Digital
メモリリンクプロバイダ

Version 1.1.2

ユーザーズ ガイド

December 18, 2019

備考：

【改版履歴】

バージョン	日付	内容
1.0.0	2019-04-26	初版.
1.1.0	2019-06-20	シリアル通信機能の追加
1.1.1	2019-08-21	OnMessage イベント発生時のメモリリークを修正
1.1.2	2019-12-18	受信処理がタイムアウトになる不具合を修正

【動作確認機種】

機種	バージョン	注意事項
Pro-face (GP4100 シリーズ)	PFXGP4114T2D	
Pro-face (LT4000M シリーズ)	PFXLM4B01DAK	

目次

1. はじめに

本書は，デジタル社のメモリリンクプロトコルを内蔵しているデバイス（以降表示器と呼称）のメモリ情報を取得するプロバイダ（以降プロバイダと呼称）のユーザーズガイドです．また本プロバイダでは，メモリリンクプロトコルのうちイーサネット通信とシリアル通信（バイナリ）に対応しています．

図 1-1 にイーサネット通信での本プロバイダと表示器との全体構成図を示し，図 1-2 にシリアル通信での全体構成図を示します．本プロバイダはイーサネット通信またはシリアル通信により表示器と通信を行います．

■ イーサネット

・ 1:1 接続

・ 1:n 接続

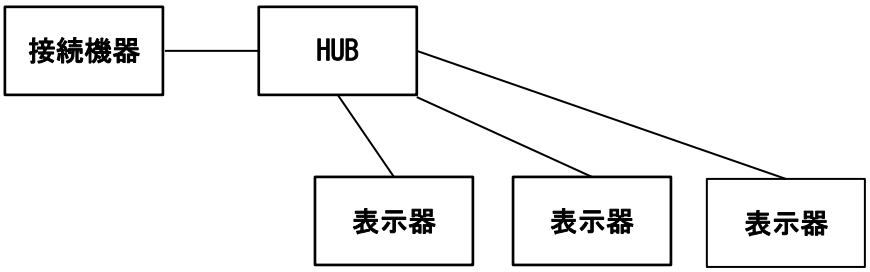
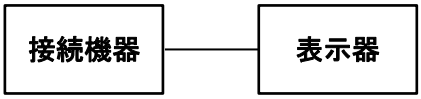


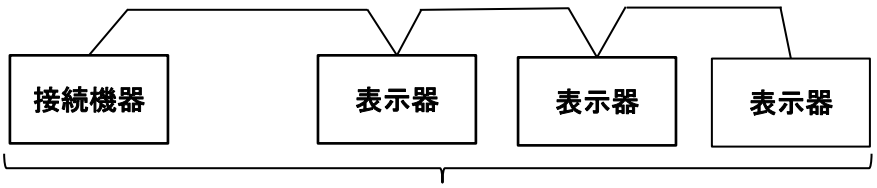
図 1-1 構成図（イーサネット）

■ シリアル

・ 1:1 接続



・ 1:n 接続



最大 32 台

(接続機器を含む)

図 1-2 構成図(シリアル)

また，本プロバイダ及びデバイスそれぞれの対応を図 1-3，図 1-4に示します.
(※一例です．全てを表しているわけではありません．)

イーサネット及びシリアル(1：1接続)はCaoController(3.2.2参照)を使い表示器からデータ取得を行います．シリアル(1:n接続)はCaoExtension(3.2.4参照)を使い指定した号機Noの表示器からデータを取得します.

図 1-3 プロバイダの構成とデバイス情報との対応図

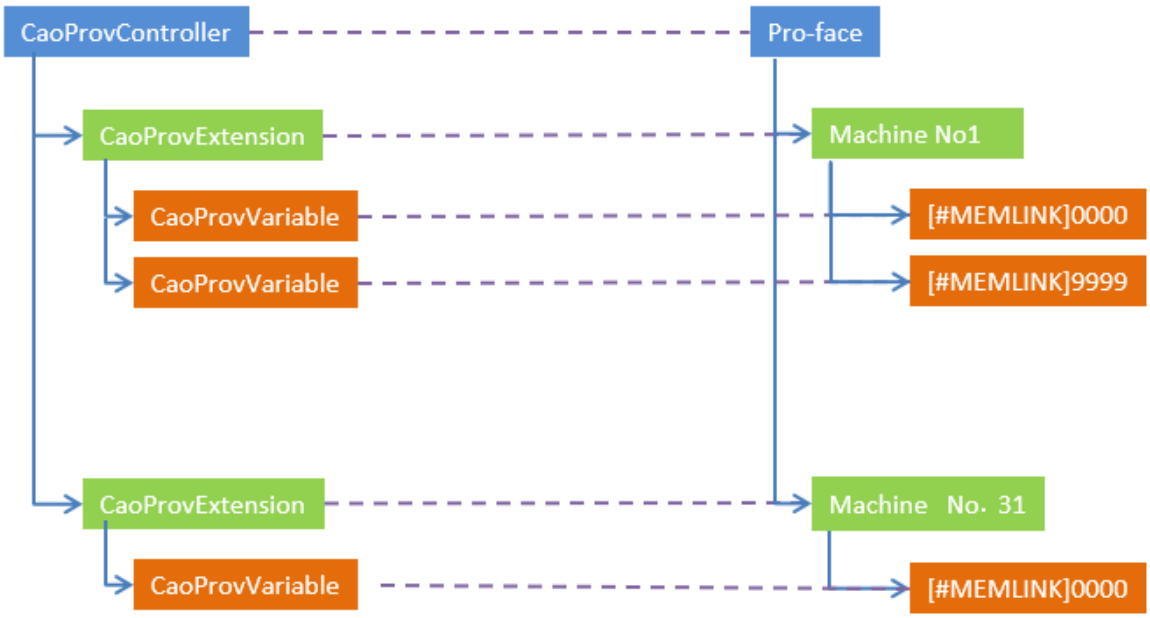


図 1-4 プロバイダの構成とデバイス情報との対応図(号機 No 指定)

1.1. プロバイダの通信仕様

本プロバイダは，通信方式により送受信を行う方法が異なります.

1.1.1. イーサネット通信仕様

イーサネット通信では，表示器から送信されるデマンドポーリングやエラー通知を常に監視するために受信用のスレッド起動し常に表示器からの受信データを確認します.

1.1.1.1. 受信スレッドの処理手順

図 1-5 に受信スレッドの処理手順を示します.

受信データの確認処理にて表示器から送信されたデータを受信します. 受信データ確認処理では，受信データがない場合であっても一定時間待機します.

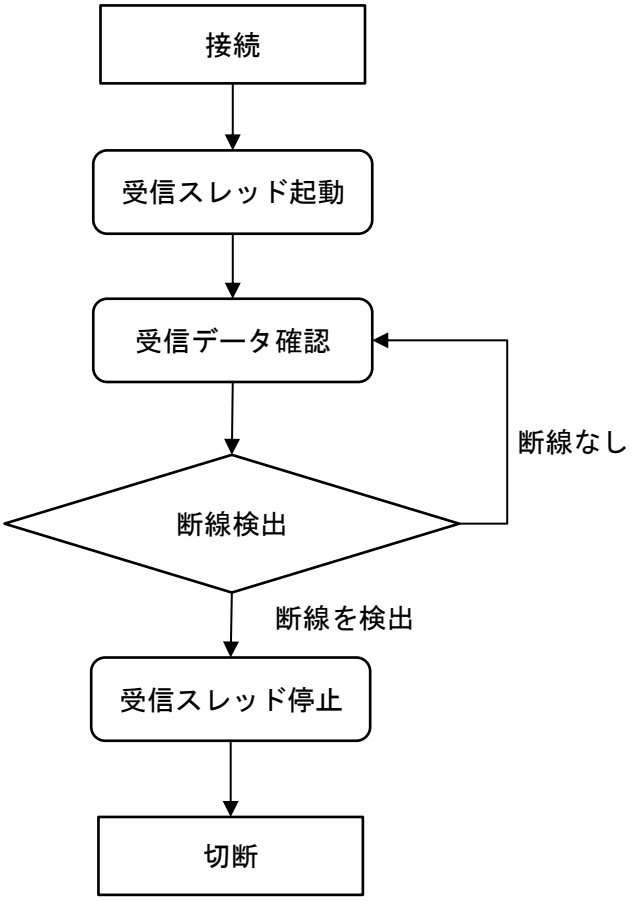


図 1-5 受信スレッド処理手順フローチャート

1.1.1.2. 断線検出

本プロバイダは，接続する際のオプションの設定により断線検出を行います．ある一定時間以内に表示器とデータ通信を行わなかった場合は断線したと判断し，受信スレッドの停止およびソケットの切断処理を行います．断線検出の有効無効および，断線検出するまでの時間の指定は，AddController の DisconnectDetectTime オプションを参照してください．データ通信とは以下の動作を指しています．TCP の Keepalive は含まれていないことに注意してください．

・メモリリンクの Read コマンド
・メモリリンクの Write コマンド
・表示器からのデマンドポーリング受信

また，断線検出がされた場合それ以降通信は行えなくなりますのでご注意ください．再度接続したい場合は，CaoController を一度削除し，再度追加してください．

1.1.1.3. イーサネット通信におけるコマンド送信から受信まで

図 1-6 にイーサネット通信時にプロバイダから取得コマンドを送信しデータを受信までの流れを示します．プロバイダはコマンドを送信後，受信スレッドからのデータ通知が来るまで待機します．この待機時間は AddController の CommandTimeout オプションで指定してください．

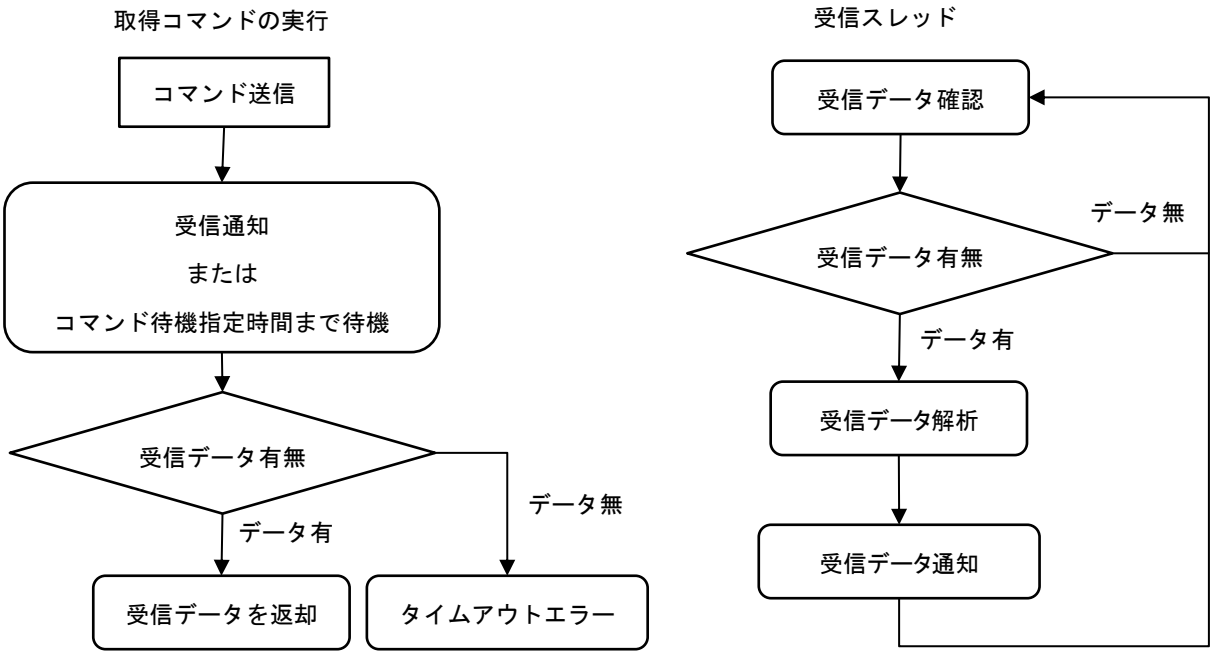


図 1-6 イーサネット通信におけるコマンド送信から受信までのフローチャート

1.1.2. シリアル通信仕様

シリアル通信では，イーサネット通信のようにデマンドポーリングやエラー通知機能がないため受信スレッドを使用せず，コマンドの送信後に受信データの確認とデータの解析を行います．

1.1.2.1. シリアル通信におけるコマンド送信から受信まで

図 1-6 にシリアル通信時にプロバイダから取得コマンドを送信しデータを受信するまでの流れを示します．プロバイダはコマンドを送信後，接続先から受信データの確認をおこない受信データがある場合はそれを返却します．

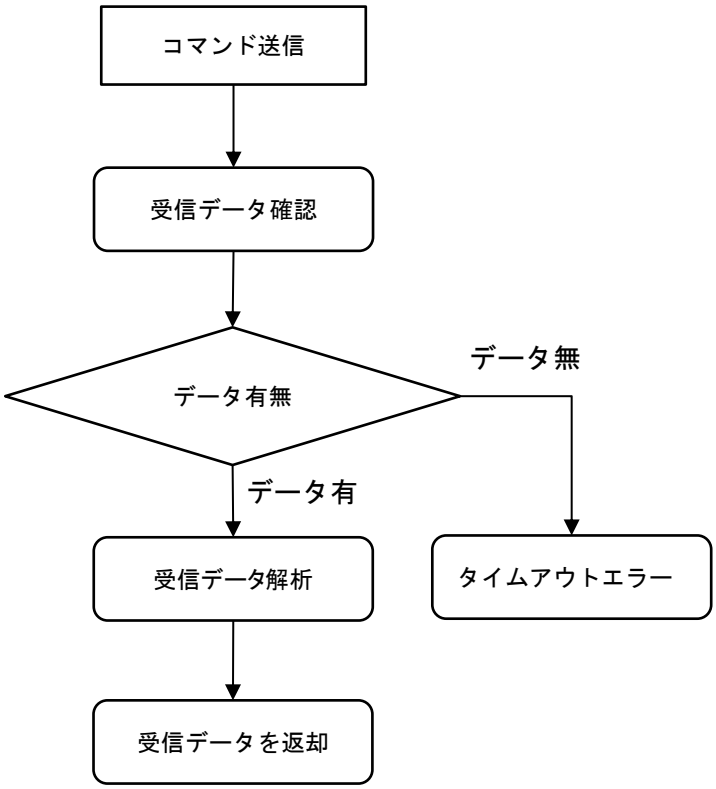


図 1-7 シリアル通信におけるコマンド送信から受信までのフローチャート

1.1.2.2. 表示器設定の注意点

シリアル接続で通信する場合は，表示器の設定を以下にして下さい.

設定内容	設定値
通信プロトコル	拡張モード
通信形式	1:1 バイナリまたは 1:n バイナリ
ETX, サムチェック	有効
ACK	有効
NAK	有効

2. アプリケーション開発のための環境セットアップ

2.1. 表示器とクライアント PC との接続

表示器とクライアント PC との接続については,お使いの Pro-face シリーズの取り扱い説明書を参照してください.

2.2. PC 開発環境のセットアップ

2.2.1. メモリリンクプロバイダの手動インストール

メモリリンクプロバイダを手動でインストールする場合は下記レジストリ登録を行う必要があります. レジストリ登録を行う場合は, 管理者権限でコマンドプロンプトを起動し, regsvr32 コマンドを実行してください.

また, CAO エンジンが動作するには予め, PC 毎に正規の ORiN2 SDK ライセンスが 1 つ登録されていなくてはなりません. ORiN2 SDK ユーザーズガイド内にある「ライセンスの追加と削除」の節を参照してください.

表 2-1 メモリリンクプロバイダ

ファイル名	CaoProvDigitalMemoryLink.dll
ProgID	CaoProv.Digital.MemoryLink
レジストリ登録	regsvr32 CaoProvDigitalMemoryLink.dll
レジストリ登録の抹消	regsvr32 /u CaoProvDigitalMemoryLink.dll

3. コマンドリファレンス

3.1. メソッド/プロパティ一覧

表 3-1 メソッド/プロパティ一覧

カテゴリ	メソッド/プロパティ[1]		機能	参照
CaoWorkspace				
	AddController	M	コントローラに接続	P.12
CaoController				
	VariableNames	P	接続可能な変数名リストの取得	P.14
	AddVariable	M	変数オブジェクトの追加	P.14
	AddExtension	M	拡張ボードオブジェクトの追加	P.15
	Execute	M	拡張コマンドの実行	P.15
	OnMessage	E	メッセージ受信イベント	P.16
CaoVariable				
	Value	P	値の取得/設定	P.16
CaoExtension				
	VariableNames	P	接続可能な変数名リストの取得	P.14
	AddVariable	M	変数オブジェクトの追加	P.14
	Execute	M	拡張コマンドの実行	P.15

3.2. メソッド・プロパティ

3.2.1. CaoWorkspace クラス

3.2.1.1. AddController メソッド

CaoWorkspace に，コントローラオブジェクトを追加します．メモリリンクプロバイダでは，AddController メソッド実行時に渡されたパラメータを参照し，該当する表示機と接続を行います．以下に，AddController メソッドの仕様を示します．

書式

```
AddController
(
        "<コントローラ名>",              // コントローラ名(任意)
        "CaoProv.Digital.MemoryLink",          // プロバイダ名 (固定)
        "<マシン名>",                    // プロバイダ実行マシン名(未使用)
        "<オプション>"                   // オプション文字列(省略可能)
```

[1] M:メソッド，P:プロパティ，E:イベントをそれぞれ示します．

)

オプション

3.2.1.1.1. オプション文字列

以下にオプション文字列に指定するオプションを示します. オプション文字列は下記に示す各オプションをカンマ(,)でつなげた文字列となります.

イーサネット通信時オプション			
オプション	必須	説明	値範囲
CONN=<通信パラメータ>	○	接続先情報を指定します.	ETH
CommandTimeout =<コマンドタイムアウト>	--	コマンド実行後からデータ受信までのタイムアウト時間(ms)を指定します. タイムアウトエラーが頻繁に発生する場合, 値を大きくして設定してください.	1-ULONG_MAX デフォルト値:1000
DisconnectDetectTime=<断線検出時間>	--	断線検出時間(ms)を指定します. 0 を設定することで断線検出が無効となります. 有効にする場合 1 以上の数値を設定してください. 表示器側がデマンドポーリング機能を有する場合, デマンドポーリング周期時間より 1000ms 以上大きな値を指定することを推奨します. 断線検出の仕様については 1.1.1.2 を参照ください.	0- LONG_MAX デフォルト値 : 0

シリアル通信時オプション			
オプション	必須	説明	値範囲
CONN=<通信パラメータ>	○	接続先情報を指定します.	COM
Timeout=<タイムアウト>	--	接続先と 1 回の通信におけるタイムアウトまでの待機時間(ms)を指定します.	0-ULONG_MAX デフォルト値:1000

使用例

```
CaoEngine caoEngin; // Engineオブジェクト
CaoWorkspace caoWorkSpace; // Workspaceオブジェクト
CaoController caoController; // Controllerオブジェクト

caoEngin = new CaoEngine();
caoWorkSpace = caoEngin.AddWorkspace("NewWrks","");
caoController = caoWorkSpace.Controllers.Add("MemoryLink",
"CaoProv.Digital.MemoryLink", "", "Conn=ETH:192.168.0.150,CommandTimeout=1000");
```

3.2.1.1.2. 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("[]")内は省略可能なことを, 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時のデフォルト値をそれぞれ示します.

TCP/IP で接続する場合

"Conn=ETH:<接続先 IP>[:<接続先ポート>[:<ローカル IP>[:<ローカルポート>]]]"

<接続先 IP> : 接続先 IP アドレスを***.***.***.***の形式で指定します.
この項目は必ず指定してください.

[<接続先ポート>] : 接続先ポート番号を指定します（デフォルト値：1024).

[<ローカル IP>] : PC 側の IP アドレスを***.***.***.***の形式で指定します.
デフォルト:未指定

[<ローカルポート>] : デフォルト:未指定

シリアルで接続する場合

"Conn=COM:<COM Port>[:BaudRate>[:<Parity>:<DataBits>:<StopBits>[:Flow]]]"

<COM Port> : COM ポート番号. '1' -COM1, '2' - COM2, ...

<BaudRate> : 通信速度. 4800, 9600, 19200, 38400, 57600, 115200

<Parity> : パリティ. 'N'-NONE, 'E'-EVEN, 'O'-ODD

<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

<Flow> : フロー制御. '0'-None, '1'-Xon/Xoff, '2'-ハードウェア制御
OR をとって指定できます.

3.2.2. CaoController クラス

3.2.2.1. VariableNames プロパティ

接続可能な変数名リストを取得します.本プロパティで取得した変数名は,後述する AddVariable メソッドの第一引数に使用することができます.

使用例

```
// ファイル名リスト取得
object variables;
variables = controller.VariableNames;
```

3.2.2.2. AddVariable メソッド

CaoController に変数オブジェクトを追加します. 変数名には 3.4.1 に示すもののみ使用できます.

以下に, AddVariable の仕様を示します.

書式

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

3.2.2.3. AddExtension メソッド

CaoController に，拡張ボードオブジェクトを追加します．以下に，AddExtension メソッドの仕様を示します．

書式

```
AddExtension
(
        "<拡張ボード名>",       // 拡張ボード名(任意)
        "<オプション>"          // オプション文字列
)
```

オプション

以下にオプション文字列に指定するオプションを示します．オプション文字列は下記に示す各オプションをカンマ(,)でつなげた文字列となります．

オプション	必須	説明	値範囲
MachineNo=<号機番号>	○	接続先の番号を指定する	0-31

3.2.2.4. Execute メソッド

ConController の拡張コマンドを実行します． 以下に，Execute の仕様を示します．

書式

```
Execute
(
        "<拡張コマンド名>",             // 拡張コマンド名
        "<オプション文字列>"            // オプション文字列
)
```

Execute で指定できる拡張コマンド一覧は 3.3 を参照してください．使用例は拡張コマンドの詳細で記述しています．

3.2.2.5. OnMessage イベント

OnMessage イベントは，イーサネット通信接続時のみ受信が可能です.

コントローラのエラー通知や状態の変化を OnMessage イベントとして受け取ることが可能です. 受け取れるイベントについては 3.5 を参照してください.

3.2.3. CaoVariable クラス

3.2.3.1. Value プロパティ

接続した表示器からデータを取得/設定します. 変数名によって動作が異なります. 詳細は，3.4. 変数一覧を参照してください.

3.2.4. CaoExtension クラス

本クラスは，イーサネット接続時は使用することができません.

シリアル（1:n 接続）時に号機番号を指定することで指定した号機番号の表示器と通信を行います.

3.2.4.1. VariableNames プロパティ

接続可能な変数名リストを取得します.本プロパティで取得した変数名は,後述する AddVariable メソッドの第一引数に使用することができます.

使用例

```
// ファイル名リスト取得
object variables;
variables = extension.VariableNames;
```

3.2.4.2. AddVariable メソッド

CaoExtension に変数オブジェクトを追加します. 変数名には 3.4.2 に示すもののみ使用できます.

以下に，AddVariable の仕様を示します.

書式

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

3.2.4.3. Execute メソッド

ConController の拡張コマンドを実行します. 以下に，Execute の仕様を示します.

書式

```
Execute
(
        "<拡張コマンド名>",              // 拡張コマンド名
        "<オプション文字列>"             // オプション文字列
)
```

Execute で指定できる拡張コマンド一覧は 3.3 を参照してください．使用例は拡張コマンドの詳細で記述しています．

3.3. コマンド一覧

コマンドでは，表示器内のメモリ情報を取得設定します．

コマンド	説明	参照
GetMemory	指定したアドレス開始位置から指定した要素数分データを取得します．	P.18
SetMemory	メモリリンクのアドレス領域にデータを設定します．	P.19
GetMemoryRaw	メモリリンクのアドレス領域データを生値で取得します．	P.19
SetMemoryRaw	メモリリンクのアドレス領域にデータを設定します．	P.20
SendRawCommand	設定値をコマンドデータとしてそのまま送信します．	P.21

取得設定コマンドの違いについて

メモリリンクプロトコルでは，2 バイトアドレスにビッグエンディアンで値がやり取りされます．Raw 系（GetMemoryRaw/SetMemoryRaw）コマンドでは，通信仕様通り，ビッグエンディアンの 2 バイトデータのまま扱います．一方(GetMemory/SetMemory)コマンドでは，指定したデータ型になるように型変換（エンディアン変換含む）したデータでやり取りを行います．

1 回の通信でやり取りができるアドレス数について

メモリリンクプロトコルの通信仕様上，1 回の通信で取得設定できるアドレスの最大数は 512 個となります．本プロバイダではアドレス数が 512 個を超える場合，複数回通信を行うためその分通信に時間がかかります．

型指定コマンドにおける VT_I1,VT_UI1 のデータ取得設定仕様について

VT_I1 か VT_UI1 でデータの取得設定を行った場合，アドレスの下位 1 バイトに対して取得設定を行います．設定をする場合は，一度設定するアドレスの情報を取得して取得したアドレスデータの上位 1 バイトをマスクしてから設定を行います．そのため VT_I1 と VT_UI1 では，ほかの型指定と比べると取得して設定する分通信量が増えるのでご注意ください．

3.3.1. GetMemory

表示器のメモリリンクのアドレスからデータを指定したデータ型で取得します．

取得するアドレス開始位置とデータの要素数，データの型を指定します．指定したアドレス開始位置から要素数分データを指定した型で取得します．

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	1	VT_UI2	取得するアドレス開始位置を指定します． 値範囲：0-9999
	2	VT_UI2	取得する要素数を指定します． データ型により指定範囲が違うため 3.4.1.4.1 を参照ください．
	3	VT_BSTR	取得するデータ型を指定します． 指定できる型は 3.4.1.4.1 参照ください．
	4	VT_BOOL	省略可能です． 読取りデータ数 1 の場合に戻り値を配列にするか指定します． TRUE：配列で返却
戻り値	VT_ARRAY | VT_VARIANT(指定したデータ型)		
	n	VT_VARIANT	指定したデータ型で要素数分データを取得

使用例

```
// 引数パラメータの作成(100番地のアドレスから50要素をVT_I2として扱う)
object[] objParam = { 100, 50, "I2"};

//データの取得
object returnValue;
returnValue = this.m_caoController.Execute("GetMemory", objParam);
//取得した object データを unsigned short 形の配列に変換
short[] retVal = (short[])returnValue;
```

3.3.2. SetMemory

表示器のメモリリンクのアドレスにデータを指定したデータ型で設定します.

アドレス開始位置と設定する要素数, データの型, 設定するデータを指定します. 指定したアドレス開始位置から指定したデータ型に合わせて設定をします.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	1	VT_UI2	設定するアドレス開始位置を指定します 値範囲：0-9999
	2	VT_UI2	設定する要素数を指定します. データ型により指定範囲が違うため 3.4.1.4.1 を参照ください
	2	VT_BSTR	設定するデータ型を指定します. 指定できる型は 3.4.1.4.1 参照ください.
	3	VT_ARRAY | VT_VARIANT(指定したデータ型)	
		n	メモリアドレス開始位置から n 番目の領域に書込むデータを指定します.
戻り値	なし		

使用例

```
// 引数パラメータの作成
object[] objParam = new object[4];
objParam[0] = 100; //設定するアドレス開始位置の指定
objParam[1] = 2; //設定する要素数
objParam[2] = "R4"; //データ型VT_R4
objParam[3] = new float[2]{1.2f, 2.3f}; //設定データ

//データの設定
this.m_caoController.Execute("SetMemory", objParam);
```

3.3.3. GetMemoryRaw

表示器のメモリリンクのアドレスからデータを生値で取得します.

アドレス開始位置とデータの要素数を指定します.

指定したアドレス開始位置からデータ要素数分データを取得します. 取得するデータは, 表示器から送られてきた生値を VT_UI2 で取得します.

項目	型説明

引数	VT_ARRAY | VT_VARIANT		
	1	VT_UI2	取得するアドレス開始位置を指定します 値範囲：0-9999
	2	VT_UI2	取得するデータの要素数を指定します. 値範囲:1-10000
	3	VT_BOOL	省略可能です. 取得する要素数が 1 の場合に戻り値を配列にするか指定します. TRUE：配列で返却
戻り値	VT_ARRAY | VT_UI2 または VT_UI2		
	n	VT_UI2	指定したアドレス領域のデータを取得

使用例

```
// 引数パラメータの作成(100番地のアドレスから50要素をVT_UI2として扱う)
object[] objParam = {100, 50};

//データの取得
object returnValue;
returnValue = this.m_caoController.Execute("GetMemoryRaw", objParam);
//取得した object データを unsigned short 形の配列に変換
ushort[] retVal = (ushort[])returnValue;
```

3.3.4. SetMemoryRaw

表示器のメモリリンクのアドレスからデータを生値で設定します.

アドレス開始位置と設定データを指定し，開始位置から指定したデータを生値で設定します.

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	1	VT_UI2	設定するアドレス開始位置を指定します 値範囲：0-9999
	2	VT_UI2	設定要素数 値範囲：1-10000
	3	VT_ARRAY | VT_UI2 または VT_UI2	
		n	メモリアドレス開始位置から設定する要素数分のデータを指定します.

項目	型説明
引数	VT_ARRAY | VT_VARIANT
戻り値	なし

使用例

```
// 引数パラメータの作成
object[] objParam = new object[3];
objParam[0] = 110; //設定するアドレス開始位置の指定
objParam[1] = 5; //設定要素数
objParam[2] = new ushort[5]{1, 2, 3, 4, 5}; //書き込みデータ
//データの設定
this.m_caoController.Execute("SetMemoryRaw", objParam);
```

3.3.5. SendRawCommand

コマンドの生データを送信します.

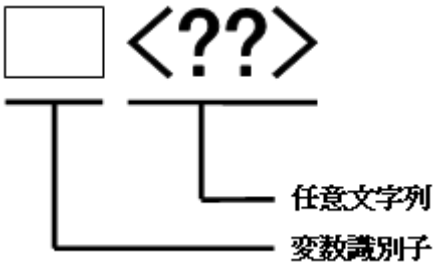
項目	型説明		
引数	VT_ARRAY | VT_UI1		
	n	VT_UI1	コマンドバイト配列
戻り値	VT_ARRAY | VT_UI1		
	n	VT_UI1	戻り値のバイト配列

3.4. 変数一覧

CaoController クラスで使用可能な変数一覧を 3.4.1 に示し, CaoExtension クラスで使用可能な変数一覧を 3.4.2 に示します. なお変数は, 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



3.4.1. CaoController クラス変数

変数名	説明	Value		参照
		get	put	
@MAKER_NAME	メーカー名を取得します.	○	-	P.22
@VERSION	DLL バージョンを取得します.	○	-	P.22
@LAST_DEVICE_ERROR	イーサネット接続時のみ追加可能 最後に通知されたエラー通知メッセージに対応するエラーコードを取得します.	○	-	P.23
MEMLINK<??>	表示器内のメモリ情報を取得/設定します.	○	○	P.23
MEMLINKRAW<??>	表示器内のメモリ情報を生値で取得/設定します	○	○	P.25

3.4.1.1. @MAKER_NAME

メーカー名の取得をします.

データ型

型説明	
VT_BSTR	メーカー名を取得します.

使用例

```
// 変数追加
CaoVariable caoVariable;
caoVariable = this.m_caoController.AddVariable ("@MAKER_NAME");
// 値取得
String strMakerName;
strMakerName = caoVariable.Value;
```

3.4.1.2. @VERSION

DLL のバージョンの取得をします.

データ型

型説明	
VT_BSTR	DLL のバージョンを取得します. *.*.*

使用例

```
// 変数追加
CaoVariable caoVariable;
caoVariable = this.m_caoController.AddVariable("@VERSION");
// 値取得
String strVersion;
strVersion = caoVariable.Value;
```

3.4.1.3. @LAST_DEVICE_ERROR

デバイス側のエラー通知メッセージ（3.5.2.1～3.5.2.3 参照）とは別に最後に通知されたメッセージ内容に対応したエラーコードを取得します. 対応するエラー番号は, デバイス側のエラー通知メッセージ(3.5.2.1～3.5.2.3)を参照してください.

型説明	
	エラーメッセージに対応したエラーコードを取得します.

使用例

```
// 変数追加
CaoVariable caoVariable;
caoVariable = this.m_caoController.AddVariable("@LAST_DEVICE_ERROR");
// エラーコード取得
object LastErrorCode = caoVariable.Value;
```

3.4.1.4. MEMLINK<??>

表示器内のメモリ情報を取得設定します.

型説明	
VTARRAY | VT_UI2	指定したメモリアドレス領域の値を取得設定します.

オプション	必須	説明	値範囲	デフォルト値
StartPos[=<メモリアドレス開始位置>]	--	取得設定を行うメモリアドレスの開始位置を指定します.	0-9999	0
Elem [=<要素数>]	--	Put/Get するデータの要素数を指定します. 指定型により値範囲が異なります. 各指定型の値範囲は 3.4.1.4.1 を参照ください.	1-20000	データ型が BSTR の場合 2 その他のデータ型=1

オプション	必須	説明	値範囲	デフォルト値
VT[=<変数型>]	--	Put/Get するデータ型を指定します.	--	UI2
Array[=True or False]	--	Put/Get する要素数が 1 の場合に配列として扱うか指定します. True=配列として扱う	--	False

3.4.1.4.1. VT オプション

Put/Get するデータ型を指定します.

指定可能なデータ型の一覧とデータ型毎の Elem オプション範囲を以下に示します.

VT	データ型	説明	要素数範囲
I1	VT_I1	8bit の符号あり整数型として読出し/書込みする. 1 アドレス(2 バイトデータ)の下位 1 バイトを 1 要素とする.	1-10000
I2	VT_I2	16bit の符号あり整数型として読出し/書込みする.	1-10000
I4	VT_I4	32bit の符号あり整数型として読出し/書込みする アドレス(2 バイトデータ)2 つ分のデータを 1 要素とする.	1-5000
UI1	VT_UI1	8bit の符号なしデータ型として読出し/書込みする. 1 アドレス(2 バイトデータ)の下位 1 バイトを 1 要素とする.	1-10000
UI2	VT_UI2	16bit の符号なし整数型として読出し/書込みする	1-10000
UI4	VT_UI4	32bit の符号なし整数型として読出し/書込みする アドレス(2 バイトデータ)2 つ分のデータを 1 要素とする	1-5000
R4	VT_R4	32bit の浮動小数点型として読出し/書込みする アドレス(2 バイトデータ)2 つ分のデータを 1 要素とする	1-5000
BSTR	VT_BSTR	1 要素 byte 分の文字コードとして読出し/書込みする. アドレス 1 つのデータが 2byte 分の文字コードとなるため要素範囲は偶数で指定してください.	2-20000

使用例

```
    // 変数追加
    CaoVariable caoVariable;
    // メモリアドレス開始位置を 110,アドレスの取得設定個数を 2 で変数を作成する
    caoVariable = this.m_caoController.AddVariable("MEMLINK_1", "StartPos=110, Elem=2,
VT=I2");
    // エラーコード取得
    object memoryData = caoVariable.Value;
    ushort[] ushMemData = (ushort[])memoryData;
    // 取得したデータに 1 加算する.
```

```
    for(int i= 0; i < ushMemData.Length ; i++ )
    {
        ushMemData[i] += 1;// 1 加算する
    }
    caoVariable.Value = ushMemData; //更新したデータを反映する
```

3.4.1.5. MEMLINK_RAW<??>

表示器内のメモリ情報を生値で取得設定します.

型説明	
VTARRAY | VT_UI2	指定したメモリアドレス領域の値を生値取得設定します.

オプション	必須	説明	値範囲	デフォルト値
StartPos[=<メモリアドレス開始位置>]	--	取得設定を行うメモリアドレスの開始位置を指定します.	0-10000	0
Elem [=<要素数>]	--	Put/Get するデータの要素数を指定します.	1-10000	1
Array[=True or False]	--	Put/Get する要素数が 1 の場合に配列として扱うか指定します. True=配列として扱う.	--	False

使用例

```
    // 変数追加
    CaoVariable caoVariable;
    // メモリアドレス開始位置を 110,アドレスの取得設定個数を 2 で変数を作成する
    caoVariable = this.m_caoController.AddVariable("MEMLINKRAW_1",
"StartPos=110,Elem=2");
    // エラーコード取得
    object memoryData = caoVariable.Value;
    ushort[] ushMemData = (ushort[])memoryData;
    // 取得したデータに 1 加算する.
    for(int i= 0; i < ushMemData.Length ; i++ )
    {
        ushMemData[i] += 1;// 1 加算する
    }
    caoVariable.Value = ushMemData; //更新したデータを反映する
```

3.4.2. CaoExtension クラス変数

以下に CaoExtension で使用できる変数一覧を示す.

変数名	説明	Value	参照

		get	put	
MEMLINK<??>	表示器内のメモリ情報を取得/設定します.	○	○	P.23
MEMLINK_RAW<??>	表示器内のメモリ情報を生値で取得/設定します	○	○	P.25

3.5. イベント一覧

デバイス側からのエラー通知メッセージや指定したデマンドポーリング受付時間を超えても通信が来なかった場合の切断メッセージを通知します.

メッセージ番号	説明	参照
プロバイダエラー通知		
0	デバイス切断通知エラー	P.27
99	受信データエラー	P.27
デバイスエラー通知		
10	生存確認タイムアウトエラー	P.27
11	受信タイムアウトエラー	P.27
12	レスポンスタイムアウトエラー	P.27
13	未確認エラーの受信	P.27

使用例

```
public Main()
{
      m_caoEngin = new CaoEngine();
      m_caoWorkSpace = m_caoEngin.AddWorkspace("NewWrks", "");
      m_caoController = m_caoWorkSpace.Controllers.Add("MemoryLink",
      "CaoProv.Digital.MemoryLink", "", "Conn=ETH:192.168.0.150,CommandTimeout=1000");

      // OnMessageイベントハンドラの登録
      m_caoController.OnMessage += new
_ICaoControllerEvents_OnMessageEventHandler(OnMessage);
}

//メッセージ受信処理
private void OnMessage(CaoMessage pICaoMsg)
{
      try
      {
            // 受信メッセージのIDを表示
```

```
            MessageBox.Show(pICaoMsg.Number.ToString());
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
}
```

3.5.1. プロバイダ側メッセージ通知

プロバイダ側から通知するメッセージとなります.

3.5.1.1. デバイス切断通知メッセージ

最後に通信した時から DisconnectDetectTimeout オプションで指定した時間通信が行われなかった際に通知されます. 通知後は, 表示器との通信ができない状態になりますので再度接続しなおしてください.

3.5.1.2. 想定外データの受信エラーメッセージ

受信データの内容が想定外のものが受信されました. Dll のバージョンとメモリリンクのバージョンをご確認ください.

3.5.2. デバイスエラー通知

接続先のデバイス（表示器）が通知してきたエラー通知メッセージを通知します.

3.5.2.1. 相手先生存監視タイムアウトエラーメッセージ

デバイス側から通知されるエラーメッセージです.

エラーメッセージの内容は,「Error Alive time out」になります.

このエラーが通知された場合, 変数@LAST_DEVICE_ERROR の値を”0x80100001”で更新します.

3.5.2.2. プロトコル間タイムアウトエラーメッセージ

デバイス側から通知されるエラーメッセージです.

エラーメッセージの内容は,「Error Response time out」

このエラーが通知された場合, 変数@LAST_DEVICE_ERROR の値を”0x80100002”で更新します.

3.5.2.3. キャラクター間タイムアウトエラーフレームメッセージ

デバイス側から通知されるエラーメッセージです.

エラーメッセージの内容は,「Error Receive time out」になります.

このエラーが通知された場合, 変数@LAST_DEVICE_ERROR の値を”0x80100003”で更新します.

3.5.2.4. 例外エラーのメッセージの受信

表示器から想定されていないメッセージを受信となります.

このエラーメッセージの内容に関しては, メモリリンクプロトコルのエラーメッセージ通知の仕様を参照してください.

例外エラーメッセージを受信した場合，変数@LAST_DEVICE_ERROR の値を”0x80100004”で更新します.

4. サンプルプログラミング

メモリリンクプロバイダでは，以下の手順でクライアント PC と表示器を接続することができます.

- CaoEngine の作成
- CaoWorkspace の作成
- CaoController の作成

表示器に接続した後は，CaoController の Execute メソッドを使用する，もしくは，CaoVariabe オブジェクトを生成することで，表示器の情報にアクセスすることができます.

4.1. メモリデータを取得して値を更新して書き込みするサンプルプログラミング

ここでは例として PLC のグローバル OM のデータレジスタの値を読み書きするサンプルプログラムを示します. 表 4-1 にサンプルプログラムの要件を記述しています.

表 4-1 サンプルプログラムの要件

要件	説明
接続先	TCP/IP で接続する
	接続先 IP アドレスは 192.168.0.150
	接続先ポート番号は 1024
処理内容	Proface[#MEMLINK110]から値を読込む.
	Proface[#MEMLINK110]から取得した値+1 を書き込む.

以降の節から具体的なコードを示します.

4.1.1. サンプルプログラム

以下にサンプルプログラムの全体像を示します.

Sample MemoryLinkGetSet.cs

```
' オブジェクト
private CaoEngine m_caoEngin;
private CaoWorkspace m_caoWorkSpace;
private CaoController m_caoController;

private List<short> m_memoryListData = new List<short>();

public MemoryLinkGetSet()
{
    //メモリ開始位置と個数を指定
    int iStartPos = 110;
    int iElem = 2;
    // 接続
    Connect();
```

```
    GetMemoryData(iStartPos,iElem);
    SetMemoryData(iStartPos,iElem);
    // 切断
    Disconnect();
}

// 接続メソッド
private void Connect()
{
    this.m_caoEngin = new CaoEngine();
    this.m_caoWorkSpace = this.m_caoEngin.AddWorkspace("NewWrks","");
    this.m_caoController = m_caoWorkSpace.Controllers.Add("MemoryLink",
                           "CaoProv.Digital.MemoryLink",
                            "",
                           "Conn=ETH:192.168.0.150");
}

// 切断メソッド
private void Disconnect()
{
    if(this.m_caoController != null)
    {
        // CaoWorkspace から CaoController を削除
        this.m_caoWorkSpace.Controllers.Remove(this.m_caoController.Name);
        // CaoController の消去
        System.Runtime.InteropServices.Marshal.ReleaseComObject(this.m_caoController);
        this.m_caoController = null;
    }
    // CaoEngine から CaoWorkspace を削除
    this.m_caoEngin.Workspaces.Remove(this.m_caoWorkSpace.Name);
    System.Runtime.InteropServices.Marshal.ReleaseComObject(this.m_caoWorkSpace);
    // CaoWorkspace を消去
    this.m_caoWorkSpace = null;
    // CaoEngine を削除
    System.Runtime.InteropServices.Marshal.ReleaseComObject(this.m_caoEngin);
    this.m_caoEngin = null;

}
// メモリデータ取得メソッド
private void GetMemoryData(int iStartPos,int iElem)
{
    // パラメータの設定
    object[] objParam = { iStartPos, iElem , "I2"};
    // データの取得
    object returnValue;
    returnValue = this.m_caoController.Execute("GetMemory", objParam);
```

```
    // 取得データを ushort の配列に変換
    short[] retVals = (short[])returnValue;

    this.m_memoryListData.Clear();
    // 取得したデータをリストに格納
    foreach(short memData in retVals)
    {
        this.m_memoryListData.Add(memData);
    }
}
// メモリデータ設定メソッド
private void SetMemoryData(int iStartPos,int iElem)
{
    //書き込み用パラメータの設定
    short[] writeData = new short[iElem];
    object[] objParam = new object[4];

    for (int i = 0; i < iElem; i++)
    {
        // データに 1 加算して設定データ用配列に格納する
        this.m_memoryListData[i]++;
        writeData[i] = this.m_memoryListData[i];
    }
    objParam[0] = iStartPos;
    objParam[1] = iElem;
    objParam[2] = "I2";
    objParam[3] = writeData;
    // 書き込みを実行する.
    this.m_caoController.Execute("SetMemory", objParam);
}
```

4.1.1.1. 接続

表示器と接続するためには，以下の手順を取ります.

(1) オブジェクトを保持するための変数を用意します. コントローラ接続に必要なオブジェクトは，CaoEngineオブジェクトとCaoWorkspaceオブジェクトとCaoControllerオブジェクトです. CaoWorkpaceオブジェクトは，CaoControllerオブジェクトをCaoWorkspacesから取得する場合

には変数を用意する必要はありません. また変数にアクセスするためのCaoVariableオブジェクトも必要になります. 以下にVB6でのコード例を示します.

```
private CaoEngine m_caoEngin;                    // CaoEngineオブジェクト用の変数
private CaoWorkspace m_workSpace;    // CaoWorkspaceオブジェクト用の変数
private CaoController m_caoController;       // CaoControllerオブジェクト用の変数
```

(2) CaoEngineオブジェクトを生成します. CaoEngineオブジェクトはNewキーワードを使って生成します.

```
' CaoEngine オブジェクトの生成
this.m_caoEngin = new CaoEngine();
```

(3) CaoWorkspaceオブジェクトを取得もしくは生成します. CaoEngineオブジェクトを生成すると, デフォルトでCaoWorkspacesオブジェクトとCaoWorkspaceオブジェクトを1つずつ生成しています. 以下にCaoWorkspaceオブジェクトを新しく生成するコード例とデフォルトのCaoWorkspaceを示します.

```
' CaoWorkspace オブジェクトの生成
this.m_caoWorkSpace = this.m_caoEngin.AddWorkspace("NewWrks","");
```

(4) CaoControllerオブジェクトを生成します. CaoControllerオブジェクトを生成するには, 使用するプロバイダ名と使用するためのパラメータを設定します. メモリリンクプロバイダでは, IPアドレスをオプションで指定します. 以下にコード例を示します.

```
// CaoController オブジェクトの生成
this.m_caoController = m_caoWorkSpace.Controllers.Add("MemoryLink",
                       "CaoProv.Digital.MemoryLink",
                        "",
                         "Conn=ETH:192.168.0.150");
```

4.1.1.2. メモリデータの取得

(1) Execute コマンドに渡す引数のパラメータを作成します.

```
// パラメータの設定
object[] objParam = { iStartpos, iElem, "I2"};
```

(2) データの取得を行います.

```
// データの取得
object returnValue;
returnValue = this.m_caoController.Execute("GetMemory", objParam);
```

(3) 取得したデータを short 型の配列に変換してリストに設定します.

```
// 取得データを short の配列に変換
short[] retVals = (short[])returnValue;

this.m_memoryListData.Clear();
// 取得したデータをリストに格納
foreach(short memData in retVals)
{
      this.m_memoryListData.Add(memData);
}
```

4.1.1.3. メモリデータの設定

(1) 書込み用パラメータの設定.書き込みデータを設定する際は，1 加算してから設定する.

```
//書き込み用パラメータの設定
short[] writeData = new short[iElem];
object[] objParam = new object[3];
// 取得したデータに 1 加算してから設定する
for (int i = 0; i < iElem; i++)
{
       // データに 1 加算して設定データ用配列に格納する
       writeData[i] = this.m_memoryList[i] + 1;
}
objParam[0] = iStartpos; // アドレス開始位置
objParam[1] = "I2";
objParam[2] = writeData; // 書込みデータ
```

(2) 設定したパラメータの内容で書き込みを実行する.

```
// 書き込みを実行する
this.m_caoController.Execute("SetMemory", objParam);
```

4.1.1.4. 切断

コントローラと切断する場合には，生成したオブジェクトを消去すると共に，オブジェクトを管理するコレクションクラスから消去するオブジェクトを削除します．以下にコード例を示します．

```
' CaoWorkspace から CaoController を削除
Call workspace.Controllers.Remove(controller.Index)
' CaoController の消去
Set controller = Nothing
```

```
    ' CaoEngine から CaoWorkspace を削除
    Call engine.Workspaces.Remove(workspace.Index)
    ' CaoWorkspace の消去
    Set workspace = Nothing
    ' CaoEngine の消去
    Set engine = Nothing
```

5. メモリリンクプロバイダエラーコード

本プロバイダには，0x8011****でマスクした以下の独自エラーコードが存在します．（表 5-1 独自エラーコード表参照）

ORiN2 の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．

表 5-1 独自エラーコード表

エラー番号	説明
0x80110002	接続パラメータ指定エラー CommandTimeoutはReceiveTimeoutよりも大きい値を指定してください．
0x80110003	想定外のデータを受信エラー 通信環境を見直してください．
0x80110004	型指定エラー データの指定型を見直してください． 指定できる型は，3.4.1.4.1を参照ください．
0x80110005	設定要素数エラー 設定データの要素数が指定している要素数と一致していません． 指定できる要素範囲は3.4.1.4.1を参照ください．
0x80110006	型指定がVT_BSTRのときに指定要素数として奇数が設定されました． 型指定がVT_BSTRのときは要素数が偶数になるようにしてください．
0x80110007	設定が1:1接続の表示器に対してCaoExtensionでコマンドを送信しています．設定コマンドを送信している場合，設定されている可能性があるので注意してください．

エラー通知によるデバイス側からのエラーメッセージに対応するエラーコードに関しては，以下の表 5-2 エラーコード表を参照してください．

表 5-2 エラーコード表

エラー番号	説明
0x80100001	相手先生存監視タイムアウトエラーメッセージ（3.5.2.1参照）
0x80100002	プロトコル間タイムアウトエラーメッセージ(3.5.2.2参照)
0x80100003	デバイス側のエラー通知アウトエラーメッセージ(3.5.2.3参照)
0x80100004	その他のエラーの受信メッセージ（3.5.2.4参照）

また，本プロバイダは，デバイスからのエラーコードを「0x801010**」でマスクして返します．

デバイス側からのエラーコード詳細は，メモリリンクコマンドエラーコード一覧を参照してください．

付録A. メモリリンクコマンドエラーコード一覧

エラー番号	説明
06	チェックサムコードが一致しない.
10	未定義コマンドを受信しました.
12	指定データ数と受信データ数が一致しません.
15	指定した表示属性がフォーマット外のデータです.
16	指定した文字サイズがフォーマット外のデータです.
17	指定した座標データがフォーマット外のデータです.
18	指定した線種コードがフォーマット外のデータです.
19	指定したタイリングパターンがフォーマット外のデータです.
1A	指定した半径が表示域を超えています.
1B	指定した開始角度 / 終了角度がフォーマット外のデータです.
1C	指定した文字種コードがフォーマット外のデータです.
1D	指定した回転コードがフォーマット外のデータです.
1E	指定した方向コードがフォーマット外のデータです.
1F	指定した強調コードがフォーマット外のデータです.
20	指定した矢印パターンがフォーマット外のデータです.
21	指定した矢印方向コードがフォーマット外のデータです.
22	指定した面取り方法がフォーマット外のデータです.
23	指定したセンターリングコードがフォーマット外のデータです.
24	指定した属性コードがフォーマット外のデータです.
25	コントラスト調整できない機種にコントラスト調整コマンドを送信しまし た.
26	指定したコントラスト設定値が範囲外です.
27	輝度調整できない機種に輝度調整コマンドを送信しました.
28	指定した輝度設定値が範囲外です.
29	流れメッセージが設定されていません.
2A	指定したフォントコードがフォーマット外です.
2B	指定したプライオリティコードがフォーマット外です.
FA	指定したシステムエリアのアドレスが範囲外です.
FB	指定したシステムエリアの範囲を超えて書き込み / 読み出しを行いました.
FC	受信したデータフォーマットに異常があります.

エラー番号	説明
FF	表示器がデータ送信できない状態が 10 秒以上続いた.

付録B. 通信プロトコルコマンド対応表

CaoController::Execute

コマンド	通信コマンド
SetMemory	Write
GetMemory	Read
SetMemoryRaw	Write
GetMemoryRaw	Read