# Modbus Master ActiveX Control
# 取扱説明書

NM－9226－B 改1

MG CO., LTD.
www.mgco.jp

本書は、Modbus Master ActiveX Control の操作のための簡潔な取扱説明書です。

## コントロール概要

ModMaster.ocx は、32bit カスタムコントロールを含んでおり、Visual Basic などの OLE コンテナアプリケーション上で、PC に接続された Modbus スレーブデバイスのデータに迅速にかつ容易にアクセスすることを可能にします。 Modbus デバイスは PC COM ポート接続か、または Modbus / TCP にてアクセスできます。 接続は、種々の COM ポートや複数の Modbus ネットワークで複数のデバイスに接続でき、非同期動作が可能です。

modMaster コントロールは、物理的接続（例えばボーレートやパリティー）を定義するプロパティー、および COM ポートやネットワークと接続するためのメソッドを提供します。 接続設立時、個々の接続は、コントロールから戻されたユニークなハンドル名により識別されています。
アプリケーションは、PollModbus や WriteModbus メソッドを使って、デバイスにメッセージ（Read / Write）を送ります。 すべてのメッセージは、バックグランド上でコントロールによって処理されます。文字列がフォーマットされ、選択されたデバイスに送られる間、アプリケーションロジックが続けられます。コントロールはすべてのレスポンスメッセージを処理し、適切なチェックサムを確認し、そして SlaveResponse イベントでメッセージ送受信の結果をアプリケーションに伝えます。 SlaveResponse イベントを処理するとすぐに、アプリケーションは、Read / Write リクエストの結果を得ることができます。

## RegSvr32

RegSvr32.exe は、Windows レジストリーに追加や変更をするためにマイクロソフトから供給された DOS アプリケーションです。 modMaster コントロールをアプリケーションで使う前に、これをレジストリーに適切にインストールして下さい。 RegSvr32.exe を実行し、フルパスのコントロール名を登録する必要があります。 以下に登録例を示します。（例えば modMaster.ocx のファイルが C:\Modbus ディレクトリに含まれている場合）

Regsvr32 c:\modbus\modMaster.ocx

## Modbus Master コントロール説明

Modbus Master コントロールは、プロパティーとメソッドで成り立っています。プロパティーは、Modbus ネットワークと接続するための操作の特徴を定義するもので、メソッドは、接続されたスレーブデバイスからのレスポンスメッセージを初期化したり読んだりするものです。

このコントロールは、以下の Modbus メッセージタイプをサポートします：

**01.** コイルステータスを読む

**02.** インプットステータスを読む

**03.** 保持レジスターを読む

**04.** インプットレジスターを読む

**05.** 単一コイルを書く

**06.** 単一レジスターを書く

**15.** 複数コイルを書く

**16.** 複数レジスターを書く

Modbus Master コントロールを使って、Modbus スレーブデバイスとデータをアクセスするアプリケーションロジックは以下のとおりです。

- 接続のためのコントロールプロパティーを設定する。
- 接続を設立するコントロールメソッドを発行する。
- 上記のメッセージタイプの 1 つを使ってコマンドを発行する。
- Slave Response イベントを置き、コントロールからのメッセージを待つ。
- メッセージ結果を読む。

以下に Modbus Master コントロールのプロパティーとメソッドの操作特徴を記述します。

## プロパティー

| | |
|---|---|
| **TransmissionMode** | MODBUS トランスミッションモード（0=ASCII / 1=RTU） |
| **TimeOut** | このコントロールのプロトコルに関連するタイムアウトです。<br>MODBUS ドライバーが、データに送った後にスレーブデバイスからのレスポンスを待つ時間です。　単位：ミリ秒 |
| **BaudRate** | 設定する連続したボーレート<br>（300 / 600 / 1200 / 2400 / 4800 / 9600 / 14400 / 19200 / 38400） |
| **Parity** | COM ポート接続のためのパリティ選択：<br>（0=None / 1=Odd / 2=Even） |
| **StopBits** | ストップビットの数 （0=1bit / 1=1.5bit / 2=2bit） |
| **DataBits** | データビットの数 （7=7bits / 8=8bits） |
| **TCPDevice** | ネットワーク接続のための Modbus/TCP サーバーを定義します。<br>IP アドレスまたはマシン名という形式になります。 このプロパティーは、次に呼び出される ConnectModbusTCP メソッドで使われます。 |

## メソッド

| | |
|---|---|
| **ConnectSerial** | PC の COM ポート経由で Modbus へ接続する<br>. |
| **ConnectModbusTCP()** | Modbus/TCP サーバーへネットワーク接続する<br>. |
| **PollModbus()** | 明示したスレーブへ Read リクエストを出す<br>（Modbus メッセージタイプ 01～04） |
| **ReadResults()** | PollModbus() リクエストに呼応してスレーブから戻ってきた結果を読みます |
| **FillWriteBuffer()** | スレーブデバイスに書かれるデータバッファを準備します。 |
| **WriteModbus()** | Write リクエストを出す<br>（Modbus メッセージタイプ 05, 06, 15, 16） |
| **ReadResults()** | WriteModbus()リクエストに呼応してスレーブから戻ってきた結果を読みます |
| **Disconnect()** | 明示した接続を解放します |

## イベント

以下に Modbus Master コントロールのイベントを記述します。

| | |
|---|---|
| **SlaveReadResponse.** | SlaveReadResponse イベントは、PollModbus メソッドで開始されたメッセージの完了すると、コントロールにより発行されます。 |
| **SlaveWriteResponse.** | SlaveResponse イベントは、WriteModbus メソッドで開始されたメッセージの完了すると、コントロールにより発行されます。 |
| **UserMsgResponse.** | SlaveResponse イベントは、SendUserMsg メソッドで開始されたメッセージの完了すると、コントロールにより発行されます。 |

## Modbus Master API リファレンス

Long ConnectSerial ( short Port )
例：　iHandle＝modMaster1.ConnectSerial（1）
（COM ポート 1 に接続する）

ConnectSerial メソッドは、接続する PC COM ポートを識別するために、ポート番号のパラメータを指定します。 指定したポート番号で、BaudRate、DataBits、StopBits や Parity プロパティーの現在値を使って接続を設立します。ConnectSerial メソッドの戻り値は、PollModbus()、WriteModbus()、ReadResponse などのメソッドに渡す、接続を識別するパラメータとなり、コントロールアプリケーションによって維持されます。 ConnectSerial メソッドが INVALID_HANDLE_VALUE（-1）を戻す時は、接続がエラーになった時です。

Long ConnectModbusTCP ( short Port )
例：　iHandle＝modMaster1.ConnectModbusTCP（502）
（modbus/TCP ポート 502 に接続する）

ConnectModbusTCP メソッドは、引数としてポート番号を使うことで TCPDevice メソッドにより指定されたネットワーク機器との modbus/TCP 接続を設立することをコントロールに指示します。 （TCP ポート番号 502 は、modbus/TCP プロトコルにより定義された標準のサービスポートです。）

接続が開始された時、ConnectModbusTCP メソッドは 0 以外の正数値を戻します。また一方、この DOES NOT は、接続が設立されていることを確認します。アプリケーションは、接続がメッセージトランザクションを受け入れる用意ができていることを確認するために、ConnectionEstablished イベントおよび ConnectionDropped イベントを処理しなければなりません。

Boolean Disconnect ( Long ConnectionHandle )
例：　Status = modMaster1.Disconnect ( iHandle )
（指定した接続ハンドルと関連したすべてのリソースを解放する）

Disconnect メソッドは、指定した接続を閉じて、他の Windows アプリケーションのために、すべての割り当てられたリソースを解放します。 接続が正しく閉じられた時の戻り値は TRUE です。

**Short PollModbus ( Long ConnectionHandle, short SlaveNodeAddress, short ModbusFunctionCode, Long Address, short Length )**

例：　**Status = modMaster1.PollModbus ( iHandle, 1, 3, 100, 10 )**

（ノード 1 から、アドレス 100 で始まる 10 の保持レジスターを読む）

PollModbus メソッドは、明示したスレーブへ Read リクエストを出します。 ConnectionHandle は、ConnectSerial メソッドまたは ConnectModbusTCP メソッドから戻されたハンドル番号と指定して下さい。SlaveNodeAddress は modbus スレーブを定義します。 ModbusFunctionCode は下記にして下さい。

| Function Code | Data Type |
|---|---|
| 01 | COIL STATUS |
| 02 | INPUT STATUS |
| 03 | HOLDING REGISTER |
| 04 | INPUT REGISTER |

Adress はポイントアドレスを定義し、範囲は 1～65535 にして下さい。 Length は、読んでくるデータポイントの数を定義し、範囲は 1～256 にして下さい。 PollModbus メソッドの戻り値が 0 の時、メッセージは接続の外に送信され、後でアプリケーションが SlaveResponse イベントを受け取ります。（もしメッセージがタイムアウトになったとしても、SlaveResponse イベントは発生します。）PollModbus メソッドの戻り値が 0 以外のエラーコードの時、SlaveResponse イベントは発生されません。

**Short ReadResults ( Long ConnectionHandle, short SlaveNodeAddress, short ModbusFunctionCode, Long Address, Long *pData )**

例：

```
Dim MyData( 10 ) As Long
        For i = 0 to 10
Status = modMaster1.ReadResults ( iHandle, 1, 3, 100+i, MyData( i ) )
        Next i
```

（PollModbus によって要求したレスポンス読む）

アプリケーションが、modbus スレーブに Read メッセージを要求した後に、modbus_master コントロールは、メッセージ結果を Read 可能になった時に、SlaveResponse イベントを発生します。ReadResults メソッドは接続時の最後のメッセージ処理の結果を取り出します。

SlaveNodeAddress と ModbusFunctionCode パラメータは、接続時に送った最後のメッセージと同じにして下さい。 スレーブからの結果は、一度に 1 つ値に戻され pData に格納されます。 ReadResults メソッドに渡されるパラメータ Address は、PollModbus メソッドでのパラメータ Address と Length により指定される範囲内である必要があります。

もしメッセージがスレーブデバイスにうまく送られ、適切な結果を得られたならば、ReadResponse は、エラーステータス 0 を戻します。 もしメッセージがタイムアウトになった時（または、メッセージ処理中になにか他のエラーが起きた時）適切なエラーが戻るようになっています。

**Short FillWriteBuffer ( Long ConnectionHandle, short Index, Long Data )**

FillWriteBuffer メソッドは、スレーブデバイスに送信されるデータによって内部バッファを準備します。データは一度に 1 つの値としてバッファに書いて下さい。それぞれの値は単一の modbus コイルまたはレジスターを表しています。バッファが満たされるとすぐ WriteModbus メソッド経由で明示したスレーブに送られます。

**Short WriteModbus ( Long ConnectionHandle, short SlaveNodeAddress, short ModbusFunctionCode, Long Address, short Length )**

**例： modModaster1.FillWriteBuffer ( iHandle, 0, 1234 )**
**Status = modMaster1.WriteModbus ( iHandle, 1, 6, 100, 1 )**
( 1234 をノード 1 のアドレス 100 に書く)

WritelModbus メソッドは、定義されたスレーブに Write リクエストを出します。ConnectionHandle は、ConnectSerial メソッドまたは ConnectModbusTCP メソッドから戻されたハンドル番号と指定して下さい。SlaveNodeAddress は modbus スレーブを定義します。ModbusFunctionCode は下記にして下さい。

| Function Code | Data Type |
|---|---|
| 05 | SINGLE COIL WRITE |
| 06 | SINGLE REGISTER WRITE |
| 15 | MULTIPLE COIL WRITE |
| 16 | MULTIPLE REGISTER WRITE |

Address はデータポイントアドレスを指定し、Length は、書き込むデータ値の数を定義します。
書き込むデータは、直前に、FillWriteBuffer メソッドを使って指定して下さい。

**Short WriteResponse ( Long ConnectionHandle, Short SlaveNodeAddress, Short ModbusFunctionCode, Long Address, Short Length )**

**例： Status = modMaster1.WriteResults ( iHandle, 1, 6, 100, 1 )**
(WriteModbus によって開始されたレスポンスを読む)

アプリケーションが、modbus スレーブに Read メッセージを要求した後に、modbus_master コントロールは、メッセージ結果を Read 可能になった時に、SlaveResponse イベントを発生します。
WriteResults メソッドは接続時の最後のメッセージ処理の結果を取り出します。

SlaveNodeAddress、ModbusFunctionCode、Address、および Length パラメータは、接続時に送った最後のメッセージと同じにして下さい。

もしメッセージがスレーブデバイスにうまく送られ、適切な結果を得られたならば、WriteResponse は、エラーステータス 0 を戻します。 もしメッセージがタイムアウトになった時（または、メッセージ処理中になにか他のエラーが起きた時）適切なエラーが戻るようになっています。