# RAC プロバイダ

## RAC 通信

Version 1.1.0

# ユーザーズ ガイド

July 17, 2012

【備考】
・本プロバイダの一部は, (財)機械振興協会 技術研究所が競輪の補助により実施した研究成果の一部を使用しています.

## 【改版履歴】

| バージョン | 日付 | 内容 |
|---|---|---|
| 1.0.0.0 | 2006-02-24 | 初版 |
| 1.1.0.0 | 2009-07-03 | リトライ処理削除（RAC はリトライ情報が無い）, Clear コマンド追加 |
| 1.1.0.1 | 2010-02-12 | エラーコード追加 |
| 1.1.0 | 2012-07-17 | ドキュメントのバージョンルールを変更 |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## 【対応機器】

| 機種 | バージョン | 注意事項 |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# 目次

# 1. はじめに

本書は，ストリーム通信で RAC(Robot Action Command)を送受信するための CAO プロバイダである，RAC プロバイダのユーザーズガイドです．

RAC プロバイダは，リモートマシンに対して，RAC の送受信を行います．通信方法として RS-232C 通信と Ethernet の Socket ストリーム通信(TCP/IP)を選択でき，TCP/IP のときはサーバ，クライアントモードを選択することができます．

この RAC プロバイダを使用することで，CAO クライアントは RAC コマンドの送受信を簡単に使用することができます．

本書は，この RAC プロバイダの機能と実装されているメソッドについて説明します．

# 2. プロバイダの概要

## 2.1. 概要

RAC プロバイダは, RAC(Robot Action Command)をストリーム通信で送受信するプロバイダです. 通信形態は, RS-232C 通信および Ethernet の Socket ストリーム通信(TCP/IP)のを選択することができます.

RAC プロバイダでは通信の同期, 非同期の設定をすることができます. また通信データを加工する機能として, Unicode 変換, エスケープ文字変換機能などがあります.

RAC プロバイダのファイル形式は DLL(Dynamic Link Library)となっており, その詳細は表 2-1 のようになっています.

**表 2-1 RAC プロバイダ**

| | |
|---|---|
| ファイル名 | CaoProvRAC.dll |
| ProgID | CaoProv.RAC |
| レジストリ登録[1] | regsvr32 CaoProvRAC.dll |
| レジストリ登録の抹消 | regsvr32 /u CaoProvRAC.dll |

[1] ORiN SDK でインストールした場合は手動で登録／抹消する必要はありません.

### 2.1.1. RAC コマンド

ここで RAC の要求コマンドおよびコマンド応答の表記規則を示します. ここで角括弧(“[ ]”)は省略可能なことを示しています.

**要求コマンド**

<トップレベルコマンド>:[<部位名>]:[<部位番号>]:[<サブコマンド>]:[<パラメータ>]

| | | |
|---|---|---|
| <トップレベルコマンド> | : | “START”, “STOP”, “PUT”, “GET”のいずれか |
| <部位名> | : | コマンドの対象になるロボットの部位 |
| <部位番号> | : | 部位名で指定された部位が複数あるときの識別番号 |
| <サブコマンド> | : | トップレベルコマンドの具体的動作内容 |
| <パラメータ> | : | サブコマンド実行に必要なパラメータ |

**コマンド応答**

<処理結果>[,<データ型>,<データ列>]

| | | |
|---|---|---|
| <処理結果> | : | HRESULT 型の整数値 |
| <データ型> | : | VARTYPE 型で表記されるデータ型 |
| <データ列> | : | データ型で変換される文字列データ |

### 2.1.2. 同期・非同期設定

RAC プロバイダでは, AddController メソッド(2.2.1)の Sync オプションを指定することで, 通信の同期, 非同期の設定をすることができます.

同期モードに設定したときは, RAC コマンド送信した後, 直ちに制御が返されます. RAC コマンドの実行結果はコントローラの OnMessage イベントで取得します.

非同期モードに設定したときは, RAC コマンドの実行結果が返されるまで, Command::Execute メソッドから制御は返されません. [2]

### 2.1.3. UNICODE 変換

AddController メソッド(2.2.1)の PacketOpt オプション内の<Convert>の 3 ビット目が TRUE の場合は以下の UNICODE 変換を行います.

(1) 送信コマンドを Unicode(WCHAR)から S-JIS(CHAR)に変換した後, 送信します.

(2) 受信した応答コマンドを S-JIS(CHAR)から Unicode(WCHAR)に変換します.

同期モードのときは, 強制的に UNICODE 変換ありに設定されます.

### 2.1.4. ISO コード, EIA コード変換

データを ASCII コードから指定したコードに変換して送受信します

[2] Command クラスの非同期実行は, 非同期モードに関係なく使用することができます.

UNICODE 変換機能と併用したとき，データは UNICODE から指定したコードに変換して送受信を行います．

### 2.1.5. エスケープ変換

AddController メソッド(2.2.1)の PacketOpt オプション内の<Escape>でエスケープ文字として使用する，文字コードを指定します．文字コードに 0 が指定されているときはエスケープ変換を行いません．

送信時にエンコード，受信時にデコードを行います．

変換規則は以下のようになります．

(3) 1 バイトの英文字(A-Z，a-z，0-9)と記号の一部(*-.@_)は変換しません．

(4) 半角スペースは“+”に変換します．

(5) その他の文字は“指定したエスケープ文字+16 進数 2 桁の文字コード”に変換します．

例)エスケープ文字“%”のとき

変換前 ： @test=This is sample.

変換後 ： @test%3DThis+is+sample.

## 2.2. メソッド・プロパティ

### 2.2.1. CaoWorkspace::AddController メソッド

RAC プロバイダでは，AddController で接続パラメータを設定し，通信の接続を行います．

このときオプションで通信形態，接続パラメータ，タイムアウトの設定，同期設定，エスケープ文字の指定を行います．

以下に AddController の引数仕様を示します．

```
AddController
(
    "<Controller 名>",      // コントローラ名
    "CaoProv.RAC",          // プロバイダ名．固定．
    "<マシン名>",           // プロバイダの実行マシン名．
    "<オプション>"          // オプション文字列
)
```

以下にオプション文字列に指定するリストを示します．ここで，通信デバイスの欄に“-”が入っているオプションは，そのデバイスを指定したときに無視されます．

**表 2-2 CaoWorkspace::AddController のオプション文字列**

| オプション | 説明 | 有効デバイス | | |
|---|---|---|---|---|
| | | Ethernet | | com |
| | | Server | Client | |
| Conn<br>=<接続パラメータ> | 必須．通信形態と接続パラメータ．<br>（参照 2.2.1.1） | ○ | ○ | ○ |
| PacketOpt<br>[=<パケットパラメータ>] | 通信パケットの設定をします．<br>（デフォルト：“4:0:0:0”）<br>（参照 2.2.1.2） | ○ | ○ | ○ |
| EtherOpt<br>[=<Ether パラメータ>] | Ethernet 通信時の設定をします．<br>（デフォルト：“0:10:900”）<br>（参照 2.2.1.3） | ○ | ○ | - |
| MyIP<br>[=<ローカル IP アドレス>] | 複数の NIC を使う場合にこのオプションで IP アドレスを指定して NIC を選択することができます．省略した場合は，自動的に選択されます．ローカルマシンに割り当てられていない IP アドレスを指定したときはエラーを返します． | ○ | ○ | - |
| Timeout<br>[=<タイムアウト時間>] | 送受信時のタイムアウト時間．(ミリ秒)<br>（デフォルト：500） | ○ | ○ | ○ |

| | | | | |
|---|---|---|---|---|
| Sync<br>[=<True/False>] | 同期設定<br>True ： 同期(デフォルト)<br>False ： 非同期 | -<br>非同期 | ○ | ○[3] |

#### 2.2.1.1. Conn オプション

以下に Conn オプションの接続パラメータ文字列を示します. ここで角括弧(“[ ]”)内は省略可能を示します. また, 各パラメータの解説中の下線部はオプションを指定しなかったときのデフォルト値になります.

**RS232C デバイス**

“Conn=com:<COM Port>[:<BaudRate>[:<Parity>:<DataBits>:<StopBits>[:<Flow>]]]”

<COM Port> : COM ポート番号. ‘1’-COM1, ‘2’-COM2, …
<BaudRate> : 通信速度. 4800, 9600, 19200, 38400, 57600, 115200.
<Parity> : パリティ. ‘N’-NONE, ‘E’-EVEN, ‘O’-ODD.
<DataBits> : データビット数. ‘7’-7bit, ‘8’-8bit.
<StopBits> : ストップビット数. ‘1’-1bit, ‘2’-2bit.
<Flow> : フロー制御. ‘1’-Xon/Xoff, ‘2’-ハードウェア制御.
OR をとって指定できます.

**EtherNet デバイス**

“Conn=eth:[<IP Address>[:<Port No>]]”

<IP Address> : TCP/IP IP アドレス. "Mode"オプションでサーバモードが指定されているときは, このパラメータの値は無視されます.
例:“127.0.0.1”, “192.168.0.1”
<Port No> : TCP/IP 接続ポート番号. 5006, 5007, …任意指定可能

#### 2.2.1.2. PacketOpt オプション

以下に PacketOpt オプションのパラメータ文字列を示します. ここで角括弧(“[ ]”)内は省略可能を示します. また, 各パラメータの解説中の下線部はオプションを指定しなかったときのデフォルト値を示します.

“PacketOpt =[<Convert>[:<Header>[:<Term>[:<Escape>]]]]”

<Convert> : 文字列変換.
下位 2 ビットで ISO 変換, EIA 変換の設定し 3 ビット目で Unicode 変換の設定をします. [4]
‘1’-ISO 変換, ‘2’-EIA 変換, ‘4’-Unicode 変換
例 1:Unicode 変換のみの場合 : 4
例 2:ISO 変換のみの場合 : 1

[3] COM 接続でサーバ側は必ず非同期にしてください. 同期に設定するとクライアントからのコマンドを受信できません.
[4] 同期モードのときは, 強制的に Unicode 変換が ON に設定されます.

例 3:Unicode 変換と EIA 変換の場合： 6

<Header> : ヘッダ指定.
‘0’-なし, ‘1’-ENQ(0x05)

<Term> : ターミネータ指定.
‘0’-CR(0x0D), ‘1’-LF(0x0A), ‘2’-CR+LF(0x0D0A)

<Escape> : エスケープ変換に使用する文字コード.
‘0’ -エスケープ変換なし
‘0’以外 -指定した文字コードでエスケープ変換

PacketOpt オプションの設定には, 以下の制限があります.

・ ISO 変換と EIA 変換を同時に ON に指定した場合はエラーになります.
・ Unicode 変換が OFF のときにエスケープ変換を設定した場合はエラーになります.

以下に設定可能な変換の組み合わせの一覧を示します.

**表 2-3 設定可能なオプション値**

| <Convert> | | | | <Escape> |
|---|---|---|---|---|
| ISO 変換 | EIA 変換 | Unicode 変換 | 設定値 | 設定値 |
| - | - | - | 0 | 0 |
| ○ | - | - | 1 | 0 |
| - | ○ | - | 2 | 0 |
| - | - | ○ | 4 | 任意 |
| ○ | - | ○ | 5 | 任意 |
| - | ○ | ○ | 6 | 任意 |

#### 2.2.1.3. EtherOpt オプション

以下に EtherOpt オプションのパラメータ文字列を示します. ここで角括弧(“[ ]”)内は省略可能を示します. また, 各パラメータの解説中の下線部はオプションを指定しなかったときのデフォルト値を示します. またデバイスに RS232C を指定したときは, このオプションは無視されます.

“EtherOpt =[<Mode>[:<ConnMax>[:<ConnTimeout>]]]”

<Mode> : 文字列変換.
‘0’-クライアントモード, ‘0’以外 -サーバモード
Conn オプションで指定したデバイスが"com"のとき,このオプションは無視されます.

<ConnMax> : サーバモード時の最大クライアント数. (デフォルト:10)
サーバモード以外のとき, この値は無視されます.

<ConnTimeout> : サーバモード時の最大通信待ち時間. (デフォルト:900)

サーバモード以外のとき，この値は無視されます．

指定した時間だけクライアントから通信がないときは，そのクライアントを切断します．このパラメータの単位は秒で指定します．

### 2.2.2. CaoController::AddCommand メソッド

非同期通信または EtherNet(サーバ)のときは，このメソッドは失敗します．

このメソッドは，同期モードで RAC コマンドを実行するためのコマンドオブジェクトを生成します．

コマンド名には，任意の名前を使用することができます．

オプション文字列は使用しません．

```
AddCommand
(
    "<コマンド名>",        // コマンド名（任意）
    "<オプション>"         // オプション文字列（未使用）
)
```

### 2.2.3. CaoController::AddVariable メソッド

非同期通信または EtherNet(サーバ)のときは，このメソッドは失敗します．

このメソッドは，RAC の GET，PUT コマンドを同期実行する変数オブジェクトを生成します．

変数名は，以下のように指定します．

“[<部位名>]:[<部位番号>]:[<サブコマンド>]”

このとき角括弧([ ])内は省略することができ，省略時には，空文字列が使用されます．

オプション文字列は使用しません．

```
AddVariable
(
    "<変数名>",            //  “[<部位名>]:[<部位番号>]:[<サブコマンド>]”
    "<オプション>"         // オプション文字列（未使用）
)
```

### 2.2.4. CaoController::Execute メソッド

非同期通信または EtherNet(サーバ)のときは，このメソッドは失敗します．有効なコマンドは，"Send"と"Clear"の二つのみで，"Clear"はバッファをクリアするだけです．引数はありません．

"Send"コマンドは引数の<RAC コマンド>に指定した RAC コマンドをそのまま送信します．RAC コマンドの文法チェックは行いません．このとき RAC コマンドは必ず文字列型で指定してください．

```
Execute
(
    "Clear",               // バッファクリア
    "",                    // (省略可)
)

Execute
(
    "Send",                // RAC コマンド送信
```

```
    "<RAC コマンド>",      //[トップコマンド]:[部位名]:[部位番号]:[サブコマンド]:[パラメータ]
)
```

### 2.2.5. CaoController::OnMessage イベント

同期通信のときはこのイベントは発生しません.

RAC プロバイダが通信デバイスからデータを受信すると, RaoController クラスの OnMessage イベントとしてクライアントにデータを受け渡します. このとき, Message::Value プロパティに受信データをそのまま格納します.

受信したデータが RAC コマンド又は応答コマンドであるかは確認しません.

### 2.2.6. CaoCommand::Execute メソッド

Parameter プロパティに設定したデータから RAC コマンドを生成, 実行します.

実行結果は Result プロパティに格納されます.

### 2.2.7. CaoCommand::put_Parameters プロパティ

実行する RAC コマンドを設定します. コマンドは文字列, 文字列配列, VARIANT 配列で指定することができ, データ型によって, 以下のように解釈されます.

**表 2-4 RAC コマンドのデータ型ごとの解釈方法**

| コマンドのデータ型 | 解釈方法 |
|---|---|
| 文字列(VT_BSTR) | 完成している RAC コマンド |
| 配列(VT_??? \| VT_ARRAY) | RAC コマンドの各要素.<br>先頭から5番目の要素までを使用し, それ以降のデータは無視します.<br>要素が5個以下のときは, 不足部分のデータを空にして RAC コマンドを作成します.<br>各要素のデータは文字列に変換されます. |

### 2.2.8. CaoCommand::get_Parameters プロパティ

Parameters プロパティに設定されているコマンドを取得します.

コマンドを設定していないときは VT_EMPTY を返します.

### 2.2.9. CaoCommand::get_Result プロパティ

直前に実行した CaoCommand::Execute メソッドの実行結果を取得します.

RAC の実行結果は, 処理結果を HRESULT で返し, 応答データを戻り値に入れます. 応答データがないときは, 戻り値が VT_EMPTY となります.

### 2.2.10. CaoVariable::put_Value プロパティ

RAC の PUT コマンドを実行します. このとき変数名と引数を組み合わせて, RAC の要求コマンドを生成します. このとき引数で指定した値は“<データ型>,<値>”という形に変換されます.

以下に生成する RAC コマンドを示します.

“PUT:<変数名>:<put_Value の引数>”

RAC のコマンド応答は, 処理結果を HRESULT で返します. 応答データは取得できません.

```
put_Value
(
    "<設定値>",              // <パラメータ>
)
```

### 2.2.11. CaoVariable:get_Value プロパティ

RAC の GET コマンドを実行します. このとき変数名から RAC の要求コマンドを生成します.

以下に生成する RAC コマンドを示します.

“GET:<変数名>:”

RAC の実行結果は, 処理結果を HRESULT で返し, 応答データを戻り値に入れます. 応答データがないときは, 戻り値が VT_EMPTY となります.

### 2.2.12. CaoMessage::Reply メソッド

引数の<返信メッセージ>は必ず文字列型で指定しなければなりません. それ以外の型のとき, このメソッドは失敗します.

指定した文字列を Message::Value プロパティの RAC の要求コマンドに対する応答コマンドとして送信します.

引数の<返信メッセージ>の応答コマンドをそのまま返信します. RAC コマンドの文法チェックは行いません.

```
Reply
(
    "<返信メッセージ>",    // 応答データ
)
```

## 2.3. 変数一覧

RAC プロバイダ固有の変数はありません.

## 2.4. エラーコード

RAC プロバイダでは，固有のエラーコードはありません．ORiN2 共通エラーについては，「ORiN2 プログラミングガイド」のエラーコードの章を参照してください．

# 3. サンプルプログラム

## 3.1. サンプルプログラム

以下に, TCP/IP 通信のサーバモードで起動する“Server.exe”とクライアントモードで起動する“Client.exe”の間で通信するサンプルを示します.

サーバ IP アドレス: 10.7.9.250

実行コマンド: GET:sample:1:Val:

**List 3-1 SampleServer.frm**

```
Private eng As CaoEngine
Private WithEvents ctrl As CaoController

Private Sub Form_Load()

    Set eng = New CaoEngine

    ' サーバモードで通信開始
    Set ctrl = eng.Workspaces(0).AddController("Sample", _
                                               "CaoProv.RAC", _
                                               "", _
                                               "Conn=eth::5006,Mode=1")

End Sub

' 受信イベント
Private Sub ctrl_OnMessage(ppCaoMess As CAOLib.ICaoMessage)

    ' 受信した内容をそのまま返信
    ppCaoMess.Reply text1.text

End Sub
```

**List 3-2 SampleClient.frm**

```
Private eng As CaoEngine
Private ctrl As CaoController
Private cmd As CaoCommand

Private Sub Form_Load()

    Set eng = New CaoEngine

    ' クライアントモードで通信開始
    Set ctrl = eng.Workspaces(0).AddController("Sample", _
                                               "CaoProv.RAC", _
                                               "", _
                                               "Conn=eth:10.7.9.250:5006")

    Set cmd = ctrl.AddCommand("sample")
End Sub

Private Sub Command1_Click()
On Error GoTo ErrProc

    ' RAC コマンド実行
    cmd.Parameters = "GET:sample:1::10"
    cmd.execute(0)

    Exit Sub
ErrProc:
    MsgBox Err.Description
End Sub
```

## 3.2. テストプログラム

CAO の RAC プロバイダ専用のテストクライアントとして RACTester を用意しています. このクライアントは, RAC プロバイダで実装されている CAO のインタフェースのみを扱うように作成されています.

### 3.2.1. 画面構成について

以下に RACTester の画面を示します.

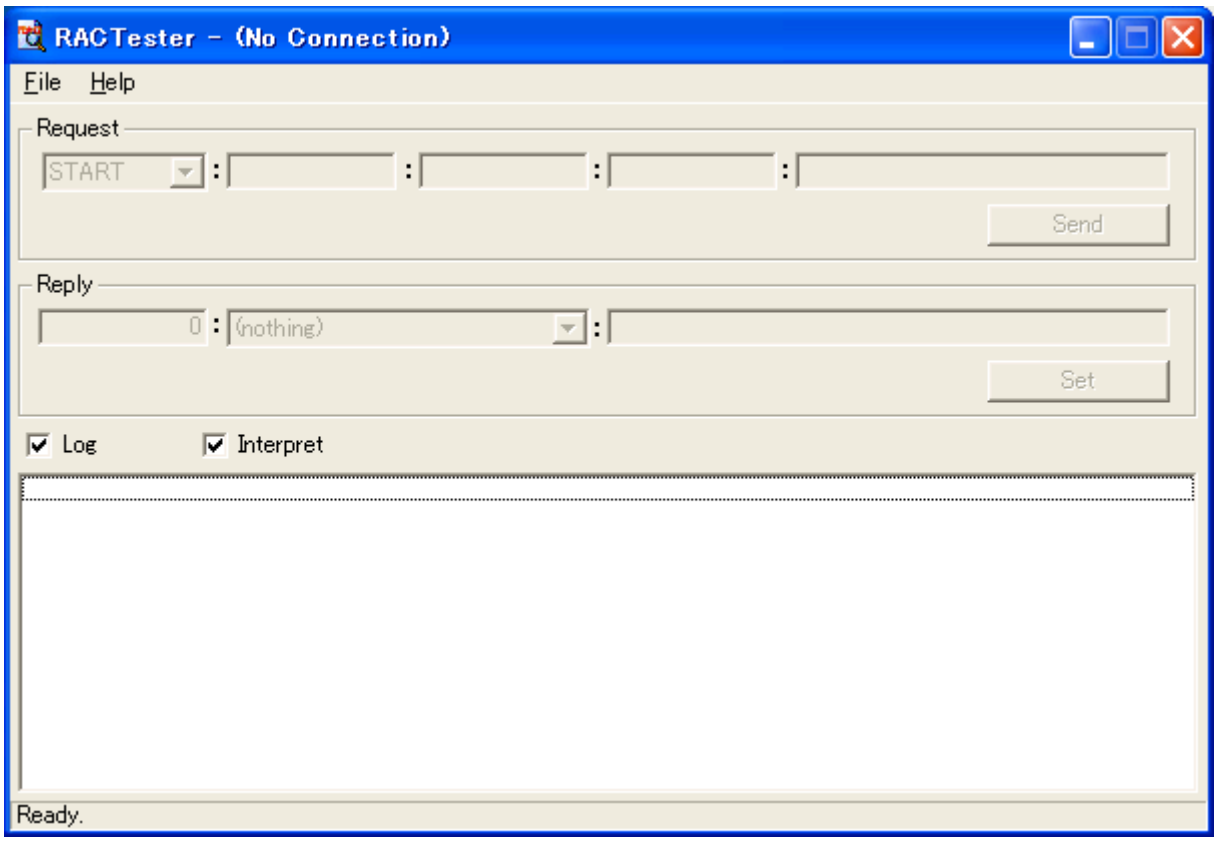


図 3-1 RACTester 画面

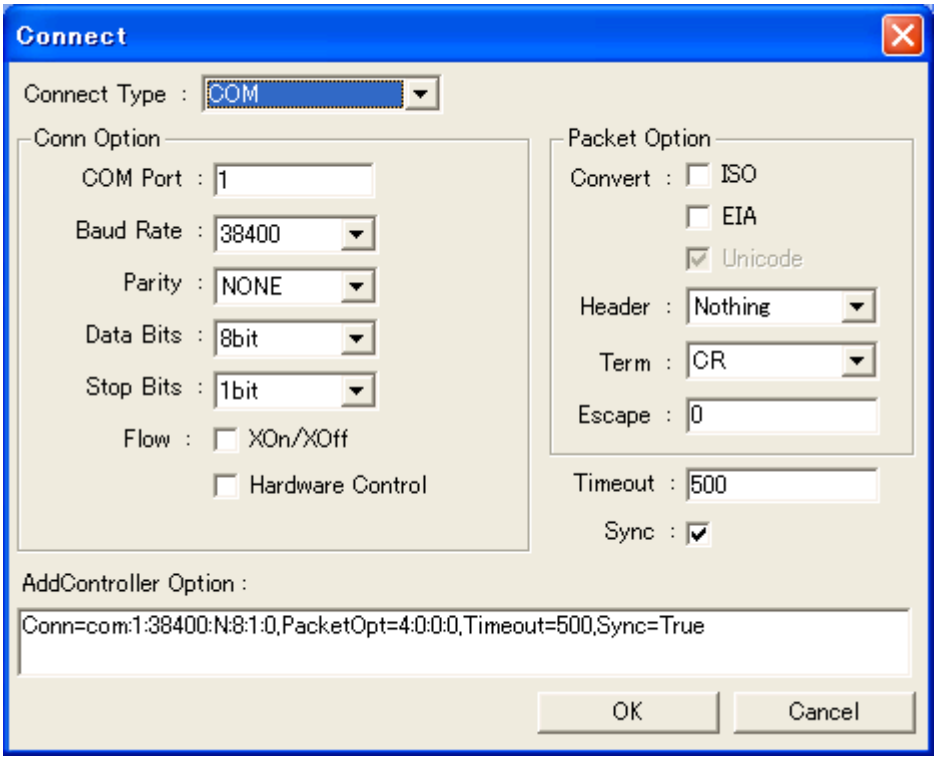


**図 3-2 “Connect”画面**