Ethernet 対応 LAN ゲートウェイ

Active X コンポーネント

説明書

Ver-1.2

ＮＫＥ株式会社

EACTIVEX-800C

はじめに

正しくお使いいただくためにこの説明書をよくお読みください。

この説明書は、Ethernet 対応 LAN ゲートウェイ(UNG-LN1N,UNG-LN2N)の通信制御を行う Active X コンポーネントについてのものです。

ご注意

- 本書の内容に関しまして、将来予告なしに変更することがあります。
- 本書の一部または全部を無断で転載することは禁止されています。
- 本書の内容に関しまして、誤りや記載もれなどお気付きの点がございましたら、お手数ですが弊社までお知らせください。

お願い

- 本製品をご使用になる前に、必ず本書をお読みください。
- 本製品を、本書に記載のない条件や、環境で使用しないでください。
- 本製品を、不正に改造、修理しないでください。
- 本製品を、航空、宇宙、医療、原子力、運輸、交通、各種安全装置など、人命、財産に大きな影響が予測される用途には使用しないでください。
- 下記の場合は、保証の対象外とさせていただきます。
  1. 火災、地震、第三者による行為、その他事故などにより、本製品の不具合が発生した場合
  2. お客様の故意、もしくは過失、誤用、乱用、その他異常な条件下での使用により、本製品の不具合が発生した場合
  3. お客様が本製品に手を加えて改造、修理した場合
  4. ユーザシステムの仕様や、使用方法に起因して損害などが発生した場合

変更履歴

バージョン	日　付	変更内容
Ver-1.0 (EACTIVEX-800A)	2004. 6. 1	初版
Ver-1.1 (EACTIVEX-800B)	2004.11.25	改訂１版 インターフェイス一覧を改訂。 インターフェイス説明に記述例を追加。 OpenComm, bValue, Value 解説を改訂。
Ver-1.1 (EACTIVEX-800C)	2015.7.14	改訂２版 対応機種に SDD-LN2N を追加

目次

1. 概要

本 Active X（NKE ActiveX Uni-IO Access 1.0 Type Library）は、Ethernet 対応 LAN ゲートウェイの、Ethernet を介した遠隔通信処理を受け持つコンポーネントです。
PC と Ethernet 対応 LAN ゲートウェイを使ったシステムを構築される場合、本 ActiveX を利用していただくと簡単にアプリケーションを作成することができます。

2. 動作環境

本 Active X と、Active X を使用したアプリケーション・ソフトウェアの動作環境は、下記の通りです。

Ethernet 対応 LAN ゲートウェイ

SDD-LN2N

UNG-LN2N

PC

１つ以上の LAN ポートを装備した、下記の対応 OS を実行するもの

対応 OS

Microsoft Windows 98＋サービスパック 1

Microsoft Windows 98 SE

Microsoft Windows Me

Microsoft Windows 2000 Professional / Server

Microsoft Windows XP Professional / Home Edition

対応言語

Microsoft Visual Basic Ver 6.0 以降

Microsoft Visual C++ Ver 6.0 以降

Microsoft Office 2000

Microsoft Office XP

Microsoft Office 2003

3. Active X の導入

本 Active X を使用するためには、PC に Active X を導入する作業が必要になります。作業は、下記の手順で行います。

① Active X ファイルのコピー

② Active X のレジストリ登録

なお、導入は、必ず上記の手順で行ってください。上記の手順以外で導入した場合、Active X は正しく動作しないことがあります。

3.1 Active X ファイルのコピー

本 Active X を使用するため、PC に Active X ファイルをコピーします。メディアに収録した下記のファイルを、所定のフォルダにコピーしてください。

< Active X ファイル名>

NkeAxIo.exe

<コピー先フォルダ>

C :¥ Windows ¥ system (Windows98/SE/Me の場合)

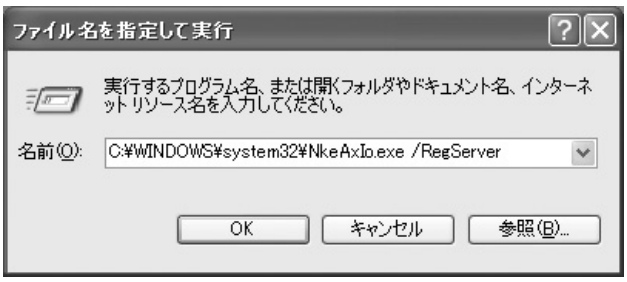
C :¥ WINNT ¥ system32 (Windows2000 の場合)

C :¥ Windows ¥ system32 (WindowsXP の場合)

3.2 Active X のレジストリ登録

Active X ファイルのコピー後、アプリケーションから使用できるようにするため、OS に Active X をレジストリ登録します。レジストリ登録は下記の方法で行ってください。

<登録方法>



上記の画面のように、ファイル名「(コピー先フォルダ)¥ NkeAxIo.exe /RegServer」を指定して、[OK] ボタンをクリックしてください。これで、登録は完了です。

4. Active X の削除

本 Active X を導入した PC から Active X を削除するには、下記の手順で行います。

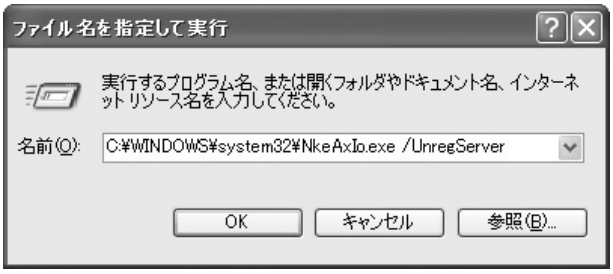
① Active X のレジストリ削除

② 古い Active X ファイルの削除

4.1 Active X のレジストリ削除

OS に登録している Active X をレジストリ削除します。レジストリ削除は、下記の方法で行ってください。

<削除方法>



上記の画面のように、ファイル名「(コピー先フォルダ)￥NkeAxIo.exe /UnregServer」を指定して、[OK]ボタンをクリックしてください。これで、削除は完了です。

4.2 古い Active X ファイルの削除

PC にコピーした古い Active X ファイルを削除します。PC 内に保存した下記のファイルを、保存先フォルダから削除してください。

< Active X ファイル名>

NkeAxIo.exe

<保存先フォルダ>

C :¥ Windows ¥ system (Windows98/SE/Me の場合)

C :¥ WINNT ¥ system32 (Windows2000 の場合)

C :¥ Windows ¥ system32 (WindowsXP の場合)

5. アプリケーション開発の準備

レジストリ登録後は、対応するプログラミング言語で Active X **を使用したアプリケーションを開発することができます。**

5.1 対応するプログラミング言語

本 Active X **は、下記の開発ツールで使用可能なプログラミング言語に対応しています。**

① Visual Basic

1: Microsoft Visual Basic 6.0 **以降**

2: Microsoft Office 2000

3: Microsoft Office XP

4: Microsoft Office 2003

② C++ **言語**

1: Microsoft Visual C++ 6.0 **以降**

5.2 インターフェイス

本 Active X **では、下記のインターフェイスを使用します。**

① INkeIoServer **インターフェイス**

プロパティ

名称	機能
Version	**本** Active X **のバージョン文字列の取得**

メソッド

名称	機能
OpenComm	NkeIoComm **インターフェイスの取得**

② INkeIoComm **インターフェイス**

プロパティ

名称	機能
UniState	UNILINE **状態を取得**
bValue	**１点単位で** I/O **状態を設定／取得**
Value	**指定点数の** I/O **状態を設定／取得**

メソッド

名称	機能
UpdateReadCache	Ethernet **対応** LAN **ゲートウェイから内部バッファに入力**
UpdateWriteCache	**内部バッファから** Ethernet **対応** LAN **ゲートウェイに出力**
GetAbnormalIdCount	**異常** ID **ターミナルの台数を取得**
DumpAbnormalId	**異常** ID **ターミナルの検出状態を取得**

5.3 プログラミングの留意点

本 Active X を使用したアプリケーションは、下記の点に留意して作成してください。

5.3-1 プログラムの流れ

本 Active X を使用したプログラムの流れは、下記の通りです。

① プログラム開始
② INkeIoServer インターフェイスの生成
③ INkeIoComm インターフェイスの生成
④ INkeIoComm インターフェイスから Ethernet 対応 LAN ゲートウェイと通信
⑤ INkeIoComm インターフェイスの破棄
⑥ INkeIoServer インターフェイスの破棄
⑦ プログラム終了

5.3-2 インターフェイスの使用法

本 Active X は、下記の２つのインターフェイスを使用して Ethernet 対応 LAN ゲートウェイと通信します。各インターフェイスの使用法は、下記の通りです。

① INkeIoServer インターフェイス

Ethernet 対応 LAN ゲートウェイの通信管理を行うオブジェクトです。管理用のオブジェクトであるため、実際に Ethernet 対応 LAN ゲートウェイと通信する機能は、ありません。

本 Active X は、プログラムの最初に INkeIoServer インターフェイスを生成、最後に破棄して使用します。なお、プログラム１本につき生成する INkeIoServer インターフェイスの数は、必ず１つにしてください。

② INkeIoComm インターフェイス

Ethernet 対応 LAN ゲートウェイの通信制御を行う通信オブジェクトです。実際に Ethernet 対応 LAN ゲートウェイと通信する機能があります。

INkeIoComm インターフェイスは、内部で INkeIoServer インターフェイスの機能を使用するので、必ず INkeIoServer インターフェイスから生成してください。単独で生成した INkeIoComm インターフェイスは、使用できません。

INkeIoComm インターフェイスは、複数を同時に生成できます。なお、同じ Ethernet 対応 LAN ゲートウェイに複数の INkeIoComm インターフェイスを生成したときは、すべてのインターフェイスに同じ Ethernet 対応 LAN ゲートウェイの状態が反映されます。

6. Visual Basic および C++ 言語によるアプリケーション開発

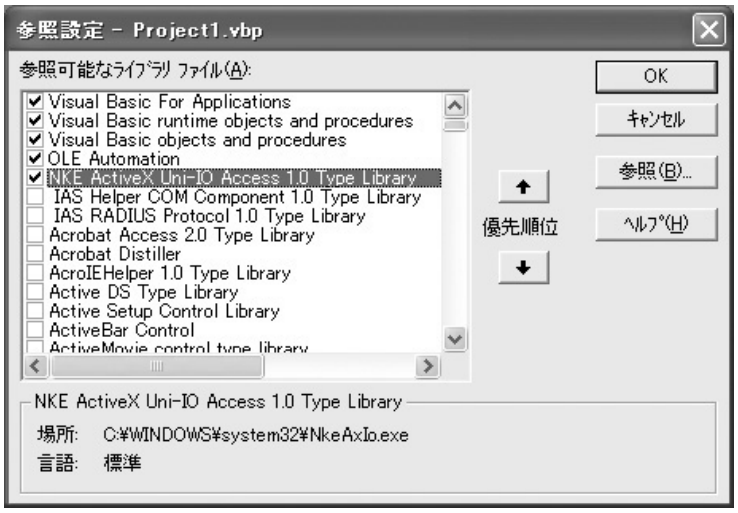
本 Active X で Visual Basic およびＣ＋＋言語によるアプリケーションを開発するときは、下記のようにします。

なお、Visual Basic およびＣ＋＋言語から Active X を使用する方法については、各種開発ツールに付属の技術資料などをご覧ください。

6.1 Visual Basic での開発

Visual Basic 6.0、Office 2000、Office XP、および Office 2003 などの Visual Basic で、本 Active X を使用したプログラム開発をする前には、必ず開発ツールから本 Active X を参照する設定を行ってください。設定手順は、下記の通りです。

<設定方法>



Visual Basic 6.0 のときは、メニューから［プロジェクト］→［参照設定］を選択すると、上記の参照設定ダイアログを表示します。Office 2000、Office XP、および Office 2003 のときは、Visual Basic Editor を起動して、Visual Basic Editor のメニューから［ツール］→［参照設定］を選択してください。画面を開いた後、画面の中の「参照可能なライブラリファイル」から「NKE ActiveX Uni-IO Access 1.0 Type Library」を選び、チェックを入れて［OK］ボタンを押してください。

6.1-1 エラー検出

Active X のプロパティ、メソッドで発生したエラーは On Error ～ GoTo などのエラー処理構文で検出してください。

6.2 Ｃ＋＋言語での開発

Visual C++ 6.0 などのＣ＋＋言語で、本 Active X を使用したプログラム開発をするには、『コンパイラＣＯＭサポート』機能を使用します。コンパイラＣＯＭサポート機能を使用するには、下記のようにします。

6.2-1 Ｃ＋＋言語ソースファイルの変更

コンパイラＣＯＭサポート機能を使用するために、Ｃ＋＋言語ソースファイルに、Active X を使用するための記述を追加します。下記の文を、Ｃ＋＋言語ソースファイルに追記してください。

<記述内容>

#import "NkeAxIo.exe"	…	1:ＰＣに導入した Active X のファイル名を、フルパス表記で記述してください。
using namespace NKEAXIOLib;	…	2:Active X のファイル名を指定した後に記述します。

上記の記述は、必ず 1: ～ 2: の順で２行ともソースコードに記載してください。記述しないとコンパイル時にエラーが発生します。また、上記の記述は、ソースファイル内で実際に Active X を使用する処理を記述する前の行に記述してください。上記の順で記述を行うこと以外、記載位置などに制限はありません。

1:は、コンパイラが、本 Active X からメソッド、プロパティ定義を自動作成するための記述です。

6.2-2 インターフェイスの使用

コンパイラＣＯＭサポート機能を使用すると、Ｃ＋＋言語から Active X のインターフェイスをクラスとして使用できるようになります。また、メソッド、プロパティは、インターフェイスのクラスに属するメンバとして、Visual Basic と同じ仕様の名前、書式で使用できます。

<記述例>

```
INkeIoServerPtr ptIoServer(__uuidof(NkeIoServer));// INkeIoServer 作成
IUnknown * ptGet = NULL;                          // OpenComm 受け取り準備
ptIoServer->OpenComm(Adr, &IoMap, Tout, &ptGet);// OpenComm 実行
INkeIoCommPtr ptIoComm;                           // INkeIoComm 作成
ptIoComm = ptGet;                                 // OpenComm を変換
strVersion = ptIoComm->Version;                   //  バージョン文字取得
```

6.2-3 エラー検出

Active X のプロパティ、メソッドで発生したエラーは、例外処理(try ～ catch 文)で検出してください。

6.3 INkeIoServer インターフェイス

INkeIoServer インターフェイスは、Visual Basic では「NkeIoServer」形、C＋＋言語では「INkeIoServerPtr」形として使用します。

INkeIoServer を使用するための宣言する書式例は、下記のとおりです。

<記述例>

（Visual Basic）

```
Private IoServer As NkeIoServer
```

（C＋＋言語）

```
INkeIoServerPtr ptIoServer(__uuidof(NkeIoServer));
```

6.3-1 プロパティ

Version

■ 機能

本 Acvite X のバージョン文字列を取得します。

■ 記述例

（Visual Basic）

```
Dim AxVer As String
AxVer = IoServer.Version
```

（C＋＋言語）

```
_bstr_t AxVer = ptIoServer->Version
```

■ 書式

値 = オブジェクト. Version　　：（Visual Basic）

値 = オブジェクト->Version;　　：（C＋＋言語）

値　　：戻り値を格納するデータです。

オブジェクト　：INkeIoServer インターフェイスを指すオブジェクト式です。

■ 引数

なし

■ 戻り値

データ型：　String　　：（Visual Basic）

　　_bstr_t　　：（C＋＋言語）

■ 解説

本 Active X のバージョン番号を文字列で返します。

6.3-2 **メソッド**

OpenComm

■ **機能**

INkeIoComm **インターフェイスを取得します。**

■ **記述例**

（Visual Basic**）**

```
Private IoComm As NkeIoComm
Dim bstrParam As String
Dim IoMap As String
Dim Timeout As Long
bstrParam = “192.168.251.1”
IoMap = “0111000000000000”
Timeout = 500
Call IoServer.OpenComm(bstrParam, IoMap, Timeout, IoComm)
```

（C＋＋言語）

```
_bstr_t bstrParam = “192.168.251.1”;
_bstr_t IoMap = “0111000000000000”;
BSTR bstrAllocation = (BSTR)IoMap;
long nTimeout = 500;
IUnknown * ptGet = NULL;
ptIoServer->OpenComm(bstrParam, &bstrIoAllocation, nTimeout, &ptGet);
```

■ **書式**

Call　**オブジェクト**.OpenComm(bstrParam,
　　pbstrIoAllocation, nTimeout, ppunkComm) **：（**Visual Basic**）**

オブジェクト->OpenComm(bstrParam,
　　pbstrIoAllocation, nTimeout, ppunkComm); **：（C＋＋言語）**

オブジェクト　：INkeIoServer **インターフェイスを指すオブジェクト式です。**

■ **引数**

bstrParam　　…　**通信パラメータ**
　（データ型：String（Visual Basic）、_bstr_t（C＋＋言語））
bstrAllocation　…　I/O **割付文字列を保存した領域**
　（データ型：String（Visual Basic）、BSTR *（C＋＋言語））
nTimeout　　…　**通信タイムアウト値**[ms]
　（データ型：Long（Visual Basic）、long（C＋＋言語））
ppunkComm　…　**通信オブジェクト**
　（データ型：Object（Visual Basic）、IUnknown **（C＋＋言語））

■ **戻り値**

なし

■ **解説**

指定の通信パラメータで INkeIoComm **インターフェースを作成します。同一コンピュータ内では、同一の通信パラメータで作成される** INkeIoComm **イン**

ターフェイスは 1 つとします。同一通信パラメータで作成された INkeIoComm インターフェイスがある場合、本メソッドでは、その INkeIoComm インターフェイスの参照を返します。
以降は、ここで取得された INkeIoComm インターフェイスを操作することでデバイスとの I/O データ操作を行います。
I/O 割付文字列の設定は、16 ビットのデータを'0'または'1'のみ記述した文字列で行います。各文字は、32 点分の I/O 設定に対応しており、

0＝入力、1＝出力

を設定します。各文字とアドレスの対応は、下記の通りです。

1 文字目　：アドレス　0 ～ 31
2 文字目　：アドレス　32 ～ 63

⋮

14 文字目　：アドレス　448 ～ 479
15 文字目　：アドレス　480 ～ 511

例えば、

strIoAllocation = “0111000000000000”

この場合、1,2,3 ビット目が'1'なので
32～63、64～95、96～127 のアドレスに位置するモジュールは出力扱いになります。

6.4 INkeIoComm インターフェイス

INkeIoComm インターフェイスは、Visual Basic では「NkeIoComm」形、C＋＋言語では「INkeIoCommPtr」形として使用します。

INkeIoComm を使用するための宣言する書式例は、下記のとおりです。

<記述例>

（Visual Basic）

```
Private IoComm As NkeIoComm
```

（C＋＋言語）

```
INkeIoCommPtr ptIoComm
```

6.4-1 プロパティ

UniState

■ 機能

UNILINE の状態を取得します。

■ 記述例

（Visual Basic）

```
Dim ULState As Long
ULState = IoComm.UniState
```

（C＋＋言語）

```
enumNkeAxIo_UniState nULState = ptIoComm->UniState;
```

■ 書式

値 = オブジェクト.UniState　：（Visual Basic）

値 = オブジェクト->UniState;　：（C＋＋言語）

値　：戻り値を格納するデータです。

オブジェクト　：INkeIoComm インターフェイスを指すオブジェクト式です。

■ 引数

なし

■ 戻り値

データ型：　Integer　：（Visual Basic）

enumNkeAxIo_UniState　：（C＋＋言語）

enumNkeAxIo_UniState_None	…	正常
enumNkeAxIo_UniState_DGShort	…	D-G 間短絡
enumNkeAxIo_UniState_LineBreak	…	D-G ライン断線
enumNkeAxIo_UniState_DPShort	…	D-24 間短絡
enumNkeAxIo_UniState_LoVolt	…	電圧降下
enumNkeAxIo_UniState_Sizing	…	サイジング中
enumNkeAxIo_UniState_Running	…	伝送中
enumNkeAxIo_UniState_OnErr	…	ON 異常

■ 解説

UNILINE の状態が返されます。

例えば、

オブジェクト.UniState = enumNkeAxIo_UniState_DGShort

この場合、D-G 間短絡が発生しています。

複数の UNILINE 状態を調べるときは、条件を OR で結合して検査します。

bValue

■ 機能

接点アドレスの I/O 状態を、１点ずつ設定／取得します。

■ 記述例

（Visual Basic）

```
Dim IOAdr As Long
Dim IOBit As Long
IOAdr = 127
IOBit = IoComm.bValue(IOAdr)          ‘ I/O 状態の取得
IOAdr = 255
IOBit = 1
IoComm.bValue(IOAdr) = IOBit          ‘ I/O 状態の設定
```

（Ｃ＋＋言語）

```
long lIOBit = ptIoComm->bValue[127]; // I/O 状態の取得
ptIoComm->bValue[255] = 1;           // I/O 状態の設定
```

■ 書式

[値の取得]

値 = オブジェクト.bValue(nAdr)　　：（Visual Basic）

値 = オブジェクト->bValue[nAdr];　　：（Ｃ＋＋言語）

[値の設定]

オブジェクト.bValue(nAdr) = 値　　：（Visual Basic）

オブジェクト->bValue[nAdr] = 値;　　：（Ｃ＋＋言語）

値　　：I/O 状態を１点単位で取得、設定するデータです。

オブジェクト　：INkeIoComm インターフェイスを指すオブジェクト式です。

■ 引数

nAdr　　…　接点アドレス [0 … 511]

（データ型：Integer（Visual Basic）、long（Ｃ＋＋言語））

■ 戻り値

データ型：　Long　　：（Visual Basic）

long　　：（Ｃ＋＋言語）

ビット値 0:Off, 1:On

■ 解説

接点アドレス*(nAdr)*の I/O 状態を、１点単位で設定、あるいは取得します。

I/O 状態は、内部バッファから設定、取得します。

内部バッファを最新にするため、bValue プロパティから I/O 状態を取得する前には、必ず UpdateReadCache メソッドを実行してください。

また、bValue プロパティに I/O 状態を設定した後は、Ethernet 対応 LAN ゲートウェイに反映するため、必ず UpdateWriteCache メソッドを実行してください。

Value

■ **機能**

接点アドレスからの I/O **状態を、指定の点数で設定／取得します。**

■ **記述例**

（Visual Basic**）**

```
Dim IOAdr As Long
Dim IOBits As Long
Dim IOData As Long
IOAdr = 105
IOBits = 4
IOData = IoComm.Value(IOAdr, IOBits)      ‘ I/O 状態の取得
IOAdr = 5
IOBits = 3
IOData = 6
IoComm.bValue(IOAdr, IOBits) = IOData     ‘ I/O 状態の設定
```

（C＋＋言語）

```
long lIOBits = ptIoComm->Value[105][4];     // I/O 状態の取得
ptIoComm->Value[5][3] = 6;                  // I/O 状態の設定
```

■ **書式**

[**値の取得**]

値 = **オブジェクト**.Value(nAdr, nBits)　　**：（**Visual Basic**）**

値 = **オブジェクト**->Value[nAdr][nBits];　　**：（C＋＋言語）**

[**値の設定**]

オブジェクト.Value(nAdr, nBits) = **値**　　**：（**Visual Basic**）**

オブジェクト->Value[nAdr][nBits] = **値**;　　**：（C＋＋言語）**

値　　**：**I/O **状態を設定／取得するデータです。**

オブジェクト　：INkeIoComm **インターフェイスを指すオブジェクト式です。**

■ **引数**

nAdr　　**…　接点アドレス** [0 … 511]

（データ型：Integer（Visual Basic）、long（**C＋＋言語**））

nBits　　**…　点数**　　[1 … 32]

（データ型：Integer（Visual Basic）、long（**C＋＋言語**））

※アドレスと点数の合計は、必ず 1 **以上、**512 **以下としてください。**

■ **戻り値**

データ型：　Long　　**：（**Visual Basic**）**

long　　**：（C＋＋言語）**

I/O **状態。最大** 32 **点分まで、設定／取得できます。**

■ **解説**

接点アドレス*(nAdr)***から** I/O **状態を点数***(nBits)***の数だけ設定あるいは取得します。**

I/O **状態は、内部バッファから設定、取得します。**

内部バッファを最新にするため、Value **プロパティから** I/O **状態を取得する前には、必ず** UpdateReadCache **メソッドを実行してください。**

また、Value **プロパティに** I/O **状態を設定した後は、**Ethernet **対応** LAN **ゲー**

トウェイに反映するため、必ず UpdateWriteCache メソッドを実行してください。

I/O 状態は、最大 32 点分まで設定／取得できます。
I/O 状態は、２進数で表記したデータを、整数に変換して設定／取得します。
２進数の最小単位はビットです。１つのビットは、I/O 状態の１点に対応します。
I/O 状態の各ビットと、引数で指定した接点アドレスの対応は、下記の通りです。

ビット 0　：接点アドレス
ビット 1　：接点アドレス + 1
⋮
ビット n　：接点アドレス + 点数 - 1

例えば、

アドレス 5 の接点を OFF
アドレス 6 の接点を ON
アドレス 7 の接点を ON

に設定したいときは、設定する I/O 状態を

OFF に設定するビット　= 0
ON に設定するビット　= 1

として、２進数に置き換えます。置き換えると、

0000 0000 0000 0000 0000 0000 0000 0110

になるので、これを整数に変換すると

0000 0000 0000 0000 0000 0000 0000 0110 = 6

になります。
（２進数から整数への置き換えには、関数電卓などを使うと、便利です。）
実際のプログラムは、下記の通りになります。

（Visual Basic）

```
Dim IOAdr As Long    ‘ 先頭の接点アドレスです。
Dim IOBits As Long   ‘ 点数です。
Dim IOData As Long   ‘ 設定したい I/O 状態です。
IOAdr = 5            ‘ アドレス=5 から設定します。
IOBits = 3           ‘  3 点分だけ設定します。
IOData = 6           ‘  I/O 状態は、整数で書きます。
IoComm.bValue(IOAdr, IOBits) = IOData
```

（C＋＋言語）

```
ptIoComm->Value[5][3] = 6;
```

なお、接点アドレスと点数の合計は、必ず 1 以上、512 以下となるように設定してください。アドレス=511、点数=32 のように、範囲外の値をとらないようにしてください。

<例>

◯ 接点アドレス = 508, 点数 = 8
　508 + 8 = 510 ≦ 512
× 接点アドレス = 511, 点数 = 32
　511 + 32 = 543 ＞ 512

6.4-2 **メソッド**

UpdateReadCache

■ **機能**

Ethernet **対応** LAN **ゲートウェイから内部バッファに値を入力します。**

■ **記述例**

（Visual Basic**）**

```
Call IoComm.UpdateReadCache
```

（Ｃ＋＋言語）

```
ptIoComm->UpdateReadCache();
```

■ **書式**

Call　**オブジェクト**.UpdateReadCache　**：（**Visual Basic**）**

オブジェクト->UpdateReadCache();　**：（Ｃ＋＋言語）**

オブジェクト　：INkeIoComm **インターフェイスを指すオブジェクト式です。**

■ **引数**

なし

■ **戻り値**

なし

■ **解説**

Ethernet **対応** LAN **ゲートウェイから読み込んだ最新の値で、内部バッファを更新します。**

内部バッファの値は、bValue**、**Value **プロパティから読み出すことができます。**

UpdateWriteCache

■ **機能**

内部バッファから Ethernet **対応** LAN **ゲートウェイに値を出力します。**

■ **記述例**

（Visual Basic**）**

```
Call IoComm.UpdateWriteCache
```

（Ｃ＋＋言語）

```
ptIoComm->UpdateWriteCache();
```

■ **書式**

Call　**オブジェクト**.UpdateWriteCache　**：（**Visual Basic**）**

オブジェクト->UpdateWriteCache();　**：（Ｃ＋＋言語）**

オブジェクト　：INkeIoComm **インターフェイスを指すオブジェクト式です。**

■ **引数**

なし

■ **戻り値**

なし

■ 解説

内部バッファに設定した値を Ethernet 対応 LAN ゲートウェイに出力します。

内部バッファの値は、bValue、Value プロパティで書き込むことができます。

GetAbnormalIdCount

■ 機能

Ethernet 対応 LAN ゲートウェイから異常 ID ターミナルの台数を取得します。

■ 記述例

（Visual Basic）

```
Dim nCount As Long
Call IoComm.GetAbnormalIdCount(nCount)
```

（C＋＋言語）

```
long lCount = 0;
ptIoComm->GetAbnormalIdCount(&lCount);
```

■ 書式

Call　オブジェクト.GetAbnormalIdCount(pnCount)：（Visual Basic）

オブジェクト->GetAbnormalIdCount(pnCount);　：（C＋＋言語）

オブジェクト　：INkeIoComm インターフェイスを指すオブジェクト式です。

■ 引数

pnCount　…　異常 ID ターミナルの台数を保存する領域

（データ型：Long（Visual Basic）、long *（C＋＋言語））

■ 戻り値

なし

■ 解説

Ethernet 対応 LAN ゲートウェイから異常 ID ターミナルの台数を取得します。

引数に指定した領域に取得します。

台数が 0 のとき、異常 ID ターミナルは存在しません。

DumpAbnormalId

■ 機能

Ethernet 対応 LAN ゲートウェイから異常 ID ターミナルの検出状態を取得します。

■ 記述例

（Visual Basic）

```
Dim EIDMap As String
Call IoComm.DumpAbnormalId(EIDMap)
```

（C＋＋言語）

```
BSTR bstrEIDMap;
ptIoComm->DumpAbnormalId(&bstrEIDMap);
```

■ 書式

Call　オブジェクト.DumpAbnormalId(pbstrEID)　：（Visual Basic）

オブジェクト->DumpAbnormalId(pbstrEID);　：（C＋＋言語）

オブジェクト ：INkeIoComm インターフェイスを指すオブジェクト式です。

■ 引数

pbstrEID　　…　異常 ID ターミナルの検出状態を保存する領域
（データ型：String（Visual Basic)、BSTR *（C＋＋言語））

■ 戻り値

なし

■ 解説

Ethernet 対応 LAN ゲートウェイから異常 ID ターミナルの検出状態を取得します。異常 ID ターミナルの検出状態は、引数で指定した文字列に取得します。

異常 ID ターミナルとは、Ethernet 対応 LAN ゲートウェイに記憶した ID と異なる ID を設定して接続したターミナル、または Ethernet 対応 LAN ゲートウェイと正常に接続できていないターミナルです。

異常 ID ターミナルの検出状態は、ターミナル 512 台分のデータを'0'または'1'のみ記述した文字列で取得します。各文字は、ターミナル 1 台分に対応しており、

0＝ID 異常未検出、1＝ID 異常検出

を設定します。

各文字とターミナルの対応は、下記の通りです。

1 文字目　　：ターミナル ID　＝　0　の検出状態
2 文字目　　：ターミナル ID　＝　1　の検出状態

⋮

511 文字目　　：ターミナル ID　＝　510 の検出状態
512 文字目　　：ターミナル ID　＝　511 の検出状態

例えば、

pbstrEID = “0111000000000000...000”

この場合、1,2,3 文字目が'1'なので ID = 1,2,3 のターミナルが、異常 ID ターミナルとなります。

NKE株式会社 [旧社名(株)中村機器エンジニアリング]

商品に関するご質問は、フリーダイヤル、もしくは E-メールにてお問い合わせください。
（AM.9:00～PM.5:00　土日、祝祭日休み）

0120-77-2018
promotion@nke.co.jp

● 本社工場　〒617-0828 京都府長岡京市馬場図所 27　　TEL 075-955-0071(代)　FAX 075-955-1063
● 伏見工場　〒612-8487 京都市伏見区羽束師菱川町 366-1　　TEL 075-931-2731(代)　FAX 075-934-8746
● NKE ホームページ ： http://www.nke.co.jp/