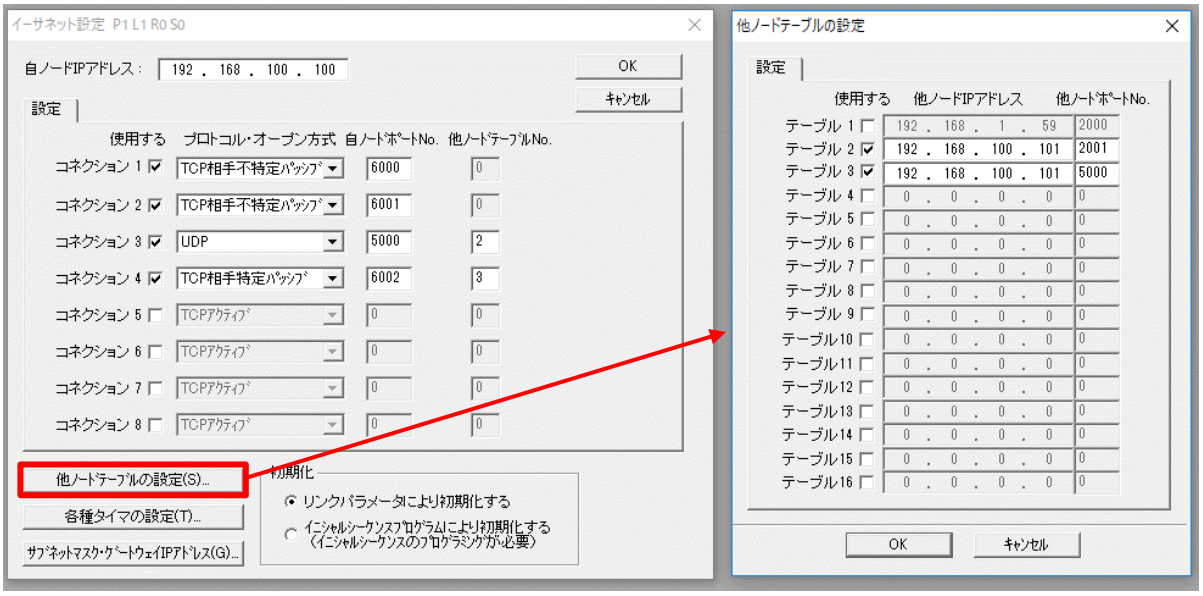
ラック No	イーサネットユニットを設置したラック No
スロット No	イーサネットユニットを設置したスロット No

リンクモジュール名にはどちらの場合も[イーサネット]または[イーサネット(32 ポート)]を設定します．

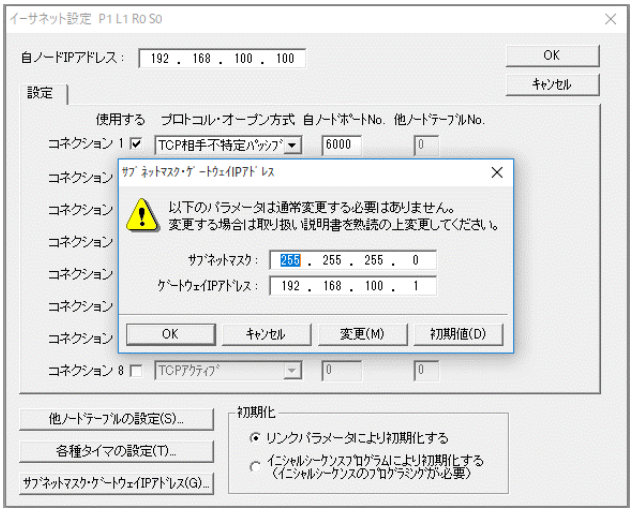


[4] 本プロバイダ開発時には，CPU 内臓ポートの他に Ethernet ユニットである FL/ET-T-V2 を使用して動作確認を行っています．

3 リンクパラメータの設定画面に戻り，[詳細設定]を押します．
イーサネット設定画面で IP アドレスやポートの設定を行います．
プロトコル・オープン方式は通常は[TCP 相手不特定パッシブ]を選択してください．
特定の IP アドレスからのみ接続を許可する場合は，[TCP 相手特定パッシブ]を選択します．
相手特定パッシブを使用する場合は，[他ノードテーブルの設定]を押し，接続元の IP アドレスの設定も行ってください．



4 必要があれば，イーサネット設定画面の[サブネットマスク・ゲートウェイ IP アドレス]を押して
サブネットマスク・ゲートウェイを設定します．



5 以降は 7.2 の手順 3～5 と同様です．

以上でイーサネット通信の設定は完了です．