Instruction
Manual

FUN プログラミング解説書

TOKYO DENSAN

東京電機産業株式会社

該当製品

DSFR-1,DSFR-8,DSFR-64
名称:　FA-M3 Universal Network Driver

履歴

2005 年 07 月 01 日	初版	新規発行
2006 年 06 月 22 日	第 2 版	LE12 対応に関する説明を追加
		Visual Basic 2005 への対応を追加
2006 年 10 月 31 日	第 3 版	SP66/67 への対応を追加
2007 年 02 月 23 日	第 4 版	Windows Vista への対応を追加
2007 年 08 月 15 日	第 5 版	Windows Server 2003 への対応を追加
2008 年 06 月 13 日	第 6 版	Windows Server 2008 への対応を追加
		Visual Basic 2008 への対応を追加
2008 年 09 月 30 日	第 7 版	Visual C# 2005/2008 への対応を追加
2010 年 03 月 30 日	第 8 版	Windows 7 への対応を追記
		Windows 2000 の記述を削除
2011 年 04 月 30 日	第 9 版	SP22/71/76 への対応を追加
		Visual Basic/ C# 2010 への対応を追加
2012 年 04 月 06 日	第 10 版	Windows 7 64 ビット版(XP モードを含む)
		Windows Server 2008 R2 への対応を追加
2014 年 04 月 18 日	第 11 版	Visual Basic 2012/2013 への対応を追加
		Visual C# 2012/2013 への対応を追加
2015 年 04 月 13 日	第 12 版	InitializeEvent の仕様を変更
		（同一ポートで複数設備の受信を可能に変更）
2015 年 08 月 28 日	第 13 版	Windows 8.1 への対応を追加
2016 年 02 月 17 日	第 14 版	Windows 10 への対応を追加
		Visual Basic/ Visual C# 2015 への対応を追加
2017 年 06 月 13 日	第 15 版	Visual Basic/ Visual C#のバージョン情報を削除
2018 年 09 月 12 日	第 16 版	OS の詳細情報を削除
2020 年 10 月 6 日	第 17 版	商標の記述を変更

ご注意

(1)当社は、本製品に含まれる機能がお客様の特定目的に適合するものを保証するものではありません。
(2)本書の内容の一部または全部を無断で転載、複製すること固くお断りします。
(3)本書の内容については将来予告無しに変更することがございます。
(4)本書の内容については万全を期して作成しておりますが、ご不明な点や誤り、記載もれなどお気付きのことがありましたら、当社営業までご連絡ください。
(5)本製品の使用によりお客様または第三者が被害を被った場合、当社の予測できない本製品の欠陥などのためにお客様または第三者が被った被害およびいかなる間接的損害に対しても当社は責任を負いかねますのでご了承ください。
(6)本製品は特定の１台のコンピュータでご使用ください。
(7)本製品をバックアップなどの目的以外でコピーして使用することは、固くお断りいたします。
(8)本製品の収められているCD－ROM（オリジナルディスク）は大切に保管してください。オリジナルディスクの無い場合は、当社所定の品質保証をお断りすることがあります。
(9)本製品の逆コンパイル、逆アセンブルなど（リバースエンジニアリング）を行うことは固くお断りします。

はじめに

■本書について

本書は、FUN(FA-M3 Universal Network Driver)を利用して FA-M3 との通信プログラムを作成する方法を記述した解説書です。本製品の性質上、開発者向けの解説書となっておりますので、Microsoft Windows 各種 OS の操作/設定方法、Ethernet の設定、Visual Basic / C#の説明、FA-M3 の設定方法などは、それぞれの取扱説明書や、市販の文献をお読み下さい。

■本書の構成

本書の構成を以下に示します。

●1章ソフトウェアの準備
FUN を利用するに当たっての注意点が書かれています。

●2章 FUN の機能を使用する
FUN を利用する方法について記述されています。

●3章 定数一覧
FUN で利用できる定数の表です。

●4章 エラーコード一覧
FUN のエラーコードの表です。

FUN
プログラミング解説書

目次

1　ソフトウェアの準備

FUN(FA-M3 Universal Network Driver)を使用するにあたっての注意点です。

■ソフトウェア使用前に準備することについて

ハードウェアの設定、オペレーティングシステム、各ドライバをそれぞれのマニュアルに従ってインストールして下さい。

2　Visual Basic/Visual C# による FUN の利用

FUN は、Microsoft Visual Basic 6.0/.NET および Microsoft Visual C#環境で開発されるアプリケーションから FA-M3 へアクセスを容易に行うためのドライバプログラムです。

本章では、Visual Basic .NET を使用して FUN を使用する方法、及び FUN で利用する関数について記述します。

Visual Basic 6.0 での使用については、2.20, 2.21 項に注意書きがございます。

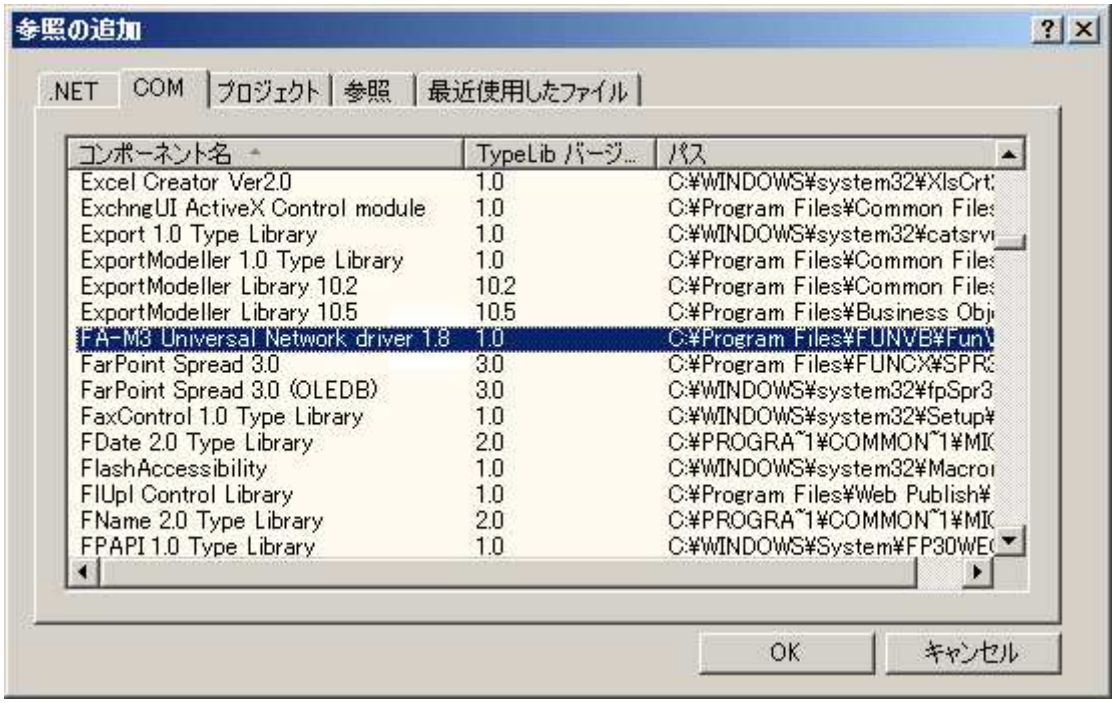
Visual C#での使用については、2.22, 2.23 項に注意書きがございます。

2.1　Visual Basic.NET で FUN を利用するための準備

Visual Basic .NET で FUN を使用するためには、FUN を使用するコンピュータにインストールをする必要があります。詳しくは FUN インストール解説書、第2章 “FUN をインストールする”を参照して下さい。

続いて、Visual Studio のメニュー「プロジェクト」から「参照の追加」を選択して「COM」のプロパティシートを表示し、「FA-M3 Unuversal Network driver」を選択して、OK を押します。

以下は Visiaul Studio 2005 での画面です。

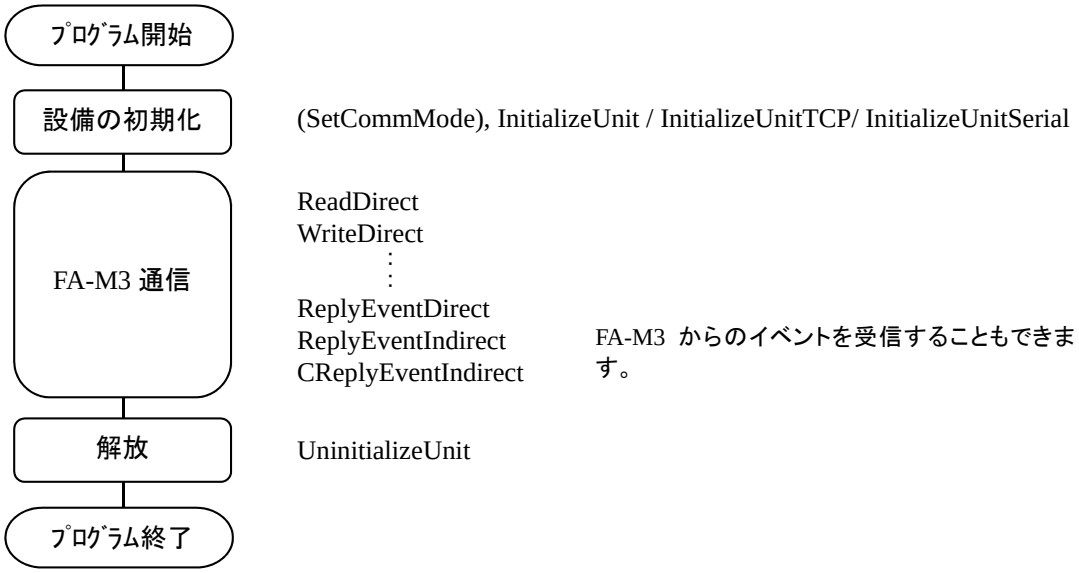


この作業により、Visual Basic .NET のプロジェクトで FUN を使用できるようになります。

2.2　処理の呼び出しについて

FUN を使用し FA-M3 とアクセスするためには、接続する設備を初期化(InitializeUnit、InitializeUnitTCP、InitializeUnitSerial)してからアクセスを行います。終了時には UninitializeUnit を呼び出し、接続を解放してからプログラムを終了します。

Ethernet 通信を初期設定以外の方法で行う場合には、初期化の前に通信モードの設定(SetCommMode)を実行する必要があります。



サンプルプログラム

このサンプルプログラムは ReadDirect によりデータレジスタ“D0001”のデータを読み込み表示します。

```
プログラムの先頭
    Imports FUNVBLib
    Imports FUNVBLib.FunError
    Imports FUNVBLib.FunRegType

以下がコードのサンプル
    Dim M3Obj As FUN = New FUN
    Dim Register(0) As Integer
    Dim DataBuf() As Integer = Nothing
    Dim nError As FunError

    nError = M3Obj.InitializeUnit(1, "192.168.1.1")
    If nError = funOK Then
        Register(0) = funD + 1
        nError = M3Obj.ReadDirect(1, 1, 1, Register, DataBuf)
        If nError = funOK Then
            TextBox1.Text = DataBuf(0)
        End If
        nError = M3Obj.UninitializeUnit(1)
    End If
```

2.3 一般関数

2.3.1 InitializeUnit

●動作

指定された設備を Ethernet 通信モード（UDP/IP プロトコル）で初期化します。

実際に設備が接続されていないときでも実行することができます。

SP66/SP67/SP71/SP76 の内蔵 Ethernet 通信ポートを使用する場合は、CPU プロパティで上位リンクサービスのプロトコルを UDP/IP に設定してください。

FUN の標準動作では、LE01 の rev3 以前または LE12 には未実装の、拡張上位リンクサービスを使用して通信を行う設定となっております。

よって、LE01 の rev3 以前、または LE12 を使用するときは、初期化前に SetCommMode メソッドに funModeDisableSeq パラメータを指定して実行する必要があります。

また、LE11/LE12/SP66/SP67/SP71/SP76 でポート番号 12291 を使用するときは、初期化前に SetCommMode メソッドに funModeChangePort パラメータを指定して実行する必要があります。

●定義

Function InitializeUnit(ByVal UnitNo As Integer, ByVal HostName As String) As FUNVBLib.FunError

●パラメータ

UnitNo	設備番号 (1～最大設備数) 最大設備数はご購入された製品のライセンスによって異なります。
HostName	FA-M3 のホスト名、または IP アドレス

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funAlreadyUsed	指定された設備はすでに初期化済み
funFailed	初期化に失敗
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1 を 192.168.1.1 として初期化します。

プログラムの先頭

```
Imports FUNVBLib
```

以下がコードのサンプル

```
Dim M3Obj As FUN = New FUN
Dim nError As FunError

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)
```

●注意

データコードがバイナリの場合は、SetCommMode メソッドに funModeBinary パラメータを指定して実行する必要があります。

2.3.2 InitializeUnitTCP

●動作

指定された設備を Ethernet 通信モード（TCP/IP プロトコル）で初期化します。

実際に設備が接続されていないときは、初期化を行うことはできません。

SP66/SP67/SP71/SP76 の内蔵 Ethernet 通信ポートを使用する場合は、CPU プロパティで上位リンクサービスのプロトコルを TCP/IP に設定してください。

FUN の標準動作では、LE01 の rev3 以前または LE12 には未実装の、拡張上位リンクサービスを使用して通信を行う設定となっております。

よって、LE01 の rev3 以前、または LE12 を使用するときは、初期化前に SetCommMode メソッドに funModeDisableSeq パラメータを指定して実行する必要があります。

また、LE11/LE12/SP66/SP67/SP71/SP76 でポート番号 12291 を使用するときは、初期化前に SetCommMode メソッドに funModeChangePort パラメータを指定して実行する必要があります。

●定義

Function InitializeUnitTCP(ByVal UnitNo As Integer, ByVal HostName As String) As FUNVBLib.FunError

●パラメータ

UnitNo	設備番号 (1～最大設備数) 最大設備数はご購入された製品のライセンスによって異なります。
HostName	FA-M3 のホスト名、または IP アドレス

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funAlreadyUsed	指定された設備はすでに初期化済み
funFailed	初期化に失敗
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1 を 192.168.1.1 として初期化します。

```
プログラムの先頭
    Imports FUNVBLib

以下がコードのサンプル
    Dim M3Obj As FUN = New FUN
    Dim nError As FunError

    nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)
```

●注意

データコードがバイナリの場合は、SetCommMode メソッドに funModeBinary パラメータを指定して実行する必要があります。

2.3.3 InitializeUnitSerial

●動作

指定された設備をシリアル通信モードで初期化します。

実際に設備が接続されていないときでも実行することができます。

Windows 7 の Windows XP モード上では、本メソッドは未サポートとなります。

●定義

```
Function InitializeUnitSerial(
    ByVal UnitNo As Integer, ByVal PortNo As Integer, ByVal CommType As FUNVBLib.FunComm,
    Optional ByVal StationNo As Integer = 1, Optional ByVal Baudrate As Integer = 0,
    Optional ByVal ByteSize As Integer = 8, Optional ByVal Parity As FUNVBLib.FunParity = funNoParity,
    Optional ByVal StopBit As Integer = 1, Optional ByVal CheckSum As Boolean = False,
    Optional ByVal TermChar As Boolean = False) As FUNVBLib.FunError
```

●パラメータ

UnitNo	設備番号 (1～最大設備数) 最大設備数はご購入された製品のライセンスによって異なります。
PortNo	使用する COM ポートの番号 (1～99)
CommType	通信タイプ 0... CPU モジュール直結　1... パソコンリンクモジュール
StationNo	ステーション番号 常に 1 を指定 （省略時は 1）
Baudrate	通信速度 300,600,1200,2400,4800,9600,14400,19200,28800,38400,57600,115200 のいずれかを指定します。 CommType が CPU モジュール直結の場合は、0 を指定すると、自動検出になります。 その際は、ByteSize 以降のパラメータは無視されます。 （省略時は 0）
ByteSize	データ長 7,8 のいずれかを指定します。（省略時は 8）
Parity	パリティ 0... パリティ無し　1... 奇数　2... 偶数 （省略時は 0）
StopBit	ストップビット長 1,2 のいずれかを指定します。（省略時は 1）
CheckSum	チェックサム True... 使用する　False... 使用しない （省略時は False）
TermChar	終端文字 True... 使用する　False... 使用しない （省略時は False）

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funAlreadyUsed	指定された設備はすでに初期化済み
funFailed	初期化に失敗
funInvalidInstall	インストールされているファイルが破損している
funCannotOpen	通信ポートがオープンできない

●例

設備 1 を、

- COM ポート = 1
- 通信タイプ = パソコンリンクモジュール
- ステーション番号 = 1
- 通信速度 = 9600
- データ長 = 8
- パリティ = なし
- ストップビット長 = 1
- チェックサム = 使用しない
- 終端文字 = 使用しない

として初期化します。

プログラムの先頭

```
Imports FUNVBLib
```

以下がコードのサンプル

```
Dim M3Obj As FUN = New FUN
Dim nError As FunError

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)
```

2.3.4 UninitializeUnit

●動作

指定された設備の使用を終了します。

実際に設備が接続されていないときでも実行することができます。

●定義

Function UninitializeUnit(ByVal UnitNo As Integer) As FUNVBLib.FunError

●パラメータ

UnitNo	設備番号 (1～最大設備数)。最大設備数はご購入された製品のライセンスによって変化します。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1 の使用を終了します。

プログラムの先頭

```
Imports FUNVBLib
```

以下がコードのサンプル

```
Dim M3Obj As FUN = New FUN
Dim nError As FunError

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

nError = M3Obj.UninitializeUnit(1)
```

2.3.5 SetTimeout

●動作

指定された設備の通信タイムアウト時間を設定します。

通信タイムアウト時間は、FA-M3 へコマンドを送信した後に応答待ちを行う時間のことです。

（通信タイムアウト時間とは、FA-M3 にコマンドを送信した後に応答待ちを行う時間のことです。）

コマンドを送信した後に通信タイムアウト時間が経過しても FA-M3 から応答がない場合はコマンドを再送するので、実際の応答待ち時間は "通信タイムアウト時間×(通信コマンド再送回数+1)" となります。

ただし、TCP/IP プロトコル使用時で SetCommMode メソッドに funModeDisableSeq パラメータを指定したときはコマンドの再送を行わないため、実際の応答待ち時間は通信タイムアウト時間と同じになります。

設定は 1/1000 秒単位で行います。初期値として、1000ms が設定されています。

●定義

Function SetTimeout(ByVal UnitNo As Integer, ByVal TimeOut As Integer) As FUNVBLib.FunError

●パラメータ

UnitNo	設備番号 (0～最大設備数)。最大設備数は製品のライセンスによって変化します。 0 を指定したときは、初期化済みのすべての設備に対して設定を行います。
TimeOut	通信タイムアウト時間 (10～60000) 通信タイムアウト時間の設定が短過ぎると通信エラーが頻発するときがあります。 回線の負荷状況や、お使いのパソコンの処理能力を考慮した上で、適切な値を設定して下さい

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1 の通信タイムアウト時間を 1 秒に設定します。

プログラムの先頭

```
Imports FUNVBLib
```

以下がコードのサンプル

```
Dim M3Obj As FUN = New FUN
Dim nError As FunError

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

nError = M3Obj.SetTimeout(1, 1000)
```

2.3.6 SetRetryCounter

●動作

指定された設備の通信コマンド再送回数を設定します。

（通信コマンド再送回数とは FA-M3 にコマンドを送信した後、通信タイムアウト時間が経過しても応答が無かった場合にコマンドの再送を行う回数のことです。）

TCP/IP プロトコル使用時で SetCommMode メソッドに funModeDisableSeq パラメータを指定したときはコマンドの再送は行われないため、本メソッドでの設定は無効となります。

初期値として、2 が設定されています。

●定義

Function SetRetryCounter(ByVal UnitNo As Integer, ByVal RetryCount As Integer) As FUNVBLib.FunError

●パラメータ

UnitNo	設備番号 (0～最大設備数)。最大設備数は製品のライセンスによって変化します。 0 を指定したときは、初期化済みのすべての設備に対して設定を行います。
RetryCount	通信コマンド再送回数 (0～12)

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1 の通信コマンド再送回数を 5 回に設定します。

```
プログラムの先頭
    Imports FUNVBLib

以下がコードのサンプル
    Dim M3Obj As FUN = New FUN
    Dim nError As FunError

    nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

    nError = M3Obj.SetRetryCounter(1, 5)
```

2.3.7 SetSign

●動作

指定された設備に対して、16 ビットデータの符号を設定します。
何も設定しない場合、および 32 ビットデータの場合は、符号ありとして動作します。

●定義

Function SetSign(ByVal UnitNo As Integer, ByVal bSigned As FUNVBLib.FunSign) As FUNVBLib.FunError

●パラメータ

UnitNo	設備番号 (0～最大設備数)。最大設備数は製品のライセンスによって変化します。 0 を指定したときは、初期化済みのすべての設備に対して設定を行います。
bSigned	funSigned.....符号あり(-32768～32767) funUnsigned.....符号無し(0～65535)

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1 の 16 ビットデータを符号無しに設定します。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunSign
```

以下がコードのサンプル

```
Dim M3Obj As FUN = New FUN
Dim nError As FunError

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

nError = M3Obj.SetSign(1, funUnsigned)
```

2.3.8 SetCommMode

●動作

指定された設備の Ethernet 通信モードを設定します。

指定された設備が Ethernet モジュールまたは SP66/SP67/SP71/SP76 の内蔵 Ethernet 通信ポートに対して通信する場合のみ設定する必要があります。使用する Ethernet 通信モジュールに合わせて設定してください。

設定を行わない場合は「データコード=ASCII, 拡張上位リンクサービスを使用する, ポート 12289 に接続」になります。

本メソッドは通信初期化の前／後、どちらでも実行可能です。

ただし、通信初期化後に実行した場合にはデータコードの設定のみが有効になります。

●定義

```
Function SetCommMode(ByVal UnitNo As Integer, ByVal CommMode As FUNVBLib.FunMode
        ) As FUNVBLib.FunError
```

●パラメータ

UnitNo	設備番号 (0～最大設備数)。最大設備数は製品のライセンスによって変化します。 0 を指定したときは、すべての設備に対して設定を行います。
CommMode	以下のパラメータを加算した値を指定します
	funModeASCII.....データコード=ASCII で通信を行います
	funModeBinary.....データコード=バイナリで通信を行います
	funModeDisableSeq.....拡張上位リンクサービスを使用しません LE01 の rev3 以前、または LE12 を使用する場合は必ず指定します
	funModeChangePort....接続するポート番号を 12291 にします LE11/LE12/SP66/SP67/SP71/SP76 を使用する場合にのみ利用可能です。 LE11/LE12 でこのパラメータを指定したときは、データコードのパラメータは、通信モジュールのデータコードの設定とは逆に設定してください。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunMode
```

設備 1 の通信モードを「データコード=バイナリ，拡張上位リンクサービスを使用する，ポート 12289 に接続」に設定します。

```
Dim M3Obj As FUN = New FUN
Dim nError As FunError

nError = M3Obj.SetCommMode(1, funModeBinary)
nError = M3Obj.InitializeUnit(1, “192.168.1.1”)
```

設備 1 の通信モードを「データコード=ASCII，拡張上位リンクサービスを使用しない，ポート 12291 に接続」に設定します。

```
Dim M3Obj As FUN = New FUN
Dim nError As FunError

nError = M3Obj.SetCommMode(1, funModeAscii + funModeDisableSeq + funModeChangePort)
nError = M3Obj.InitializeUnit(1, “192.168.1.1”)
```

2.3.9 SetEventMode

●動作

指定された設備がイベントを通知する時のデータ型を設定します。

設定を行わない場合は 16 ビット整数型で通知されます。

シリアル通信の場合は、イベント受信前に funEventChar パラメータを指定して実行し、常に文字列（終端文字処理なし）で受信する必要があります。

●定義

Function SetEventMode(ByVal UnitNo As Integer, ByVal EventMode As Short) As FUNVBLib.FunError

●パラメータ

UnitNo	設備番号 (0～最大設備数)。最大設備数は製品のライセンスによって変化します。 0 を指定したときは、初期化済みのすべての設備に対して設定を行います。
EventMode	funEventInteger.....16 ビット整数でイベントを通知
	funEventChar.....文字列でイベントを通知（終端文字処理なし）
	0～255..... 文字列でイベントを通知（指定した値を終端文字とする）

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1 の通信モードを文字列（終端文字処理無し）によるイベント通知に設定します。

```
プログラムの先頭
    Imports FUNVBLib
    Imports FUNVBLib.FunEventMode

以下がコードのサンプル
    Dim M3Obj As FUN = New FUN
    Dim nError As FunError

    nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

    nError = M3Obj.SetEventMode(1, funEventChar)
```

2.4 シリアル通信に関する注意点

Windows のデバイスマネージャを利用して COM ポートの番号を変更されたときは、必ずマシンの再起動を行ってください。

シリアル通信でイベントを受信する場合は、必ずイベント受信前に SetEventMode メソッドに funEventChar パラメータを指定して実行し、常に文字列（終端文字処理なし）で受信する必要があります。

2.5　16 ビットデータアクセス関数

2.5.1　ReadDirect

●動作

指定したデバイスから 1 ワード単位でデータを読み込みます。

●定義

Function ReadDirect(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByRef Register As System.Array, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～32)
Register	データを読み込むレジスタを指定した Integer 型配列変数を設定します。 レジスタは、レジスタ ID × &H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。

0... D レジスタ　1... I リレー　2... R レジスタ
3... E リレー　4... B レジスタ
5... タイマー現在値（カウントダウン形）
6... カウンタ現在値（カウントダウン形）
7... タイマー現在値（カウントアップ形）
8... カウンタ現在値（カウントアップ形）
9... タイムアップリレー　10... カウントアップリレー　11... タイマー設定値
12... カウンタ設定値　13... W レジスタ　14... L リレー
15... X リレー　16... Y リレー　17... Z レジスタ
18... M リレー　19... V レジスタ　20... F レジスタ

RetVal	読み込んだデータを格納する Integer 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。　このとき、レジスタ ID にリレーを指定した場合は、指定したアドレスから 16 点分のデータがビット 1～ビット 16 に代入された 16 ビットワードデータが返されます。
FunInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003 のデータを変数 DataBuf に読み込む。

```
Dim DataBuf() As Integer = Nothing
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, "192.168.1.1")

Register(0) = funD + 3
nError = M3Obj.ReadDirect(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の D00003 と I00004～I00019 のデータを変数 DataBuf に読み込む。

```
Dim DataBuf() As Integer = Nothing
Dim Register(1) As Integer
Dim nError As funError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register(0) = funD + 3
Register(1) = funI + 4
nError = M3Obj.ReadDirect(1, 2, 2, Register, DataBuf)
```

2.5.2 WriteDirect

●動作

指定したデバイスに 1 ワード単位でデータを書き込みます。

●定義

Function WriteDirect(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByRef Register As System.Array, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～32)
Register	データを書き込むレジスタを指定した Integer 型配列変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ 5... タイマー現在値（カウントダウン形） 6... カウンタ現在値（カウントダウン形） 7... タイマー現在値（カウントアップ形） 8... カウンタ現在値（カウントアップ形） 9... タイムアップリレー　10... カウントアップリレー　13... W レジスタ 14... L リレー　16... Y リレー　19... V レジスタ 20... F レジスタ タイマー設定値とカウンタ設定値への書き込みはできません。 配列の長さは DataNum で指定した数以上の長さでなければなりません。
WriteVal	レジスタに書き込むデータを指定した Integer 型配列変数を設定します。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は-32768～65535(タイマ,カウンタは 0～32767)の範囲で指定します。 レジスタ ID にリレーを選択した場合は、(アドレス)～(アドレス+15)に(データのビット 1)～(データのビット 16)が書き込まれます。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003 に 4 を書き込む。

```
Dim DataBuf(0) As Integer
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1")

DataBuf(0) = 4
Register(0) = funD + 3
nError = M3Obj.WriteDirect(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の D00003 に 4、R00005 に 6 を書き込む。

```
Dim DataBuf(1) As Integer
Dim Register(1) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register(0) = funD + 3
Register(1) = funR + 5
DataBuf(0) = 4
DataBuf(1) = 6
nError = M3Obj.WriteDirect(1, 2, 2, Register, DataBuf)
```

2.5.3 ReadDump

●動作

デバイスの指定したアドレスから、連続したデータを 1 ワード単位で読み込みます。

●定義

Function ReadDump(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～64)
Register	データを読み込む先頭レジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。

0... D レジスタ	1... I リレー	2... R レジスタ
3... E リレー	4... B レジスタ	
5... タイマー現在値（カウントダウン形）		
6... カウンタ現在値（カウントダウン形）		
7... タイマー現在値（カウントアップ形）		
8... カウンタ現在値（カウントアップ形）		
9... タイムアップリレー	10... カウントアップリレー	11... タイマー設定値
12... カウンタ設定値	13... W レジスタ	14... L リレー
15... X リレー	16... Y リレー	17... Z レジスタ
18... M リレー	19... V レジスタ	20... F レジスタ

RetVal	読み込んだデータを格納する Integer 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。 このとき、レジスタ ID にリレーを指定した場合は、指定したアドレスから 16 点分のデータがビット 1～ビット 16 に代入された 16 ビットワードデータが返されます。
FunInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003～D00055 のデータを変数 DataBuf に読み込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf() As Integer = Nothing
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

Register = funD + 3
nError = M3Obj.ReadDump(1, 2, 53, Register, DataBuf)
```

2.5.4 WriteDump

●動作

デバイスの指定したアドレスから、連続したデータを 1 ワード単位で書き込みます。

●定義

Function WriteDump(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～64)
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ 5... タイマー現在値（カウントダウン形） 6... カウンタ現在値（カウントダウン形） 7... タイマー現在値（カウントアップ形） 8... カウンタ現在値（カウントアップ形） 9... タイムアップリレー　10... カウントアップリレー　13... W レジスタ 14... L リレー　16... Y リレー　19... V レジスタ 20... F レジスタ タイマー設定値とカウンタ設定値への書き込みはできません。
WriteVal	書き込むデータを格納した Integer 型配列です。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は-32768～65535(タイマ,カウンタは 0～32767)の範囲で指定します。 レジスタ ID にリレーを選択した場合は、(アドレス)～(アドレス+15)に(データのビット 1)～(データのビット 16)が書き込まれます。

●戻り値

funOK	正常終了
FunInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003～D00006 に 1,2,3,4 を書き込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf(3) As Integer
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register = funD + 3
DataBuf(0) = 1
DataBuf(1) = 2
DataBuf(2) = 3
DataBuf(3) = 4
nError = M3Obj.WriteDump(1, 2, 4, Register, DataBuf)
```

2.5.5 ReadDumpEx

●動作

デバイスの指定したアドレスから、連続したデータを 1 ワード単位で読み込みます。
ReadDump よりも高速ですが、アクセス時のデータの同時性は保障されません。

●定義

Function ReadDumpEx(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByVal Register As Integer, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～524288) 内部では最大 251 データ単位でアクセスを行います。
Register	データを読み込む先頭レジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ　13... W レジスタ 14... L リレー　15... X リレー　16... Y リレー 17... Z レジスタ　18... M リレー　19... V レジスタ 20... F レジスタ レジスタ ID にリレーを選択する場合は、アドレスは(16 の倍数+1)でなければなりません。
RetVal	読み込んだデータを格納する Integer 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。 このとき、レジスタ ID にリレーを指定した場合は、指定したアドレスから 16 点分のデータがビット 1～ビット 16 に代入された 16 ビットワードデータが返されます。
FunInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003～D01002 のデータを変数 DataBuf に読み込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf() As Integer = Nothing
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, "192.168.1.1")

Register = funD + 3
nError = M3Obj.ReadDumpEx(1, 2, 1000, Register, DataBuf)
```

2.5.6 WriteDumpEx

●動作

デバイスの指定したアドレスから、連続したデータを 1 ワード単位で書き込みます。
WriteDump よりも高速ですが、アクセス時のデータの同時性は保障されません。

●定義

Function WriteDumpEx(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByVal Register As Integer, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～524288) 内部では最大 246 データ単位でアクセスを行います。
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ　13... W レジスタ 14... L リレー　16... Y リレー　19... V レジスタ 20... F レジスタ レジスタ ID にリレーを選択する場合は、アドレスは(16 の倍数+1)でなければなりません。
WriteVal	書き込むデータを格納した Integer 型配列です。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は-32768～65535 の範囲で指定します。 レジスタ ID にリレーを選択した場合は、(アドレス)～(アドレス+15)に(データのビット 1)～(データのビット 16)が書き込まれます。

●戻り値

funOK	正常終了
FunInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003～D00006 に 1,2,3,4 を書き込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf(3) As Integer
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register = funD + 3
DataBuf(0) = 1
DataBuf(1) = 2
DataBuf(2) = 3
DataBuf(3) = 4
nError = M3Obj.WriteDumpEx(1, 2, 4, Register, DataBuf)
```

2.6　32 ビットデータアクセス関数

2.6.1　LReadDirect

●動作

指定したデバイスから 2 ワード単位でデータを読み込みます。

●定義

Function LReadDirect(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByRef Register As System.Array, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～16)
Register	データを読み込むレジスタを指定した Integer 型配列変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ　9... タイムアップリレー 10... カウントアップリレー　13… W レジスタ　14… L リレー 15… X リレー　16… Y リレー　17... Z レジスタ 18... M リレー　19... V レジスタ　20... F レジスタ 配列の長さは DataNum で指定した数以上の長さでなければなりません。
RetVal	読み込んだデータを格納する Integer 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。 このとき、レジスタ ID にリレーを指定した場合は、指定したアドレスから 32 点分のデータがビット 1～ビット 32 に代入された 32 ビットワードデータが返されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003,D00004 の 2 ワードデータを変数 DataBuf に読み込む。

```
Dim DataBuf() As Integer = Nothing
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1")

Register(0) = funD + 3
nError = M3Obj.LReadDirect(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の D00003,D00004 と I00004～I00035 のデータを変数 DataBuf に読み込む。

```
Dim DataBuf() As Integer = Nothing
Dim Register(1) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register(0) = funD + 3
Register(1) = funI + 4

nError = M3Obj.LReadDirect(1, 2, 2, Register, DataBuf)
```

2.6.2 LWriteDirect

●動作

指定したデバイスに 2 ワード単位でデータを書き込みます。

●定義

Function LWriteDirect(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByRef Register As System.Array, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～16)
Register	データを書き込むレジスタを指定した Integer 型配列変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ　9... タイムアップリレー 10... カウントアップリレー　13... W レジスタ　14... L リレー 16... Y リレー　19... V レジスタ　20... F レジスタ 配列の長さは DataNum で指定した数以上の長さでなければなりません。
WriteVal	レジスタに書き込むデータを指定した Integer 型配列変数を設定します。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は-2147483648～2147483647(タイマ,カウンタは 0～2147450879)の範囲で指定します。 レジスタ ID にリレーを選択した場合は、(アドレス)～(アドレス+31)に(データのビット 1)～(データのビット 32)が書き込まれます。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003,D00004 に&H12345678 を書き込む例。

```
Dim DataBuf(0) As Integer
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

DataBuf(0) = &H12345678
Register(0) = funD + 3
nError = M3Obj.LWriteDirect(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の D00003,D00004 に&H12345678、R00005,R00006 に&H567890ab を書き込む。

```
Dim DataBuf(1) As Integer
Dim Register(1) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

Register(0) = funD + 3
Register(1) = funR + 5
DataBuf(0) = &H12345678
DataBuf(1) = &H567890ab
nError = M3Obj.LWriteDirect(1, 2, 2, Register, DataBuf)
```

2.6.3 LReadDump

●動作

デバイスの指定したアドレスから、連続したデータを 2 ワード単位で読み込みます。

●定義

Function LReadDump(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～32)
Register	データを読み込む先頭レジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。

0... D レジスタ	1... I リレー	2... R レジスタ
3... E リレー	4... B レジスタ	9... タイムアップリレー
10... カウントアップリレー	13... W レジスタ	14... L リレー
15... X リレー	16... Y リレー	17... Z レジスタ
18... M リレー	19... V レジスタ	20... F レジスタ

RetVal	読み込んだデータを格納する Integer 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。 このとき、レジスタ ID にリレーを指定した場合は、指定したアドレスから 32 点分のデータがビット 1～ビット 32 に代入された 32 ビットワードデータが返されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,D00004～D00061,D00062 のデータを変数 DataBuf に読み込む。

```
プログラムの先頭
    Imports FUNVBLib
    Imports FUNVBLib.FunRegType

以下がコードのサンプル
    Dim DataBuf() As Integer = Nothing
    Dim Register As Integer
    Dim nError As FunError
    Dim M3Obj As FUN = New FUN

    nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

    Register = funD + 3
    nError = M3Obj.LReadDump(1, 2, 30, Register, DataBuf)
```

2.6.4 LWriteDump

●動作

デバイスの指定したアドレスから、連続したデータを 2 ワード単位で書き込みます。

●定義

Function LWriteDump(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～32)
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ　9... タイムアップリレー 10... カウントアップリレー　13... W レジスタ　14... L リレー 16... Y リレー　19... V レジスタ　20... F レジスタ
WriteVal	書き込むデータを格納した Integer 型配列です。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は-2147483648～2147483647 の範囲で指定します。 レジスタ ID にリレーを選択した場合は、(アドレス)～(アドレス+31)に(データのビット 1)～(データのビット 32)が書き込まれます。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,D00004～D00011,D00012 に 1,2,3,4,5 を書き込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf(4) As Integer
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register = funD + 3
DataBuf(0) = 1
DataBuf(1) = 2
DataBuf(2) = 3
DataBuf(3) = 4
DataBuf(4) = 5
nError = M3Obj.LWriteDump(1, 2, 5, Register, DataBuf)
```

2.6.5 LReadDumpEx

●動作

デバイスの指定したアドレスから、連続したデータを 2 ワード単位で読み込みます。
LReadDump よりも高速ですが、アクセス時のデータの同時性は保障されません。

●定義

Function LReadDumpEx(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByVal Register As Integer, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～262144) 内部では最大 125 データ単位でアクセスを行います。
Register	データを読み込む先頭レジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ　13... W レジスタ 14... L リレー　15... X リレー　16... Y リレー 17... Z レジスタ　18... M リレー　19... V レジスタ 20... F レジスタ レジスタ ID にリレーを選択する場合は、アドレスは(16 の倍数+1)でなければなりません。
RetVal	読み込んだデータを格納する Integer 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。 このとき、レジスタ ID にリレーを指定した場合は、指定したアドレスから 32 点分のデータがビット 1～ビット 32 に代入された 32 ビットワードデータが返されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,D00004～D02001,D02002 のデータを変数 DataBuf に読み込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf() As Integer = Nothing
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register = funD + 3
nError = M3Obj.LReadDumpEx(1, 2, 1000, Register, DataBuf)
```

2.6.6 LWriteDumpEx

●動作

デバイスの指定したアドレスから、連続したデータを 2 ワード単位で書き込みます。
LWriteDump よりも高速ですが、アクセス時のデータの同時性は保障されません。

●定義

Function LWriteDumpEx(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～262144) 内部では最大 123 データ単位でアクセスを行います。
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ　13... W レジスタ 14... L リレー　16... Y リレー　19... V レジスタ 20... F レジスタ レジスタ ID にリレーを選択する場合は、アドレスは(16 の倍数+1)でなければなりません。
WriteVal	書き込むデータを格納した Integer 型配列です。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は-2147483648～2147483647 の範囲で指定します。 レジスタ ID にリレーを選択した場合は、(アドレス)～(アドレス+31)に(データのビット 1)～(データのビット 32)が書き込まれます。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,D00004～D00011,D00012 に 1,2,3,4,5 を書き込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf(4) As Integer
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register = funD + 3
DataBuf(0) = 1
DataBuf(1) = 2
DataBuf(2) = 3
DataBuf(3) = 4
DataBuf(4) = 5
nError = M3Obj.LWriteDumpEx(1, 2, 5, Register, DataBuf)
```

2.7　64 ビットデータアクセス関数

Visual Basic.NET および Visual C#で使用する関数です。

Visual Basic 6.0 では使用できませんので、ご注意ください。

2.7.1　HReadDirect

●動作

指定したデバイスから 4 ワード単位でデータを読み込みます。

●定義

Function HReadDirect(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByRef Register As System.Array, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～8)
Register	データを読み込むレジスタを指定した Integer 型配列変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ　9... タイムアップリレー 10... カウントアップリレー　13… W レジスタ　14… L リレー 15… X リレー　16… Y リレー　17... Z レジスタ 18... M リレー　19... V レジスタ　20... F レジスタ 配列の長さは DataNum で指定した数以上の長さでなければなりません。
RetVal	読み込んだデータを格納する Long 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。 このとき、レジスタ ID にリレーを指定した場合は、指定したアドレスから 64 点分のデータがビット 1～ビット 64 に代入された 64 ビットワードデータが返されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003,4,5,6 の 4 ワードデータを変数 DataBuf に読み込む。

```
Dim DataBuf() As Long = Nothing
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register(0) = funD + 3
nError = M3Obj.HReadDirect(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の D00003,4,5,6 と I00004～I00067 のデータを変数 DataBuf に読み込む。

```
Dim DataBuf() As Long = Nothing
Dim Register(1) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register(0) =funD + 3
Register(1) =funI + 4

nError = M3Obj.HReadDirect(1, 2, 2, Register, DataBuf)
```

2.7.2 HWriteDirect

●動作

指定したデバイスに 4 ワード単位でデータを書き込みます。

●定義

Function HWriteDirect(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByRef Register As System.Array, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～8)
Register	データを書き込むレジスタを指定した Integer 型配列変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ　9... タイムアップリレー 10... カウントアップリレー　13... W レジスタ　14... L リレー 16... Y リレー　19... V レジスタ　20... F レジスタ 配列の長さは DataNum で指定した数以上の長さでなければなりません。
WriteVal	レジスタに書き込むデータを指定した Long 型配列変数を設定します。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は-9223372036854775808～9223372036854775807 の範囲で指定します。 レジスタ ID にリレーを選択した場合は、(アドレス)～(アドレス+63)に(データのビット 1)～(データのビット 64)が書き込まれます。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003,4,5,6 に&H123456789abcdef0 を書き込む例。

```
Dim DataBuf(0) As Long
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

DataBuf(0) = &H123456789abcdef0
Register(0) = funD + 3
nError = M3Obj.HWriteDirect(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の D00003,4,5,6 に&H123456789abcdef0、R00005,6,7,8 に&H567890abcdef1234 を書き込む。

```
Dim DataBuf(1) As Long
Dim Register(1) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

Register(0) = funD + 3
Register(1) = funR + 5
DataBuf(0) = &H123456789abcdef0
DataBuf(1) = &H567890abcdef1234
nError = M3Obj.HWriteDirect(1, 2, 2, Register, DataBuf)
```

2.7.3 HReadDump

●動作

デバイスの指定したアドレスから、連続したデータを 4 ワード単位で読み込みます。

●定義

Function HReadDump(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～16)
Register	データを読み込む先頭レジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。

0... D レジスタ	1... I リレー	2... R レジスタ
3... E リレー	4... B レジスタ	9... タイムアップリレー
10... カウントアップリレー	13... W レジスタ	14... L リレー
15... X リレー	16... Y リレー	17... Z レジスタ
18... M リレー	19... V レジスタ	20... F レジスタ

RetVal	読み込んだデータを格納する Long 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。 このとき、レジスタ ID にリレーを指定した場合は、指定したアドレスから 64 点分のデータがビット 1～ビット 64 に代入された 64 ビットワードデータが返されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,4,5,6～D00059,60,61,62 のデータを変数 DataBuf に読み込む。

```
プログラムの先頭
    Imports FUNVBLib
    Imports FUNVBLib.FunRegType

以下がコードのサンプル
    Dim DataBuf() As Long = Nothing
    Dim Register As Integer
    Dim nError As FunError
    Dim M3Obj As FUN = New FUN

    nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

    Register = funD + 3
    nError = M3Obj.HReadDump(1, 2, 15, Register, DataBuf)
```

2.7.4 HWriteDump

●動作

デバイスの指定したアドレスから、連続したデータを 4 ワード単位で書き込みます。

●定義

Function HWriteDump(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～16)
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ　9... タイムアップリレー 10... カウントアップリレー　13... W レジスタ　14... L リレー 16... Y リレー　19... V レジスタ　20... F レジスタ
WriteVal	書き込むデータを格納した Long 型配列です。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は-9223372036854775808～9223372036854775807 の範囲で指定します。 レジスタ ID にリレーを選択した場合は、(アドレス)～(アドレス+63)に(データのビット 1)～(データのビット 64)が書き込まれます。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,4,5,6～D00019,20,21,22 に 1,2,3,4,5 を書き込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf(4) As Long
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register = funD + 3
DataBuf(0) = 1
DataBuf(1) = 2
DataBuf(2) = 3
DataBuf(3) = 4
DataBuf(4) = 5
nError = M3Obj.HWriteDump(1, 2, 5, Register, DataBuf)
```

2.7.5 HReadDumpEx

●動作

デバイスの指定したアドレスから、連続したデータを 4 ワード単位で読み込みます。
HReadDump よりも高速ですが、アクセス時のデータの同時性は保障されません。

●定義

Function HReadDumpEx(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByVal Register As Integer, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～131072) 内部では最大 62 データ単位でアクセスを行います。
Register	データを読み込む先頭レジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ　13... W レジスタ 14... L リレー　15... X リレー　16... Y リレー 17... Z レジスタ　18... M リレー　19... V レジスタ 20... F レジスタ レジスタ ID にリレーを選択する場合は、アドレスは(16 の倍数+1)でなければなりません。
RetVal	読み込んだデータを格納する Long 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。 このとき、レジスタ ID にリレーを指定した場合は、指定したアドレスから 64 点分のデータがビット 1～ビット 64 に代入された 64 ビットワードデータが返されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,4,5,6～D03999,4000,4001,4002 のデータを変数 DataBuf に読み込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf() As Long = Nothing
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register = funD + 3
nError = M3Obj.HReadDumpEx(1, 2, 1000, Register, DataBuf)
```

2.7.6 HWriteDumpEx

●動作

デバイスの指定したアドレスから、連続したデータを 4 ワード単位で書き込みます。
HWriteDump よりも高速ですが、アクセス時のデータの同時性は保障されません。

●定義

Function HWriteDumpEx(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～131072) 内部では最大 61 データ単位でアクセスを行います。
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　1... I リレー　2... R レジスタ 3... E リレー　4... B レジスタ　13... W レジスタ 14... L リレー　16... Y リレー　19... V レジスタ 20... F レジスタ レジスタ ID にリレーを選択する場合は、アドレスは(16 の倍数+1)でなければなりません。
WriteVal	書き込むデータを格納した Long 型配列です。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は-9223372036854775808～9223372036854775807 の範囲で指定します。 レジスタ ID にリレーを選択した場合は、(アドレス)～(アドレス+63)に(データのビット 1)～(データのビット 64)が書き込まれます。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,4,5,6～D00019,20,21,22 に 1,2,3,4,5 を書き込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf(4) As Long
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register = funD + 3
DataBuf(0) = 1
DataBuf(1) = 2
DataBuf(2) = 3
DataBuf(3) = 4
DataBuf(4) = 5
nError = M3Obj.HWriteDumpEx(1, 2, 5, Register, DataBuf)
```

2.8　単精度浮動小数点データアクセス関数

2.8.1　FReadDirect

●動作

指定したデバイスから単精度浮動小数点としてデータを読み込みます。

●定義

Function FReadDirect(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByRef Register As System.Array, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～16)
Register	データを読み込むレジスタを指定した Integer 型配列変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ 配列の長さは DataNum で指定した数以上の長さでなければなりません。
RetVal	読み込んだデータを格納する Single 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003,D00004 の単精度浮動小数点データを変数 DataBuf に読み込む。

```
Dim DataBuf() As Single = Nothing
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1")

Register(0) = funD + 3
nError = M3Obj.FReadDirect(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の D00003,D00004 と I00004～I00035 の単精度浮動小数点データを変数 DataBuf に読み込む。

```
Dim DataBuf() As Single = Nothing
Dim Register(1) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register(0) = funD + 3
Register(1) = funI + 4

nError = M3Obj.FReadDirect(1, 2, 2, Register, DataBuf)
```

2.8.2 FWriteDirect

●動作

指定したデバイスに単精度浮動小数点データを書き込みます。

●定義

Function FWriteDirect(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByRef Register As System.Array, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～16)
Register	データを書き込むレジスタを指定した Integer 型配列変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ 配列の長さは DataNum で指定した数以上の長さでなければなりません。
WriteVal	レジスタに書き込むデータを指定した Single 型配列変数を設定します。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は±3.402823E+038 の範囲で指定します。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003,D00004 に 1.23456 を書き込む例。

```
Dim DataBuf(0) As Single
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

DataBuf(0) = 1.23456
Register(0) = funD + 3
nError = M3Obj.FWriteDirect(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の D00003,D00004 に 1.23456、R00005,R00006 に 5.67890 を書き込む。

```
Dim DataBuf(1) As Single
Dim Register(1) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

Register(0) = funD + 3
Register(1) = funR + 5
DataBuf(0) = 1.23456
DataBuf(1) = 5.67890
nError = M3Obj.FWriteDirect(1, 2, 2, Register, DataBuf)
```

2.8.3 FReadDump

●動作

デバイスの指定したアドレスから、連続した単精度浮動小数点データを読み込みます。

●定義

Function FReadDump(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～32)
Register	データを読み込む先頭レジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ
RetVal	読み込んだデータを格納する Single 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,D00004～D00061,D00062 のデータを変数 DataBuf に読み込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf() As Single = Nothing
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register = funD + 3
nError = M3Obj.FReadDump(1, 2, 30, Register, DataBuf)
```

2.8.4 FWriteDump

●動作

デバイスの指定したアドレスから、連続した単精度浮動小数点データを書き込みます。

●定義

Function FWriteDump(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～32)
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ
WriteVal	レジスタに書き込むデータを指定した Single 型配列変数を設定します。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は±3.402823E+038 の範囲で指定します。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,D00004～D00011,D00012 に 1.11, 2.22, 3.33, 4.44, 5.55 を書き込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf(4) As Single
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register = funD + 3
DataBuf(0) = 1.11
DataBuf(1) = 2.22
DataBuf(2) = 3.33
DataBuf(3) = 4.44
DataBuf(4) = 5.55
nError = M3Obj.FWriteDump(1, 2, 5, Register, DataBuf)
```

2.8.5 FReadDumpEx

●動作

デバイスの指定したアドレスから、連続した単精度浮動小数点データを読み込みます。
FReadDump よりも高速ですが、アクセス時のデータの同時性は保障されません。

●定義

Function FReadDumpEx(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByVal Register As Integer, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～262144)
Register	データを読み込む先頭レジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ
RetVal	読み込んだデータを格納する Single 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,D00004～D00601,D00602 のデータを変数 DataBuf に読み込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf() As Single = Nothing
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register = funD + 3
nError = M3Obj.FReadDumpEx(1, 2, 300, Register, DataBuf)
```

2.8.6 FWriteDumpEx

●動作

デバイスの指定したアドレスから、連続した単精度浮動小数点データを書き込みます。
FWriteDump よりも高速ですが、アクセス時のデータの同時性は保障されません。

●定義

Function FWriteDumpEx(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～262144)
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ
WriteVal	レジスタに書き込むデータを指定した Single 型配列変数を設定します。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は±3.402823E+038 の範囲で指定します。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,D00004～D00011,D00012 に 1.11, 2.22, 3.33, 4.44, 5.55 を書き込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf(4) As Single
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register = funD + 3
DataBuf(0) = 1.11
DataBuf(1) = 2.22
DataBuf(2) = 3.33
DataBuf(3) = 4.44
DataBuf(4) = 5.55
nError = M3Obj.FWriteDumpEx(1, 2, 5, Register, DataBuf)
```

2.9　倍精度浮動小数点データアクセス関数

2.9.1　DReadDirect

●動作

指定したデバイスから倍精度浮動小数点としてデータを読み込みます。

●定義

Function DReadDirect(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByRef Register As System.Array, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～8)
Register	データを読み込むレジスタを指定した Integer 型配列変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ 配列の長さは DataNum で指定した数以上の長さでなければなりません。
RetVal	読み込んだデータを格納する Double 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003,4,5,6 の倍精度浮動小数点データを変数 DataBuf に読み込む。

```
Dim DataBuf() As Double = Nothing
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register(0) = funD + 3
nError = M3Obj.DReadDirect(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の D00003,4,5,6 と I00004～I00067 の倍精度浮動小数点データを変数 DataBuf に読み込む。

```
Dim DataBuf() As Double = Nothing
Dim Register(1) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register(0) = funD + 3
Register(1) = funI + 4

nError = M3Obj.DReadDirect(1, 2, 2, Register, DataBuf)
```

2.9.2 DWriteDirect

●動作

指定したデバイスに倍精度浮動小数点データを書き込みます。

●定義

Function DWriteDirect(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByRef Register As System.Array, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～8)
Register	データを書き込むレジスタを指定した Integer 型配列変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ 配列の長さは DataNum で指定した数以上の長さでなければなりません。
WriteVal	レジスタに書き込むデータを指定した Double 型配列変数を設定します。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は±1.79769313486231570E+308 の範囲で指定します。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003,4,5,6 に 1.23456 を書き込む例。

```
Dim DataBuf(0) As Double
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

DataBuf(0) = 1.23456
Register(0) = funD + 3
nError = M3Obj.DWriteDirect(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の D00003,4,5,6 に 1.23456、R00005,6,7,8 に 5.67890 を書き込む。

```
Dim DataBuf(1) As Double
Dim Register(1) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

Register(0) = funD + 3
Register(1) = funR + 5
DataBuf(0) = 1.23456
DataBuf(1) = 5.67890
nError = M3Obj.DWriteDirect(1, 2, 2, Register, DataBuf)
```

2.9.3 DReadDump

●動作

デバイスの指定したアドレスから、連続した倍精度浮動小数点データを読み込みます。

●定義

Function DReadDump(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～16)
Register	データを読み込む先頭レジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ
RetVal	読み込んだデータを格納する Double 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,4,5,6～D00059,60,61,62 のデータを変数 DataBuf に読み込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf() As Double = Nothing
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register = funD + 3
nError = M3Obj.DReadDump(1, 2, 15, Register, DataBuf)
```

2.9.4 DWriteDump

●動作

デバイスの指定したアドレスから、連続した倍精度浮動小数点データを書き込みます。

●定義

Function DWriteDump(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～16)
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ
WriteVal	レジスタに書き込むデータを指定した Double 型配列変数を設定します。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は±1.79769313486231570E+308 の範囲で指定します。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,4,5,6～D00019,20,21,22 に 1.11, 2.22, 3.33, 4.44, 5.55 を書き込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf(4) As Double
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register = funD + 3
DataBuf(0) = 1.11
DataBuf(1) = 2.22
DataBuf(2) = 3.33
DataBuf(3) = 4.44
DataBuf(4) = 5.55
nError = M3Obj.DWriteDump(1, 2, 5, Register, DataBuf)
```

2.9.5 DReadDumpEx

●動作

デバイスの指定したアドレスから、連続した倍精度浮動小数点データを読み込みます。
DReadDump よりも高速ですが、アクセス時のデータの同時性は保障されません。

●定義

Function DReadDumpEx(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByVal Register As Integer, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～131072)
Register	データを読み込む先頭レジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ
RetVal	読み込んだデータを格納する Double 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,4,5,6～D01199,1200,1201,1202 のデータを変数 DataBuf に読み込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf() As Double = Nothing
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register = funD + 3
nError = M3Obj.DReadDumpEx(1, 2, 300, Register, DataBuf)
```

2.9.6 DWriteDumpEx

●動作

デバイスの指定したアドレスから、連続した倍精度浮動小数点データを書き込みます。
DWriteDump よりも高速ですが、アクセス時のデータの同時性は保障されません。

●定義

Function DWriteDumpEx(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～131072)
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ
WriteVal	レジスタに書き込むデータを指定した Double 型配列変数を設定します。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は±1.79769313486231570E+308 の範囲で指定します。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の D00003,4,5,6～D00019,20,21,22 に 1.11, 2.22, 3.33, 4.44, 5.55 を書き込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf(4) As Double
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register = funD + 3
DataBuf(0) = 1.11
DataBuf(1) = 2.22
DataBuf(2) = 3.33
DataBuf(3) = 4.44
DataBuf(4) = 5.55
nError = M3Obj.DWriteDumpEx(1, 2, 5, Register, DataBuf)
```

2.10　文字列データアクセス関数

2.10.1　CReadDump

●動作

デバイスの指定したアドレスから、連続したデータを文字列として読み込みます。

●定義

```
Function CReadDump(
        ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
        ByVal Register As Integer, ByRef RetVal As String,
        Optional ByVal TermChar As Short = 0) As FUNVBLib.FunError
```

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの文字列長(1～128)
Register	データを読み込む先頭レジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ
RetVal	読み込んだデータを格納する String 型変数です。
TermChar	0～255 を指定した場合は，その文字コードに対応する文字を終端文字として，読み込んだ文字列に終端文字がある場合は終端文字より前の文字列を RetVal に返します。 -1 を指定した場合は，終端文字の処理は行なわずに DataNum で指定した全ての文字を返します。 省略時は，Chr$(0)が終端文字となります。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
FunInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003 から，0 を終端文字として最大 53 文字の文字列データを変数 DataBuf に読み込む。

```
Dim DataBuf As String = Nothing
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

Register = funD + 3
nError = M3Obj.CReadDump(1, 2, 53, Register, DataBuf)
```

設備 1、CPU2 の B00100 から，終端文字処理無しで 128 文字の文字列データを変数 DataBuf に読み込む。

```
Dim DataBuf As String = Nothing
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

Register = funB + 100
nError = M3Obj.CReadDump(1, 2, 128, Register, DataBuf, -1)
```

2.10.2 CWriteDump

●動作

デバイスの指定したアドレスから、連続したデータを文字列として書き込みます。

●定義

```
Function CWriteDump(
        ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
        ByVal Register As Integer, ByVal WriteVal As String,
        Optional ByVal SuppChar As Short = 0) As FUNVBLib.FunError
```

●パラメータ

UnitNo	定義ファイルで設定した設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの文字列長(1～128) データアクセスは 16 ビット単位で行なわれるため，文字列長が奇数の場合は文字列の最後に SuppChar で指定した文字が追加されます。
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ タイマー設定値とカウンタ設定値への書き込みはできません。
WriteVal	書き込むデータを格納した String 型変数を設定します。 DataNum が 0 の時は，文字列長の分だけデータを書き込みます。 文字列長が DataNum で指定した数より大きい場合は，DataNum 文字だけ書き込まれます。 文字列長が DataNum で指定した数より小さい場合は，SuppChar で指定した文字で不足分が埋められます。
SuppChar	WriteVal の文字列長が DataNum よりも小さい場合に，不足分を埋める文字の文字コードとして 0～255 を指定します。 省略時は 0 になります。

●戻り値

funOK	正常終了
FunInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003～D00007 に”ABCDEFG”を書き込む（不足分は 0 で埋める）。

```
Dim DataBuf As String
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register = funD + 3
DataBuf = “ABCDEFG”
nError = M3Obj.CWriteDump(1, 2, 10, Register, DataBuf)
```

設備 1、CPU2 の B00100～B00163 に”0123456789”を書き込む（不足分はスペースで埋める）。

```
Dim DataBuf As String
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register = funB + 100
DataBuf = “0123456789”
nError = M3Obj.CWriteDump(1, 2, 128, Register, DataBuf, Asc (“ “))
```

2.10.3 CReadDumpEx

●動作

デバイスの指定したアドレスから、連続したデータを文字列として読み込みます。
CReadDump よりも高速ですが、アクセス時のデータの同時性は保障されません。

●定義

```
Function CReadDumpEx(
        ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
        ByVal Register As Integer, ByRef RetVal As String,
        Optional ByVal TermChar As Short = 0) As FUNVBLib.FunError
```

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの文字列長(1～1048576)
Register	データを読み込む先頭レジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ
RetVal	読み込んだデータを格納する String 型変数です。
TermChar	0～255 を指定した場合は，その文字コードに対応する文字を終端文字として，読み込んだ文字列に終端文字がある場合は終端文字より前の文字列を RetVal に返します。 -1 を指定した場合は，終端文字の処理は行なわずに DataNum で指定した全ての文字を返します。 省略時は，Chr$(0)が終端文字となります。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
FunInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003 から，0 を終端文字として最大 53 文字の文字列データを変数 DataBuf に読み込む。

```
Dim DataBuf As String = Nothing
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register = funD + 3
nError = M3Obj.CReadDumpEx(1, 2, 53, Register, DataBuf)
```

設備 1、CPU2 の B00100 から，終端文字処理無しで 1024 文字の文字列データを変数 DataBuf に読み込む。

```
Dim DataBuf As String = Nothing
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj. InitializeUnitTCP(1, “192.168.1.1”)

Register = funB + 100
nError = M3Obj.CReadDumpEx(1, 2, 1024, Register, DataBuf, -1)
```

2.10.4 CWriteDumpEx

●動作

デバイスの指定したアドレスから、連続したデータを文字列として書き込みます。
CWriteDump よりも高速ですが、アクセス時のデータの同時性は保障されません。

●定義

```
Function CWriteDumpEx(
        ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
        ByVal Register As Integer, ByVal WriteVal As String,
        Optional ByVal SuppChar As Short = 0) As FUNVBLib.FunError
```

●パラメータ

UnitNo	定義ファイルで設定した設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの文字列長(1～1048576) データアクセスは 16 ビット単位で行なわれるため，文字列長が奇数の場合は文字列の最後に SuppChar で指定した文字が追加されます。
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ タイマー設定値とカウンタ設定値への書き込みはできません。
WriteVal	書き込むデータを格納した String 型変数を設定します。 DataNum が 0 の時は，文字列長の分だけデータを書き込みます。 文字列長が DataNum で指定した数より大きい場合は，DataNum 文字だけ書き込まれます。 文字列長が DataNum で指定した数より小さい場合は，SuppChar で指定した文字で不足分が埋められます。
SuppChar	WriteVal の文字列長が DataNum よりも小さい場合に，不足分を埋める文字の文字コードとして 0～255 を指定します。 省略時は 0 になります。

●戻り値

funOK	正常終了
FunInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の D00003～D00007 に”ABCDEFG”を書き込む（不足分は 0 で埋める）。

```
Dim DataBuf As String
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register = funD + 3
DataBuf = “ABCDEFG”
nError = M3Obj.CWriteDumpEx(1, 2, 10, Register, DataBuf)
```

設備 1、CPU2 の B00100～B00611 に”0123456789”を書き込む（不足分はスペースで埋める）。

```
Dim DataBuf As String
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register = funB + 100
DataBuf = “0123456789”
nError = M3Obj.CWriteDumpEx(1, 2, 1024, Register, DataBuf, Asc (“ “))
```

2.11　1 ビットデータアクセス関数

2.11.1　BReadDirect

●動作

リレーデバイスからビットデータを読み込みます。

●定義

Function BReadDirect(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByRef Register As System.Array, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～32)
Register	データを読み込むリレーを指定した Integer 型配列変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 1... I リレー　3... E リレー　9... タイムアップリレー 10... カウントアップリレー　14... L リレー　15... X リレー 16... Y リレー　18... M リレー リレーデバイス以外は指定できません。
RetVal	読み込んだデータを格納する Integer 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の I00003 のデータを変数 DataBuf に読み込む。

```
Dim DataBuf() As Integer = Nothing
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

Register(0) = funI + 3
nError = M3Obj.BReadDirect(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の I00003 と E00004 のデータを変数 DataBuf に読み込む。

```
Dim DataBuf() As Integer = Nothing
Dim Register(1) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

Register(0) = funI + 3
Register(1) = funE + 4
nError = M3Obj.BReadDirect(1, 2, 2, Register, DataBuf)
```

2.11.2 BWriteDirect

●動作

リレーデバイスにデータを書き込みます。

●定義

Function BWriteDirect(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByRef Register As System.Array, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～32)
Register	データを書き込むリレーを指定した Integer 型配列変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 1... I リレー　3... E リレー　9... タイムアップリレー 10... カウントアップリレー　14... L リレー　16... Y リレー リレーデバイス以外は指定できません。 配列の長さは DataNum で指定した数以上の長さでなければなりません。
WriteVal	レジスタに書き込むデータを指定した Integer 型配列変数を設定します。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は 0 または 1 を指定します。 0,1 以外を指定したときは、1 として扱われます。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の I00003 に 1 を書き込む。

```
Dim DataBuf(0) As Integer
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

DataBuf(0) = 1
Register(0) = funI + 3
nError = M3Obj.BWriteDirect(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の I00003 に 1、E00005 に 0 を書き込む。

```
Dim DataBuf(1) As Integer
Dim Register(1) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register(0) = funI + 3
Register(1) = funE + 5
DataBuf(0) = 1
DataBuf(1) = 0
nError = M3Obj.BWriteDirect(1, 2, 2, Register, DataBuf)
```

2.11.3 BReadDump

●動作

リレーデバイスの指定したアドレスから、連続したデータを読み込みます。

●定義

Function BReadDump(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～256)
Register	データを読み込む先頭レジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 1... I リレー　3... E リレー　9... タイムアップリレー 10... カウントアップリレー　14... L リレー　15... X リレー 16... Y リレー　18... M リレー リレーデバイス以外は指定できません。
RetVal	読み込んだデータを格納する Integer 型配列です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の I00003～I00102 のデータを変数 DataBuf に読み込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf() As Integer = Nothing
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register = funI + 3
nError = M3Obj.BReadDump(1, 2, 100, Register, DataBuf)
```

2.11.4 BWriteDump

●動作

リレーデバイスの指定したアドレスから、連続したデータを書き込みます。

●定義

Function BWriteDump(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～256)
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 1... I リレー　3... E リレー　9... タイムアップリレー 10... カウントアップリレー　14... L リレー　16... Y リレー リレーデバイス以外は指定できません。
WriteVal	書き込むデータを格納した Integer 型配列です。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は 0 または 1 を指定します。 0,1 以外を指定したときは、1 として扱われます。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の I00003～I00007 に 1,0,1,1,0 を書き込む。

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim DataBuf(4) As Integer
Dim Register As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

Register = funI + 3
DataBuf(0) = 1
DataBuf(1) = 0
DataBuf(2) = 1
DataBuf(3) = 1
DataBuf(4) = 0
nError = M3Obj.BWriteDump(1, 2, 5, Register, DataBuf)
```

2.11.5 BReadDirectStr

●動作

リレーデバイスからビットデータを文字列として読み込みます。
読み込んだデータは"0"か"1"の文字列となって返されます。

●定義

Function BReadDirectStr(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByRef Register As System.Array, ByRef RetVal As String) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～32)
Register	データを読み込むリレーを指定した Integer 型配列変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 1... .I リレー　3... E リレー　9... タイムアップリレー 10...カウントアップリレー　14…Lリレー　15…X リレー 16…Y リレー　18... M リレー リレーデバイス以外は指定できません。
RetVal	読み込んだデータを格納する String 型変数です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の I00003 のデータを変数 DataBuf に読み込む。

```
Dim DataBuf As String = Nothing
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register(0) = funI + 3
nError = M3Obj.BReadDirectStr(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の I00003 と E00004 のデータを変数 DataBuf に読み込む。

```
Dim DataBuf As String = Nothing
Dim Register(1) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

Register(0) = funI + 3
Register(1) = funE + 4
nError = M3Obj.BReadDirectStr(1, 2, 2, Register, DataBuf)
```

2.11.6 BWriteDirectStr

●動作

リレーデバイスにデータを文字列で書き込みます。
書き込むデータは"0"か"1"の文字列で設定します。

●定義

Function BWriteDirectStr(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByRef Register As System.Array, ByVal WriteVal As String) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～32)
Register	データを書き込むリレーを指定した Integer 型配列変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 1... .I リレー　3... E リレー　9... タイムアップリレー 10...カウントアップリレー　14…Lリレー　16…Y リレー リレーデバイス以外は指定できません。 配列の長さは DataNum で指定した数以上の長さでなければなりません。
WriteVal	レジスタに書き込むデータを指定した String 型変数を設定します。 文字列の長さは DataNum で指定した数以上の長さでなければなりません。 値は"0"または"1"の文字列で指定します。 “0”,”1”以外を指定したときは、”1”として扱われます。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunRegType
```

設備 1、CPU2 の I00003 に 1 を書き込む。

```
Dim DataBuf As String
Dim Register(0) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

DataBuf= “1”
Register(0) = funI + 3
nError = M3Obj.BWriteDirectStr(1, 2, 1, Register, DataBuf)
```

設備 1、CPU2 の I00003 に 1、E00005 に 0 を書き込む。

```
Dim DataBuf As String
Dim Register(1) As Integer
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

Register(0) = funI + 3
Register(1) = funE + 5
DataBuf = "10"
nError = M3Obj.BWriteDirectStr(1, 2, 2, Register, DataBuf)
```

2.11.7　BReadDumpStr

●動作

リレーデバイスの指定したアドレスから文字列で連続したデータを読み込みます。
読み込んだデータは"0"か"1"の文字列となって返されます。

●定義

Function BReadDumpStr(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
ByVal Register As Integer, ByRef RetVal As String) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
DataNum	読み込むデータの数(1～256)
Register	データを読み込む先頭レジスタを指定した Long 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 1... I リレー　3... E リレー　9... タイムアップリレー 10... カウントアップリレー　14... L リレー　15... X リレー 16... Y リレー　18... M リレー
RetVal	読み込んだデータを格納する String 型変数です。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の I00003～I00102 のデータを変数 DataBuf に読み込む。

```
プログラムの先頭
    Imports FUNVBLib
    Imports FUNVBLib.FunRegType

以下がコードのサンプル
    Dim DataBuf As String = Nothing
    Dim Register As Integer
    Dim nError As FunError
    Dim M3Obj As FUN = New FUN

    nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

    Register = funI + 3
    nError = M3Obj.BReadDumpStr(1, 2, 100, Register, DataBuf)
```

2.11.8　BWriteDumpStr

●動作

リレーデバイスの指定したアドレスから文字列で連続したデータを書き込みます。
書き込むデータは"0"か"1"の文字列で設定します。

●定義

Function BWriteDumpStr(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByVal WriteVal As String) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(1～256)
Register	データを書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 1... I リレー　3... E リレー　9... タイムアップリレー 10... カウントアップリレー　14... L リレー　16... Y リレー リレーデバイス以外は指定できません。
WriteVal	書き込むデータを格納した String 型変数です。 文字列の長さは DataNum で指定した数以上の長さでなければなりません。 値は”0”または”1”を指定します。 “0”,”1”以外を指定したときは、”1”として扱われます。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 の I00003～I00007 に 1,0,1,1,0 を書き込む。

```
プログラムの先頭
    Imports FUNVBLib
    Imports FUNVBLib.FunRegType

以下がコードのサンプル
    Dim DataBuf As String
    Dim Register As Integer
    Dim nError As FunError
    Dim M3Obj As FUN = New FUN

    nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

    Register = funI + 3
    DataBuf="10110"
    nError = M3Obj.BWriteDumpStr(1, 2, 5, Register, DataBuf)
```

2.12　特殊モジュールアクセス関数

2.12.1　SReadDump

●動作

特殊モジュールの指定したデータ位置から、連続したデータを読み込みます。

●定義

```
Function SReadDump(
        ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal MUnit As Integer,
        ByVal MSlot As Integer, ByVal MChannel As Integer, ByVal DataNum As Integer,
        ByRef RetVal As System.Array) As FUNVBLib.FunError
```

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
MUnit	データを読み込むモジュールユニット番号(0～7)
MSlot	データを読み込むモジュールスロット番号(1～16)
MChanel	データを読み込む先頭データ位置番号
DataNum	読み込むデータの数(1～64)
RetVal	読み込んだデータを格納する Integer 型配列。

●戻り値

funOK	正常終了。RetVal に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2、モジュールユニット番号 0、モジュールスロット番号 9、先頭データ位置番号 5 から 20 点のデータを読み込む。

```
プログラムの先頭
    Imports FUNVBLib

以下がコードのサンプル
    Dim DataBuf() As Integer = Nothing
    Dim nError As FunError
    Dim M3Obj As FUN = New FUN

    nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1")

    nError = M3Obj.SReadDump(1, 2, 0, 9, 5, 20, DataBuf)
```

2.12.2 SWriteDump

●動作

特殊モジュールの指定したデータ位置から、データを書き込みます。

●定義

Function SWriteDump(
ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal MUnit As Integer,
ByVal MSlot As Integer, ByVal MChannel As Integer, ByVal DataNum As Integer,
ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
MUnit	データを書き込むモジュールユニット番号(0～7)
MSlot	データを書き込むモジュールスロット番号(1～16)
MChanel	データを書き込む先頭データ位置番号
DataNum	書き込むデータの数(1～64)
WriteVal	書き込むデータを格納した Integer 型配列です。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は-32768～65535 の範囲で指定します。 詳細は、各モジュールの取扱説明書を参照して下さい。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2、モジュールユニット番号 0、モジュールスロット番号 9、先頭データ位置番号 5 から 1, 2, 3, 4, 5 を書き込む。

```
プログラムの先頭
    Imports FUNVBLib

以下がコードのサンプル
    Dim DataBuf(4) As Integer
    Dim nError As FunError
    Dim M3Obj As FUN = New FUN

    nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

    DataBuf(0) = 1
    DataBuf(1) = 2
    DataBuf(2) = 3
    DataBuf(3) = 4
    DataBuf(4) = 5
    nError = M3Obj.SWriteDump(1, 2, 0, 9, 5, 5, DataBuf)
```

2.12.3 SLReadDump

●動作

特殊モジュールの指定したデータ位置から、2 ワード単位で連続したデータを読み込みます。

●定義

Function SLReadDump(
ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal MUnit As Integer,
ByVal MSlot As Integer, ByVal MChannel As Integer, ByVal DataNum As Integer,
ByRef RetVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを読み込む CPU の番号(1～4)
MUnit	データを読み込むモジュールユニット番号(0～7)
MSlot	データを読み込むモジュールスロット番号(1～16)
MChanel	データを読み込む先頭データ位置番号
DataNum	読み込むデータの数(1～32)
RetVal	読み込んだデータを格納する Integer 型配列。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2、モジュールユニット番号 0、モジュールスロット番号 9、先頭データ位置番号 5 から 20 点のデータを 2 ワード単位で読み込む。

```
プログラムの先頭
    Imports FUNVBLib

以下がコードのサンプル
    Dim DataBuf() As Integer = Nothing
    Dim nError As FunError
    Dim M3Obj As FUN = New FUN

    nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

    nError = M3Obj.SLReadDump(1, 2, 0, 9, 5, 20, DataBuf)
```

2.12.4 SLWriteDump

●動作

特殊モジュールの指定したデータ位置から、2 ワード単位でデータを書き込みます。

●定義

Function SLWriteDump(
ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal MUnit As Integer,
ByVal MSlot As Integer, ByVal MChannel As Integer, ByVal DataNum As Integer,
ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	データを書き込む CPU の番号(1～4)
MUnit	データを書き込むモジュールユニット番号(0～7)
MSlot	データを書き込むモジュールスロット番号(1～16)
MChanel	データを書き込む先頭データ位置番号
DataNum	書き込むデータの数(1～32)
WriteVal	書き込むデータを格納した Integer 型配列です。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は-2147483648～2147483647 の範囲で指定します。 詳細は、各モジュールの取扱説明書を参照して下さい。

●戻り値

funOK	正常終了、RetVal に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2、モジュールユニット番号 0、モジュールスロット番号 9、先頭データ位置番号 5 から 1, 2, 3, 4, 5 を 2 ワード単位で書き込む。

```
プログラムの先頭
    Imports FUNVBLib

以下がコードのサンプル
    Dim DataBuf(4) As Integer
    Dim nError As FunError
    Dim M3Obj As FUN = New FUN

    nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

    DataBuf(0) = 1
    DataBuf(1) = 2
    DataBuf(2) = 3
    DataBuf(3) = 4
    DataBuf(4) = 5
    nError = M3Obj.SLWriteDump(1, 2, 0, 9, 5, 5, DataBuf)
```

2.13　データアクセスに関する注意点

末尾が"Ex"となっているメソッドは、高速アクセスを行うためのコマンドを使用しております。
このコマンドは CPU 直結の通信や、LC11/LC12/LE12 モジュールではサポートされておりません。
また、LE01/LE11 モジュールや SP66/SP67/SP71/SP76 の内蔵 Ethernet ポートを使用している場合でも、データコードがバイナリのときは使用することができません。

高速アクセス関数が使用できるモジュール／データコードの組み合わせは以下の通りとなります。

データコード	Ethernet 通信				シリアル通信	
	LE01	LE11	LE12	SP66/SP67/SP71/SP76	LC11/LC12	CPU 直結
ASCII	○	○		○		
バイナリ					―	―

64 ビットデータアクセス関数は、Visual Basic.NET および Visual C#で使用する為の関数です。
Visual Basic 6.0 では使用できませんので、ご注意ください。

2.14　PLC 情報アクセス関数

2.14.1　GetStatus

●動作

指定した設備、CPU のシーケンスプログラム、BASIC プログラムの動作状態を読み込みます。

●定義

Function GetStatus(ByVal UnitNo As Integer, ByVal CpuNo As Integer,
ByRef CpuStat As Integer, ByRef ProgStat As Integer) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	プログラム状態を読み込む CPU の番号(1～4)
CpuStat	CPU ステータスを受け取る Integer 型変数
ProgStat	プログラムステータスを受け取る Integer 型変数

●戻り値

funOK	正常終了 CpuStat と ProgStat に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

CPU ステータス、プログラムステータスなどの詳細は FA-M3 の IM「パソコンリンクコマンド取扱説明書(IM 34M6P41 –01)」を参照してください。

●例

設備 1、CPU2の CPU モジュール、プログラム状態を読み込む。

プログラムの先頭

```
Imports FUNVBLib
```

以下がコードのサンプル

```
Dim nError As FunError
Dim nCpuStatus As Integer
Dim nProgramStatus As Integer
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

nError = M3Obj.GetStatus(1, 2, nCpuStatus, nProgramStatus)
```

2.14.2 GetSysInfo

●動作

指定した設備、CPU のシステム情報を読み込みます。

●定義

Function GetSysInfo(
ByVal UnitNo As Integer, ByVal CpuNo As Integer,
ByRef SystemID As String, ByRef Revision As String,
ByRef CpuType As Integer, ByRef ProgAreaSize As Integer) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	システム情報を読み込む CPU の番号(1～4)
SystemID	システム ID
Revision	レビジョン
CpuType	CPU タイプ
ProgAreaSize	プログラムエリアサイズ
	（単位） K ステップ :シーケンス CPU
	K バイト :BASIC CPU

●戻り値

funOK	正常終了、読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funCommError	通信エラーが発生
funNotInitialized	指定された設備は初期化されていない
funInvalidInstall	インストールされているファイルが破損している

システム情報の詳細は FA-M3 の IM「パソコンリンクコマンド取扱説明書(IM 34M6P41 –01)」を参照してください。

●例

設備 1、CPU2 のシステム情報を読み込む。

```
プログラムの先頭
    Imports FUNVBLib

以下がコードのサンプル
    Dim nError As FunError
    Dim SystemID As String = Nothing
    Dim Revision As String = Nothing
    Dim CpuType As Integer
    Dim ProgAreaSize As Integer
    Dim M3Obj As FUN = New FUN

    nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

    nError = M3Obj.GetSysInfo(1, 2, SystemID, Revision, CpuType, ProgAreaSize)
```

2.14.3 GetModuleInfo

●動作

指定した設備、CPU、ユニット番号に対して、実装モジュール情報を読み込みます。

●定義

Function GetModuleInfo(
ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal MUnit As Integer,
ByRef ModuleName As System.Array, ByRef IoType As System.Array,
ByRef IoNum As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	モジュール情報を読み込む CPU の番号(1～4)
MUnit	ユニット番号 (0～7)
ModuleName	モジュール名を受け取る配列
IoType	入出力種別を受け取る配列
IoNum	入出力リレー点数を受け取る配列

●戻り値

funOK	正常終了 ModuleName, IoType, IoNum の(0)～(15)に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

実装モジュール情報の詳細は FA-M3 の IM「パソコンリンクコマンド取扱説明書 (IM 34M6P41 –01)」を参照してください。

●例

設備 1、CPU2、ユニット 0 に対してモジュール情報を読み込む。

```
プログラムの先頭
    Imports FUNVBLib

以下がコードのサンプル
    Dim nError As FunError
    Dim ModName() As String = Nothing
    Dim IoType() As Integer = Nothing
    Dim IoNum() As Integer = Nothing
    Dim M3Obj As FUN = New FUN

    nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1")

    nError = M3Obj.GetModuleInfo(1, 2, 0, ModName, IoType, IoNum)
```

2.14.4 GetLedInfo

●動作

指定した設備、CPU の ERR LED、または ALM LED の点灯要因を読み込みます。

●定義

```
Function GetLedInfo(
        ByVal UnitNo As Integer, ByVal CpuNo As Integer,
        ByVal LedType As FUNVBLib.FunLedType,
        ByRef LedInfo As System.Array) As FUNVBLib.FunError
```

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	LED 点灯要因を読み込む CPU の番号(1～4)
LedType	LED タイプ（M3LedErr…ERR 要因指定、M3LedAlm…ALM 要因指定)
LedInfo	LED 点灯要因を格納する Long 型配列変数 ERR 要因指定のときは、長さ 1 の配列が返されます ALM 要因指定のときは、長さ 16 の配列が返されます

●戻り値

funOK	正常終了、LedInfo に読み込んだデータが格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

LED 点灯要因の詳細は FA-M3 の IM「パソコンリンクコマンド取扱説明書(IM 34M6P41 –01)」を参照してください。

●例

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunLedType
```

設備 1、CPU2 に対して ERR LED 点灯要因を読み込む例。

```
Dim nError As FunError
Dim nLedInfo() As Integer = Nothing
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

nError = M3Obj.GetLedInfo(1, 2, M3LedErr, nLedInfo)
```

設備 1、CPU2 に対して ALM LED 点灯要因を読み込む。

```
Dim nError As FunError
Dim nLedInfo() As Integer = Nothing
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

nError = M3Obj.GetLedInfo(1, 2, M3LedAlm, nLedInfo)
```

2.14.5 ResetAlarmInfo

●動作

指定した設備、CPU の現在のアラーム情報を消去します。

●定義

Function ResetAlarmInfo(ByVal UnitNo As Integer, ByVal CpuNo As Integer) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	アラーム情報を消去する CPU の番号(1～4)

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2 に対してアラーム情報の消去を実行する。

プログラムの先頭

```
Imports FUNVBLib
```

以下がコードのサンプル

```
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

nError = M3Obj.ResetAlarmInfo(1, 2)
```

2.14.6 GetDate

●動作

指定した設備、CPU の日付時刻を読み込みます。

●定義

Function GetDate(ByVal UnitNo As Integer, ByVal CpuNo As Integer,
ByRef DateTime As Date) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	日付時刻を読み込む CPU の番号(1～4)
DateTime	日付時刻を読み込む Date 型変数

●戻り値

funOK	正常終了、DateTime に日付時刻が格納されます。
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

設備 1、CPU2の日付時刻を読み込む。

プログラムの先頭

```
Imports FUNVBLib
```

以下がコードのサンプル

```
Dim dt As Date
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitTCP(1, “192.168.1.1”)

nError = M3Obj.GetDate(1, 2, dt)
```

2.14.7 SetDate

●動作

指定した設備、CPU の日付時刻を変更します。

●定義

Function SetDate(ByVal UnitNo As Integer, ByVal CpuNo As Integer,
ByVal DateTime As Date) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	日付時刻を読み込む CPU の番号(1～4)
DateTime	設定する日付時刻を格納した Date 型変数

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funInvalidInstall	インストールされているファイルが破損している

●例

プログラムの先頭

```
Imports FUNVBLib
```

設備 1、CPU2の日付時刻を PC の現在時刻に設定する。

```
Dim dt As Date
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

dt = Now
nError = M3Obj.SetDate(1, 2, dt)
```

設備 1、CPU2の日付時刻を 1999/09/09 09:09:09 に設定する。

```
Dim dt As Date
Dim nError As FunError
Dim M3Obj As FUN = New FUN

nError = M3Obj.InitializeUnitSerial(1, 1, 1, 1, 9600, 8, 0, 1, False, False)

dt = "1999/09/09 09:09:09"
nError = M3Obj.SetDate(1, 2, dt)
```

2.15　イベント制御関数

2.15.1　InitializeEvent

●動作

イベント受信機能を初期化します。

●定義

Function InitializeEvent(ByVal UnitNo As Integer, ByVal Port As Integer) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
nPortNo	イベントを受信するポート番号 5000～49151 の範囲で、同じ PC で動作するほかのアプリケーションで使用されていないポートを指定します。 1 つのポート番号で、複数設備のイベントを受信する事ができます。 使用する通信プロトコルは、設備の初期化時に使用した関数により決まります。 （InitializeUnit の時：UDP、InitializeUnitTCP の時：TCP、InitializeUnitSerial の時：シリアル）

●戻り値

funOK	正常終了 以後、指定したポートでイベントを受信すると FAM3Event が発生します
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funAlreadyUsed	指定された設備はすでに初期化済み
funFailed	初期化に失敗した
funInvalidInstall	インストールされているファイルが破損している
funInvalidCommType	現在選択している通信種別では、使用できない機能

●例

設備 1 を、ポート 5000 でイベントを受信するように初期化します。

プログラムの先頭

```
Imports FUNVBLib
```

クラスの先頭

```
Private WithEvents M3Obj As FUN
```

以下がコードのサンプル

```
Dim nError As FunError
M3Obj = New FUN

nError = M3Obj.InitializeUnit(1, “192.168.1.1”)

nError = M3Obj.InitializeEvent(1, 5000)
```

注意：イベントを利用するときは必ず WithEvents 変数を利用してください。

```
Private WithEvents M3Obj As FUN
```

2.15.2 ReplyEventDirect

●動作

直接指定のイベントに対する応答コマンドの発行を行います。
FA-M3 から PC へのイベント発行を直接指定に設定している場合に使用します。

●定義

Function ReplyEventDirect(ByVal UnitNo As Integer, ByVal Response As Integer) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
nResponse	応答として返す値

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funFailed	イベントを受信した状態ではない
funInvalidInstall	インストールされているファイルが破損している
funInvalidCommType	現在選択している通信種別では、使用できない機能

●例

設備 1 のイベントに対して、2 を応答として発行します。

```
Dim nError As FunError

nError = M3Obj.ReplyEventDirect(1, 2)
```

注意:イベントを利用するときは必ず WithEvents 変数を利用してください。
Private WithEvents M3Obj As FUN

2.15.3 ReplyEventIndirect

●動作

間接指定のイベントに対する応答コマンドを 16 ビット整数で発行します。
FA-M3 から PC へのイベント発行を間接指定に設定している場合に使用します。

●定義

Function ReplyEventIndirect(ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer, ByVal Register As Integer, ByRef WriteVal As System.Array) As FUNVBLib.FunError

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	応答を書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(0～64)
Register	応答を書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0...D レジスタ　1... .I リレー　2... R レジスタ 3... E リレー　4... B レジスタ 5... タイマー現在値（カウントダウン形） 6... カウンタ現在値（カウントダウン形） 7... タイマー現在値（カウントアップ形） 8... カウンタ現在値（カウントアップ形） 9... タイムアップリレー　10... カウントアップリレー　13…Wレジスタ 14…Lリレー　16…Y リレー　19... V レジスタ 20... F レジスタ DataNum が 0 のときは無視されます。
WriteVal	書き込む応答データを格納した Integer 型配列です。 配列の長さは DataNum で指定した数以上の長さでなければなりません。 値は-32768～65535(タイマ,カウンタは 0～32767)の範囲で指定します。 レジスタ ID にリレーを選択した場合は、(アドレス)～(アドレス+15)に (データのビット 1)～(データのビット 16)が書き込まれます。 DataNum が 0 のときは無視されます。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funFailed	イベントを受信した状態ではない
funInvalidInstall	インストールされているファイルが破損している
funInvalidCommType	現在選択している通信種別では、使用できない機能

●例

設備 1 のイベントに対して、CPU2、D00003～D00007 へ 1,2,3,4,5 の書き込み処理を応答として発行します。

```
Dim DataBuf(4) As Integer
Dim Register As Integer
Dim nError As FunError

Register = funD + 3
DataBuf(0) = 1
DataBuf(1) = 2
DataBuf(2) = 3
DataBuf(3) = 4
DataBuf(4) = 5
nError = M3Obj.ReplyEventIndirect (1, 2, 5, Register, DataBuf)
```

注意:イベントを利用するときは必ず WithEvents 変数を利用してください。
Private WithEvents M3Obj As FUN

2.15.4 CReplyEventIndirect

●動作

間接指定のイベントに対する応答コマンドを文字列で発行します。
FA-M3 から PC へのイベント発行を間接指定に設定している場合に使用します。

●定義

```
Function CReplyEventIndirect(
        ByVal UnitNo As Integer, ByVal CpuNo As Integer, ByVal DataNum As Integer,
        ByVal Register As Integer, ByVal WriteVal As String,
        Optional ByVal SuppChar As Short = 0) As FUNVBLib.FunError
```

●パラメータ

UnitNo	初期化済みの設備番号
CpuNo	応答を書き込む CPU の番号(1～4)
DataNum	書き込むデータの数(0～128) データアクセスは 16 ビット単位で行なわれるため，文字列長が奇数の場合は文字列の最後に SuppChar で指定した文字が追加されます。
Register	応答を書き込むレジスタを指定した Integer 型変数を設定します。 レジスタ ID×&H100000 + アドレス の形式で指定します。 レジスタ ID の詳細は以下のとおりです。 0... D レジスタ　2... R レジスタ　4... B レジスタ 13... W レジスタ　20... F レジスタ
WriteVal	書き込む応答データを格納した文字列です。 DataNum が 0 の時は，文字列長の分だけデータを書き込みます。 文字列長が DataNum で指定した数より大きい場合は，DataNum 文字だけ書き込まれます。 文字列長が DataNum で指定した数より小さい場合は，SuppChar で指定した文字で不足分が埋められます。
SuppChar	WriteVal の文字列長が DataNum よりも小さい場合に，不足分を埋める文字の文字コードとして 0～255 を指定します。 省略時は 0 になります。

●戻り値

funOK	正常終了
funInvalidParameter	パラメータの値が不正
funNotInitialized	指定された設備は初期化されていない
funCommError	通信エラーが発生
funFailed	イベントを受信した状態ではない
funInvalidInstall	インストールされているファイルが破損している
funInvalidCommType	現在選択している通信種別では、使用できない機能

●例

設備 1 のイベントに対して、CPU2、D00003～D00007 へ”ABCDEFG”を書き込む処理（不足分は 0 で埋める）を応答として発行

```
Dim DataBuf As String
Dim Register As Integer
Dim nError As FunError

Register = funD + 3
DataBuf = “ABCDEFG”
nError = M3Obj.CReplyEventIndirect (1, 2, 10, Register, DataBuf)
```

設備 1 のイベントに対して、CPU2、B00100～B00163 へ”0123456789”を書き込む処理（不足分はスペースで埋める）を応答として発行

```
Dim DataBuf As String
Dim Register As Integer
Dim nError As FunError

Register = funB + 100
DataBuf = “0123456789”
nError = M3Obj.CReplyEventIndirect (1, 2, 128, Register, DataBuf, Asc (" "))
```

注意：イベントを利用するときは必ず WithEvents 変数を利用してください。

```
Private WithEvents M3Obj As FUN
```

2.16　イベント

2.16.1　FAM3Event

●動作

FA-M3 から受信した TCP/UDP イベントを通知します。

SetEventMode の設定が funEventInteger のときは，このイベントが発生します。

イベントを受信したら、ReplyEventDirect，ReplyEventIndirect，CReplyEventIndirect のいずれかで FA-M3 に応答を返してください。

●定義

Sub FAM3Event(ByVal UnitNo As Integer, ByVal DataNum As Integer, ByRef EventData As System.Array)

●パラメータ

UnitNo	イベントを受信した設備番号
DataNum	受信したデータのサイズ
EventData	受信したイベントデータを格納したバッファ

●例

イベントを受信し、DataNum が 0 のときは直接指定で、それ以外のときは間接指定で応答を発行する。

```
Private Sub M3Obj_FAM3Event(ByVal UnitNo As Integer, ByVal DataNum As Integer, ByRef EventData As System.Array)
Handles M3Obj.FAM3Event

    Dim nError As FunError
    Dim Register As Integer
    Dim DataBuf (1) As Integer

    If DataNum = 0 Then
        nError = M3Obj.ReplyEventDirect(UnitNo, 0)
    Else
        Register = funD + 1
        DataBuf (0) = 1
        DataBuf (1) = 2
        nError = M3Obj.ReplyEventIndirect(UnitNo, 1, 2, Register, DataBuf)
    End If
End Sub
```

注意 1：イベントを利用するときは必ず WithEvents 変数を利用してください。

例） Private WithEvents M3Obj As FUN

注意 2：Visual Basic .NET を使用する場合、

イベントの関数の中からフォーム上のコントロール（テキストボックスなど）の値に対してアクセスする場合は、

FUN のオブジェクトを宣言したフォームで

"Control.CheckForIllegalCrossThreadCalls=False"

という命令文を実行してください。

例）

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    Control.CheckForIllegalCrossThreadCalls = False
End Sub
```

●注意

シリアル通信はサポートしていません。

2.16.2　FAM3EventChar

●動作

FA-M3 から受信したイベントを通知します。

SetEventMode の設定が funEventChar のときは，このイベントが発生します。

イベントを受信したら、ReplyEventDirect，ReplyEventIndirect，CReplyEventIndirect のいずれかで FA-M3 に応答を返してください。

通信種別がシリアル通信のときは、応答を返す必要はありません。

●定義

Sub FAM3EventChar(ByVal UnitNo As Integer, ByVal DataNum As Integer, ByVal EventData As String)

●パラメータ

UnitNo	イベントを受信した設備番号
DataNum	受信した文字列データの長さ
EventData	受信したイベントデータを格納した文字列

●例

イベントを受信し、EventData が”ABC”のときは直接指定で、それ以外のときは間接指定で応答を発行する。

```
Private Sub M3Obj_FAM3EventChar(ByVal UnitNo As Integer, ByVal DataNum As Integer, ByVal EventData As String) Handles
M3Obj.FAM3EventChar

    Dim nError As FunError
    Dim Register As Integer
    Dim DataBuf (1) As Integer

    If EventData = “ABC” Then
        nError = M3Obj.ReplyEventDirect(UnitNo, 0)
    Else
        Register = funD + 1
        DataBuf (0) = 1
        DataBuf (1) = 2
        nError = M3Obj.ReplyEventIndirect(UnitNo, 1, 2, Register, DataBuf)
    End If
End Sub
```

注意 1：イベントを利用するときは必ず WithEvents 変数を利用してください。

例） Private WithEvents M3Obj As FUN

注意 2：Visual Basic .NET を使用する場合、

イベントの関数の中からフォーム上のコントロール（テキストボックスなど）の値に対してアクセスする場合は、

FUN のオブジェクトを宣言したフォームで

"Control.CheckForIllegalCrossThreadCalls=False"

という命令文を実行してください。

例） Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Control.CheckForIllegalCrossThreadCalls = False

End Sub

●注意

シリアル通信の場合、FA-M3 から送信されるイベント送信データには、必ずヘッダ・フッタを付加してください。

詳しくは、FA-M3 の「パソコンリンクモジュール取扱説明書」の、A4.イベント送信機能の章をご覧ください。

また、文字列の先頭 2 文字は"OK"，"ER"以外の文字列にする必要があります。

これらの文字列を使用すると、パソコンリンクコマンドの応答と間違えて、誤動作を起こします。

2.17　イベント関連のメソッド／イベントに関する注意点

イベント関連のメソッドやイベントは、ご使用になる通信方法、モジュールによっては、ご利用できない機能があります。
ご注意ください。

	Ethernet			シリアル	
	LE01	LE11/LE12	SP66/SP67/SP71/SP76	LC11/LC12	CPU 直結
InitializeEvent	○			○	
ReplyEventDirect	○				
ReplyEventIndirect	○				
CReplyEventIndirect	○				

また、Visual Basic .NET を使用していて、
イベントの関数の中からフォーム上のコントロール（テキストボックスなど）の値に対してアクセスする場合は、FUN のオブジェクトを宣言したフォームで

```
Control.CheckForIllegalCrossThreadCalls=False
```

という命令文を実行してください。

例）フォームがロードされたときに設定を行う

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    Control.CheckForIllegalCrossThreadCalls = False
End Sub
```

また同様に、Visual C#を使用していて、
イベントの関数の中からフォーム上のコントロールに対してアクセスする場合は、
FUN のオブジェクトを宣言したフォームで

```
CheckForIllegalCrossThreadCalls = false;
```

という命令文を実行してください。

2.18　マルチスレッド／プロセスで FUN を利用する場合の注意点

マルチスレッド／プロセスで FUN を利用する場合には、以下の点にご注意ください。

- 設備の初期化(InitializeUnit)、および解放(UninitializeUnit)の処理が、複数のスレッド、またはプロセスで重複しないようにして下さい。
- できれば、設備の初期化(InitializeUnit)、および解放(UninitializeUnit)の処理は、いずれか1つのスレッド、またはプロセスで行うことをお勧めいたします。

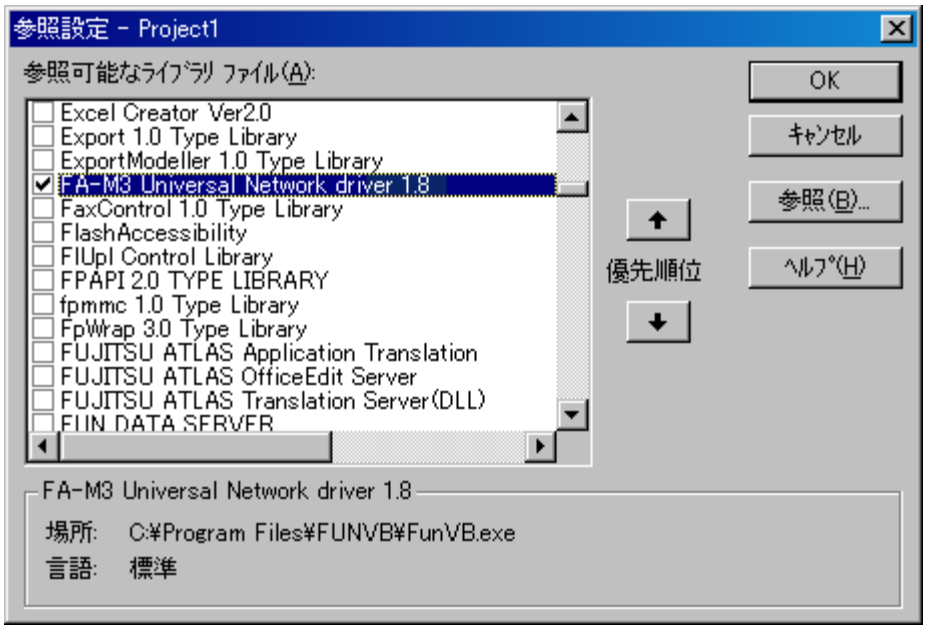
2.19　64 ビット OS 上で動作する場合の注意点

本製品を 64 ビット OS 上で動作する場合は、WOW 64 サブシステム (32 ビットエミュレーター) 上で動作します。
本製品を使用するアプリケーションは 64 ビット OS 上で動作させる場合でも、必ず 32 ビットアプリケーションとして作成してください。

2.20　Visual Basic 6.0 で FUN を利用するための準備

Visual Basic 6.0 で FUN を使用するためには、FUN を使用するコンピュータにインストールをする必要があります。詳しくは FUN インストール解説書、第2章 “FUN をインストールする”を参照して下さい。

続いて，Visual Basic のメニュー「プロジェクト」から「参照設定」を選択して，「FA-M3 Unuversal Network driver」にチェックを付けて、OK を押します。



この作業により，Visual Basic のプロジェクトで FUN を使用できるようになります。

2.21 Visual Basic 6.0 で FUN を利用する場合の注意点

Visual Basic 6.0 で FUN を利用する場合には、以下の点にご注意ください。

- 本ドキュメント中のプログラム例は、Visual Basic.NET 向けに書かれており、そのままでは Visual Basic 6.0 では動作いたしません。
  以下に、3 ページに書かれているサンプルプログラムを例に、その違いを示します。
  ご理解の上、ご利用ください。

Visual Basic.NET の場合:

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunError
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim M3Obj As FUN = New FUN
Dim Register(0) As Integer
Dim DataBuf  () As Integer = Nothing
Dim nError As FunError

nError = M3Obj.InitializeUnit(1, "192.168.1.1")
If nError = funOK Then
    Register(0) = funD + 1
    nError = M3Obj.ReadDirect(1, 1, 1, Register, DataBuf)
    If nError = funOK Then
        TextBox1.Text = DataBuf(0)
    End If
    nError = M3Obj.UninitializeUnit(1)
End If
```

Visual Basic 6.0 の場合:

プログラムの先頭

```
' Imports は不要
```

コードのサンプル

```
Dim M3Obj As FUN                ' New は不要
Set M3Obj = New FUN             ' Set ステートメントでオブジェクトを設定
Dim Register(0) As Long         ' Integer でなく Long で宣言
Dim DataBuf() As Long           ' Integer でなく Long で宣言, = Nothing は不要
Dim nError As FunError

nError = M3Obj.InitializeUnit(1, "192.168.1.1")
If nError = funOK Then
    Register(0) = funD + 1
    nError = M3Obj.ReadDirect(1, 1, 1, Register, DataBuf)
    If nError = funOK Then
        TextBox1.Text = DataBuf(0)
    End If
    nError = M3Obj.UninitializeUnit(1)
End If
```

は相違点

2.22　Visual C#で FUN を利用するための準備

手順は Visual Basic.NET の場合と変わりません。
2.1 を参照してください。

2.23　Visual C#で FUN を利用する場合の注意点

Visual C#で FUN を利用する場合には、以下の点にご注意ください。

- 本ドキュメント中のプログラム例は、Visual Basic .NET 向けに書かれており、そのままでは Visual C#では動作いたしません。
  以下に、3 ページに書かれているサンプルプログラムを例に、その違いを示します。
  ご理解の上、ご利用ください。

Visual Basic.NET の場合:

プログラムの先頭

```
Imports FUNVBLib
Imports FUNVBLib.FunError
Imports FUNVBLib.FunRegType
```

以下がコードのサンプル

```
Dim M3Obj As FUN = New FUN
Dim Register(0) As Integer
Dim DataBuf  () As Integer = Nothing
Dim nError As FunError

nError = M3Obj.InitializeUnit(1, "192.168.1.1")
If nError = funOK Then
    Register(0) = funD + 1
    nError = M3Obj.ReadDirect(1, 1, 1, Register, DataBuf)
    If nError = funOK Then
        TextBox1.Text = DataBuf(0)
    End If
    nError = M3Obj.UninitializeUnit(1)
End If
```

Visual C#の場合:

プログラムの先頭

```
using FUNVBLib;
```

以下がコードのサンプル

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
if (nError == FunError.funOK)
{
    int[] Register = new int[] { (int)FunRegType.funD + 1 };
    System.Array arrRegister = Register;
    System.Array arrDataBuf;
    nError = M3Obj.ReadDirect(1, 1, 1, ref arrRegister, out arrDataBuf);
    if (nError == FunError.funOK)
    {
        int[] DataBuf = (int[])arrDataBuf;
        TextBox1.Text = DataBuf[0].ToString();
    }
    nError = M3Obj.UninitializeUnit(1);
}
```

・ Visual C#ではパラメータの省略指定はできません。したがって本文中パラメータ省略時のデフォルト動作の記述のある以下のメソッドでは、必ず全てのパラメータを指定する必要があります。

InitializeUnitSerial
CReadDump(Ex)
CWriteDump(Ex)
CReplyEventIndirect

・ Visual C#では、FA-M3 からのイベントを受信する場合は、以下のようにしてイベントハンドラを登録します。通常 FUN オブジェクト作成直後に行います。

```
M3Obj.FAM3Event += new _IFUNEvents_FAM3EventEventHandler(M3Obj_FAM3Event);
M3Obj.FAM3EventChar += new
    _IFUNEvents_FAM3EventCharEventHandler(M3Obj_FAM3EventChar);
```

参考までに本文中の各メソッドの使用例を Visual C#用に置き換えたコードを示します。ただし例が複数ある場合は最後のものだけを示します。

InitializeUnit

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
```

InitializeUnitTCP

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
```

InitializeUnitSerial

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
```

UninitializeUnit

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
nError = M3Obj.UninitializeUnit(1);
```

SetTimeout

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
nError = M3Obj.SetTimeout(1, 1000);
```

SetRetryCounter

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
nError = M3Obj.SetRetryCounter(1, 5);
```

SetSign

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
nError = M3Obj.SetSign(1, FunSign.funUnsigned);
```

SetCommMode

```
FUN M3Obj = new FUN();
int mode = (int)FunMode.funModeASCII + (int)FunMode.funModeDisableSeq + (int)FunMode.funModeChangePort;
FunError nError = M3Obj.SetCommMode(1, (FunMode)mode);
nError = M3Obj.InitializeUnit(1, "192.168.1.1");
```

SetEventMode

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
nError = M3Obj.SetEventMode(1, (short)FunEventMode.funEventChar);
```

ReadDirect

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int[] Register = new int[] { (int)FunRegType.funD + 3, (int)FunRegType.funI + 4 };
System.Array arrRegister = Register;
System.Array arrDataBuf;
nError = M3Obj.ReadDirect(1, 2, 2, ref arrRegister, out arrDataBuf);
```

WriteDirect

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int[] Register = new int[] { (int)FunRegType.funD + 3, (int)FunRegType.funR + 5 };
int[] DataBuf = new int[] { 4, 6 };
System.Array arrRegister = Register;
System.Array arrDataBuf = DataBuf;
nError = M3Obj.WriteDirect(1, 2, 2, ref arrRegister, ref arrDataBuf);
```

ReadDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
int Register = (int)FunRegType.funD + 3;
System.Array arrDataBuf;
nError = M3Obj.ReadDump(1, 2, 53, Register, out arrDataBuf);
```

WriteDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
int[] DataBuf = new int[] { 1, 2, 3, 4 };
System.Array arrDataBuf = DataBuf;
nError = M3Obj.WriteDump(1, 2, 4, Register, ref arrDataBuf);
```

ReadDumpEx

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
System.Array arrDataBuf;
nError = M3Obj.ReadDumpEx(1, 2, 1000, Register, out arrDataBuf);
```

WriteDumpEx

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
int[] DataBuf = new int[] { 1, 2, 3, 4 };
System.Array arrDataBuf = DataBuf;
nError = M3Obj.WriteDumpEx(1, 2, 4, Register, ref arrDataBuf);
```

LReadDirect

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int[] Register = new int[] { (int)FunRegType.funD + 3, (int)FunRegType.funI + 4 };
System.Array arrRegister = Register;
System.Array arrDataBuf;
nError = M3Obj.LReadDirect(1, 2, 2, ref arrRegister, out arrDataBuf);
```

LWriteDirect

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
int[] Register = new int[] { (int)FunRegType.funD + 3, (int)FunRegType.funR + 5 };
int[] DataBuf = new int[] { 0x12345678, 0x567890ab };
System.Array arrRegister = Register;
System.Array arrDataBuf = DataBuf;
nError = M3Obj.LWriteDirect(1, 2, 2, ref arrRegister, ref arrDataBuf);
```

LReadDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
System.Array arrDataBuf;
nError = M3Obj.LReadDump(1, 2, 30, Register, out arrDataBuf);
```

LWriteDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
int[] DataBuf = new int[] { 1, 2, 3, 4, 5 };
System.Array arrDataBuf = DataBuf;
nError = M3Obj.LWriteDump(1, 2, 5, Register, ref arrDataBuf);
```

LReadDumpEx

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
System.Array arrDataBuf;
nError = M3Obj.LReadDumpEx(1, 2, 1000, Register, out arrDataBuf);
```

LWriteDumpEx

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
int[] DataBuf = new int[] { 1, 2, 3, 4, 5 };
System.Array arrDataBuf = DataBuf;
nError = M3Obj.LWriteDumpEx(1, 2, 5, Register, ref arrDataBuf);
```

HReadDirect

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int[] Register = new int[] { (int)FunRegType.funD + 3, (int)FunRegType.funI + 4 };
System.Array arrRegister = Register;
System.Array arrDataBuf;
nError = M3Obj.HReadDirect(1, 2, 2, ref arrRegister, out arrDataBuf);
```

HWriteDirect

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
int[] Register = new int[] { (int)FunRegType.funD + 3, (int)FunRegType.funR + 5 };
long[] DataBuf = new long[] { 0x123456789abcdef0, 0x567890abcdef1234 };
System.Array arrRegister = Register;
System.Array arrDataBuf = DataBuf;
nError = M3Obj.HWriteDirect(1, 2, 2, ref arrRegister, ref arrDataBuf);
```

HReadDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
System.Array arrDataBuf;
nError = M3Obj.HReadDump(1, 2, 15, Register, out arrDataBuf);
```

HWriteDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
long[] DataBuf = new long[] { 1, 2, 3, 4, 5 };
System.Array arrDataBuf = DataBuf;
nError = M3Obj.HWriteDump(1, 2, 5, Register, ref arrDataBuf);
```

HReadDumpEx

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
System.Array arrDataBuf;
nError = M3Obj.HReadDumpEx(1, 2, 1000, Register, out arrDataBuf);
```

HWriteDumpEx

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
long[] DataBuf = new long[] { 1, 2, 3, 4, 5 };
System.Array arrDataBuf = DataBuf;
nError = M3Obj.HWriteDumpEx(1, 2, 5, Register, ref arrDataBuf);
```

FReadDirect

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int[] Register = new int[] { (int)FunRegType.funD + 3, (int)FunRegType.funI + 4 };
System.Array arrRegister = Register;
System.Array arrDataBuf;
nError = M3Obj.FReadDirect(1, 2, 2, ref arrRegister, out arrDataBuf);
```

FWriteDirect

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
int[] Register = new int[] { (int)FunRegType.funD + 3, (int)FunRegType.funR + 5 };
float[] DataBuf = new float[] { 1.23456f, 5.67890f };
System.Array arrRegister = Register;
System.Array arrDataBuf = DataBuf;
nError = M3Obj.FWriteDirect(1, 2, 2, ref arrRegister, ref arrDataBuf);
```

FReadDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
System.Array arrDataBuf;
nError = M3Obj.FReadDump(1, 2, 30, Register, out arrDataBuf);
```

FWriteDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
float[] DataBuf = new float[] { 1.11f, 2.22f, 3.33f, 4.44f, 5.55f };
System.Array arrDataBuf = DataBuf;
nError = M3Obj.FWriteDump(1, 2, 5, Register, ref arrDataBuf);
```

FReadDumpEx

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
System.Array arrDataBuf;
nError = M3Obj.FReadDumpEx(1, 2, 300, Register, out arrDataBuf);
```

FWriteDumpEx

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
float[] DataBuf = new float[] { 1.11f, 2.22f, 3.33f, 4.44f, 5.55f };
System.Array arrDataBuf = DataBuf;
nError = M3Obj.FWriteDumpEx(1, 2, 5, Register, ref arrDataBuf);
```

DReadDirect

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int[] Register = new int[] { (int)FunRegType.funD + 3, (int)FunRegType.funI + 4 };
System.Array arrRegister = Register;
System.Array arrDataBuf;
nError = M3Obj.DReadDirect(1, 2, 2, ref arrRegister, out arrDataBuf);
```

DWriteDirect

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
int[] Register = new int[] { (int)FunRegType.funD + 3, (int)FunRegType.funR + 5 };
double[] DataBuf = new double[] { 1.23456d, 5.67890d };
System.Array arrRegister = Register;
System.Array arrDataBuf = DataBuf;
nError = M3Obj.DWriteDirect(1, 2, 2, ref arrRegister, ref arrDataBuf);
```

DReadDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
System.Array arrDataBuf;
nError = M3Obj.DReadDump(1, 2, 15, Register, out arrDataBuf);
```

DWriteDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
double[] DataBuf = new double[] { 1.11d, 2.22d, 3.33d, 4.44d, 5.55d };
System.Array arrDataBuf = DataBuf;
nError = M3Obj.DWriteDump(1, 2, 5, Register, ref arrDataBuf);
```

DReadDumpEx

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
System.Array arrDataBuf;
nError = M3Obj.DReadDumpEx(1, 2, 300, Register, out arrDataBuf);
```

DWriteDumpEx

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int Register = (int)FunRegType.funD + 3;
double[] DataBuf = new double[] { 1.11d, 2.22d, 3.33d, 4.44d, 5.55d };
System.Array arrDataBuf = DataBuf;
nError = M3Obj.DWriteDumpEx(1, 2, 5, Register, ref arrDataBuf);
```

CReadDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
int Register = (int)FunRegType.funB + 100;
string DataBuf = "";
short termChar = -1;
nError = M3Obj.CReadDump(1, 2, 128, Register, out DataBuf, termChar);
```

CWriteDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funB + 100;
string DataBuf = "0123456789";
char suppChar = ' ';
nError = M3Obj.CWriteDump(1, 2, 128, Register, DataBuf, (short)suppChar);
```

CReadDumpEx

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int Register = (int)FunRegType.funB + 100;
string DataBuf = "";
short termChar = -1;
nError = M3Obj.CReadDumpEx(1, 2, 1024, Register, out DataBuf, termChar);
```

CWriteDumpEx

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int Register = (int)FunRegType.funB + 100;
string DataBuf = "0123456789";
char suppChar = ' ';
nError = M3Obj.CWriteDumpEx(1, 2, 1024, Register, DataBuf, (short)suppChar);
```

BReadDirect

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
int[] Register = new int[] { (int)FunRegType.funI + 3, (int)FunRegType.funE + 4 };
System.Array arrRegister = Register;
System.Array arrDataBuf;
nError = M3Obj.BReadDirect(1, 2, 2, ref arrRegister, out arrDataBuf);
```

BWriteDirect

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int[] Register = new int[] { (int)FunRegType.funI + 3, (int)FunRegType.funE + 5 };
int[] DataBuf = new int[] { 1, 0 };
System.Array arrRegister = Register;
System.Array arrDataBuf = DataBuf;
nError = M3Obj.BWriteDirect(1, 2, 2, ref arrRegister, ref arrDataBuf);
```

BReadDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int Register = (int)FunRegType.funI + 3;
System.Array arrDataBuf;
nError = M3Obj.BReadDump(1, 2, 100, Register, out arrDataBuf);
```

BWriteDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
int Register = (int)FunRegType.funI + 3;
int[] DataBuf = new int[] { 1, 0, 1, 1, 0 };
System.Array arrDataBuf = DataBuf;
nError = M3Obj.BWriteDump(1, 2, 5, Register, ref arrDataBuf);
```

BReadDirectStr

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int[] Register = new int[] { (int)FunRegType.funI + 3, (int)FunRegType.funE + 4 };
System.Array arrRegister = Register;
string DataBuf;
nError = M3Obj.BReadDirectStr(1, 2, 2, ref arrRegister, out DataBuf);
```

BWriteDirectStr

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int[] Register = new int[] { (int)FunRegType.funI + 3, (int)FunRegType.funE + 5 };
string DataBuf = "10";
System.Array arrRegister = Register;
nError = M3Obj.BWriteDirectStr(1, 2, 2, ref arrRegister, DataBuf);
```

BReadDumpStr

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
int Register = (int)FunRegType.funI + 3;
string DataBuf;
nError = M3Obj.BReadDumpStr(1, 2, 100, Register, out DataBuf);
```

BWriteDumpStr

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
int Register = (int)FunRegType.funI + 3;
string DataBuf = "10110";
nError = M3Obj.BWriteDumpStr(1, 2, 5, Register, DataBuf);
```

SReadDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
System.Array arrDataBuf;
nError = M3Obj.SReadDump(1, 2, 3, 4, 5, 20, out arrDataBuf);
```

SWriteDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
int[] DataBuf = new int[] { 1, 2, 3, 4, 5 };
System.Array arrDataBuf = DataBuf;
nError = M3Obj.SWriteDump(1, 2, 3, 4, 5, 5, ref arrDataBuf);
```

SLReadDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
System.Array arrDataBuf;
nError = M3Obj.SLReadDump(1, 2, 3, 4, 5, 20, out arrDataBuf);
```

SLWriteDump

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
int[] DataBuf = new int[] { 1, 2, 3, 4, 5 };
System.Array arrDataBuf = DataBuf;
nError = M3Obj.SLWriteDump(1, 2, 3, 4, 5, 5, ref arrDataBuf);
```

GetStatus

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
int nCpuStatus;
int nProgramStatus;
nError = M3Obj.GetStatus(1, 2, out nCpuStatus, out nProgramStatus);
```

GetSysInfo

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
string SystemID;
string Revision;
int CpuType;
int ProgAreaSize;
nError = M3Obj.GetSysInfo(1, 2, out SystemID, out Revision, out CpuType, out ProgAreaSize);
```

GetModuleInfo

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
System.Array arrModName;
System.Array arrIoType;
System.Array arrIoNum;
nError = M3Obj.GetModuleInfo(1, 2, 0, out arrModName, out arrIoType, out arrIoNum);
```

GetLedInfo

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
System.Array arrLedInfo;
nError = M3Obj.GetLedInfo(1, 2, FunLedType.M3LedAlm, out arrLedInfo);
```

ResetAlarmInfo

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
nError = M3Obj.ResetAlarmInfo(1, 2);
```

GetDate

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitTCP(1, "192.168.1.1");
System.DateTime dt;
nError = M3Obj.GetDate(1, 2, out dt);
```

SetDate

```
FUN M3Obj = new FUN();
FunError nError = M3Obj.InitializeUnitSerial(1, 1, FunComm.funCommLC, 1, 9600, 8, FunParity.funNoParity, 1, false, false);
System.DateTime dt;
dt = DateTime.ParseExact("1999/09/09 09:09:09", "yyyy/MM/dd HH:mm:ss", null);
nError = M3Obj.SetDate(1, 2, dt);
```

InitializeEvent

```
M3Obj = new FUN();
M3Obj.FAM3Event += new _IFUNEvents_FAM3EventEventHandler(M3Obj_FAM3Event);
M3Obj.FAM3EventChar += new _IFUNEvents_FAM3EventCharEventHandler(M3Obj_FAM3EventChar);

FunError nError = M3Obj.InitializeUnit(1, "192.168.1.1");
nError = M3Obj.InitializeEvent(1, 5000);
```

ReplyEventDirect

```
FunError nError = M3Obj.ReplyEventDirect(1, 2);
```

ReplyEventIndirect

```
int Register = (int)FunRegType.funD + 3;
int[] DataBuf = new int[] { 1, 2, 3, 4, 5 };
System.Array arrDataBuf = DataBuf;
FunError nError = M3Obj.ReplyEventIndirect(1, 2, 5, Register, ref arrDataBuf);
```

CReplyEventIndirect

```
int Register = (int)FunRegType.funB + 100;
string DataBuf = "0123456789";
short suppChar = (short)' ';
FunError nError = M3Obj.CReplyEventIndirect(1, 2, 128, Register, DataBuf, suppChar);
```

FAM3Event

```
private void M3Obj_FAM3Event(int UnitNo, int DataNum, ref Array EventData)
{
    FunError nError;
    if (DataNum == 0)
    {
        nError = M3Obj.ReplyEventDirect(UnitNo, 0);
    }
    else
    {
        int Register = (int)FunRegType.funD + 1;
        int[] DataBuf = new int[] { 1, 2 };
        System.Array arrDataBuf = DataBuf;
        nError = M3Obj.ReplyEventIndirect(UnitNo, 1, 2, Register, ref arrDataBuf);
    }
}
```

FAM3EventChar

```
private void M3Obj_FAM3EventChar(int UnitNo, int DataNum, string EventData)
{
    FunError nError;
    if (EventData.Equals("ABC"))
    {
        nError = M3Obj.ReplyEventDirect(UnitNo, 0);
    }
    else
    {
        int Register = (int)FunRegType.funD + 1;
        int[] DataBuf = new int[] { 1, 2 };
        System.Array arrDataBuf = DataBuf;
        nError = M3Obj.ReplyEventIndirect(UnitNo, 1, 2, Register, ref arrDataBuf);
    }
}
```

3 定数一覧

FUN では，以下の定数が定義されています。

クラス	定数名	値	意味
FunLedType	M3LedErr	1	ERR 要因指定
	M3LedAlm	2	ALM 要因指定
FunSign	funUnsigned	0	符号無し(0～65535)
	funSigned	1	符号あり(-32768～32767)
FunMode	funModeASCII	0	データコード=ASCII で通信を行います
	funModeBinary	1	データコード=バイナリで通信を行います
	funModeDisableSeq	2	拡張上位リンクサービスを使用しません
	funModeChangePort	4	接続するポート番号を 12291 にします
FunComm	funCommCPU	0	CPU モジュール直結
	funCommLC	1	パソコンリンクモジュール
FunParity	funNoParity	0	パリティ無し
	funOddParity	1	奇数
	funEvenParity	2	偶数
FunRegType	funREG	&H0100000	-
	funD	&H0000000	D レジスタ
	funI	&H0100000	I リレー
	funR	&H0200000	R レジスタ
	funE	&H0300000	E リレー
	funB	&H0400000	B レジスタ
	funTP	&H0500000	タイマー現在値（カウントダウン形）
	funCP	&H0600000	カウンタ現在値（カウントダウン形）
	funTI	&H0700000	タイマー現在値（カウントアップ形）
	funCI	&H0800000	カウンタ現在値（カウントアップ形）
	funTU	&H0900000	タイムアップリレー
	funCU	&H0a00000	カウントアップリレー
	funTS	&H0b00000	タイマー設定値
	funCS	&H0c00000	カウンタ設定値
	funW	&H0d00000	W レジスタ
	funL	&H0e00000	L リレー
	funX	&H0f00000	X リレー
	funY	&H1000000	Y リレー
	funZ	&H1100000	Z レジスタ
	funM	&H1200000	M リレー
	funV	&H1300000	V レジスタ
	funF	&H1400000	F レジスタ
FunEventMode	funEventInteger	-1	16 ビット整数でイベントを通知
	funEventChar	-2	文字列でイベントを通知（終端文字処理なし）

4　エラーコード一覧

FUN では，以下のエラーコードが定義されています。

定義文字列	番号	意味
funOK	0	正常終了
funInvalidParameter	1	不正なパラメータが指定されました
funNotInitialized	2	指定された設備は初期化されていません
funCommError	3	通信エラーが発生しました
funAlreadyUsed	4	指定された設備はすでに初期化済みです
funFailed	5	関数の実行に失敗しました
funInvalidInstall	6	インストールされているファイルが破損しています
funInvalidCommType	7	現在選択している通信種別では、使用できない機能です
funCannotOpen	8	通信ポートがオープンできません

空白ページ


TOKYO DENSAN