2009/11

M10

リモートキーボードエミュレータ

有限会社リビッグ

横浜市港南区上大岡西 1-12-2-801

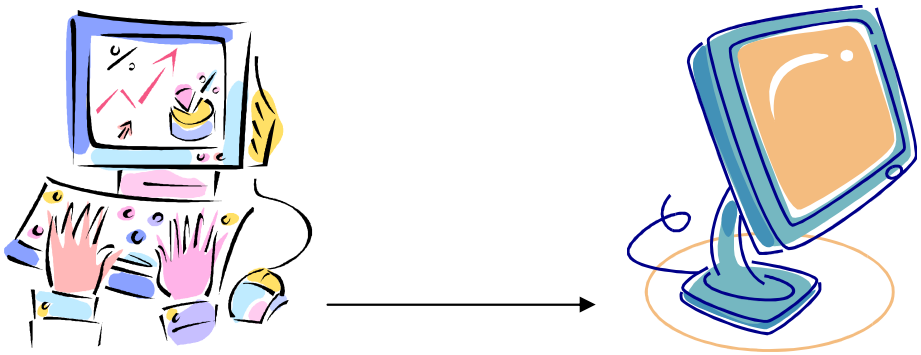
Tel: 045-843-7122

http://www.ribig.co.jp

目次

はじめに

このプログラムは、コンピュータ（操作するコンピュータ）からキーボード操作に対応する文字列を他のコンピュータ（操作されるコンピュータ）に送ることで、指定キーボード操作を他のコンピュータ上で再現するものです。



操作する側のコンピュータ　　操作される側のコンピュータ

1. 操作される側のコンピュータで M10.EXE を起動します（ Windows XP ではサービスとして登録可能です）。M10 が起動している間、操作する側からのキーボード操作文字列を待ち受けます。文字列を受け取ると指定キーボード操作を実行しているコンピュータ上で再現します。

   M10.EXE を実行するには .NET Framework 3.0 以上が必要です。対象 OS は Windows XP/ Vista/7 32bit/64bit (ただし、XP/7 では Windows サービスとして実行できません。通常のアプリケーションとして実行します　)

2. 操作する側のコンピュータには、M10.EXE が理解できる形式の文字列を TCP 経由で M10.EXE に送るプログラムが必要です。本プログラムには C# で書かれたサンプルが含まれます(Visual Studio 2008, .NET Framework 3.0 で動作).

   対象 OS は.NET Framework 3.0 以上がインストールされた Windows XP/Vista/7 32bit/64bit.

操作される側のコンピュータ

M10.EXE は Windows サービスと登録したり、通常のプログラムとして実行できます。

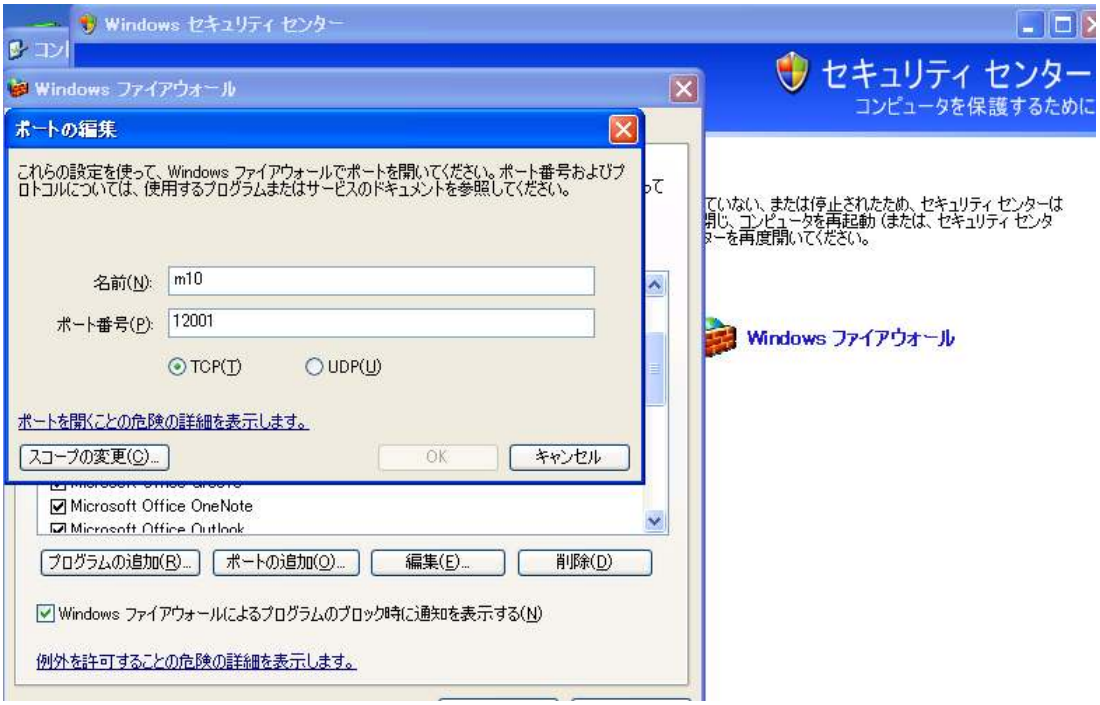
M10 はネットワークサーバプログラムです。利用するためには、通常のネットワークサーバプログラムを Windows で実行するために行う設定（ポート指定やファイアーウオール解除）が必要です

M10 導入

登録作業するには管理者ユーザでログインしてください。

ファイアーウオールの設定

M10 は既定ではポート 12001 を使います。他の任意のポートを指定することもできます。インストールする前に Windows ファイアーウオールで、M10 が使うポートを開いてください。特にインストール時に Windows サービスとして登録する場合、インストール過程でサービスが起動されますので、この設定を行っていないとサービスは自動起動できません。

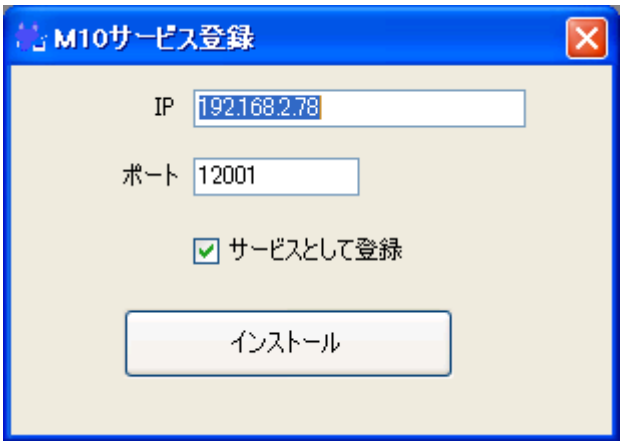


１．「コントロールパネル」－「セキュリティセンタ」－「Windows ファイアウォール」
２．“例外”タブを選択
３．“ポートの追加”ボタンをクリック
４．M10 の使うポート番号（既定 12001 ）を指定。名前は任意（ここでは　M10 を指定）
５．[OK]ボタンで完了

インストール

付属の USB キーを接続してください。M10 利用時には、必ずキーを接続したままにしておいてください。

ディスクの「サーバ」フォルダ内の M10Inst.exe を実行します(USB キーが接続されていないと起動しません)。次のウィンドウが開きます。



IP

プログラムが最初にコンピュータ上で見つけた IP アドレスが設定されます。変更可能です。複数のネットワークカードが装着されている場合、M10 が待ち受けるネットワークの IP アドレスになっていることを確認してください。

ポート

既定の 12001 が設定されます。変更可能です。

サービスとして登録

XP までの OS ではチェックされ、Vista 以降ではチェックされません。Vista 以降ではチェックを付けずにインストールしてください。

M10 を実行するコンピュータの IP アドレスはクライアント側で指定する必要がありますので、控えておいてください。

[インストール]ボタンをクリックするとインストールが開始します。[サービスとして登録]をチェックするとコマンドプロンプトが自動で開きますが、何もしないでインストール完了までお待ちください。

以上ですべての準備が整います。別のコンピュータ上で M10 クライアント(9 ページ)を実行して、キー操作が再生されるか確認してみてください。

設定とアンインストール

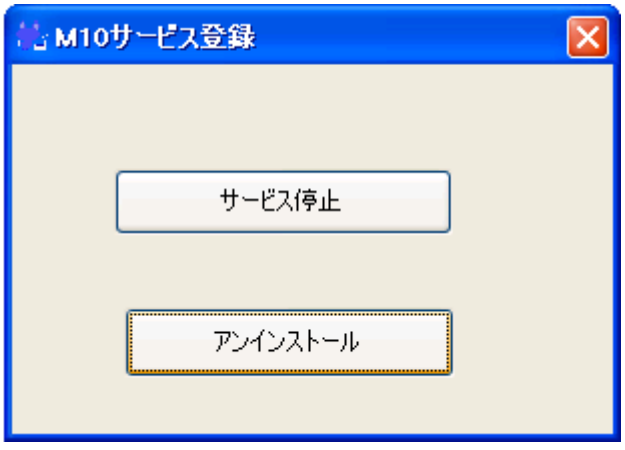
インストールが完了すると、「スタート」－「すべてのプログラム」に「M10」が作成されます。

M10
プログラム本体

M10Inst
M10 をアンインストールしたり、M10 サービスの開始/停止をします。

設定ファイル
インストール時に指定した IP/ポートが書き込まれた設定ファイルです。

M10 がサービスとして実行している時に M10Inst を実行すると、次のようなウィンドウが表示されます。



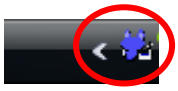
[サービス停止]ボタンで M10 サービスを停止できます。再度 M10Inst を実行すると、サーを開始できます。

[アンインストール]ボタンで M10 をコンピュータから削除できます。

通常のプログラムとして M10 を実行

M10 は通常のプログラムと同様に実行できます。

スタートアップに入れておくとログイン後に自動的に起動されるため便利です。



起動するとタスクトレイに　M10 のアイコンが現れます。

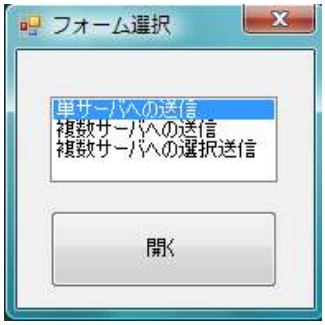
M10 は同時に 1 つしか実行できません。サービスとして実行している間は、通常のプログラムとして実行できません。逆に、通常のプログラムとして実行している間は、サービスとして実行できません。

操作する側のコンピュータ

基本クライアントとして M10_Client.exe を添付しています。

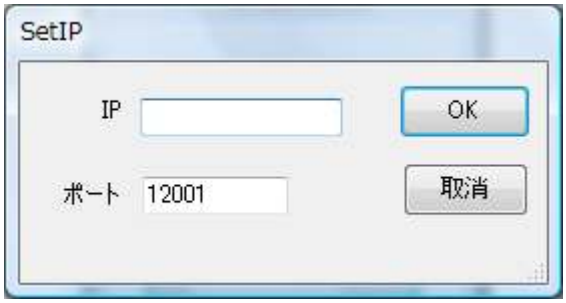
M10_Client.exe を実行するには、**M10_Client.EXE と M10Lib.DLL の 2 つのファイル**を実行するコンピュータの任意のハードディスクスク上のフォルダにコピーします。そのフォルダから M10_Client.exe を実行してください。

M10_Client 実行
実行すると次のウィンドウが現れます。



“単サーバへの送信“を選択して、「開く」ボタンをクリックしてください。次のウィンドウが開きます。

もし、M10 が起動しているコンピュータの IP とポートが指定されていない場合には、IP/ポートを指定するウィンドウが表示します。キー操作文字列を受け取る M10 側の IP とポートを指定してください（ M10 インストール時の IP とポートをを指定します ）

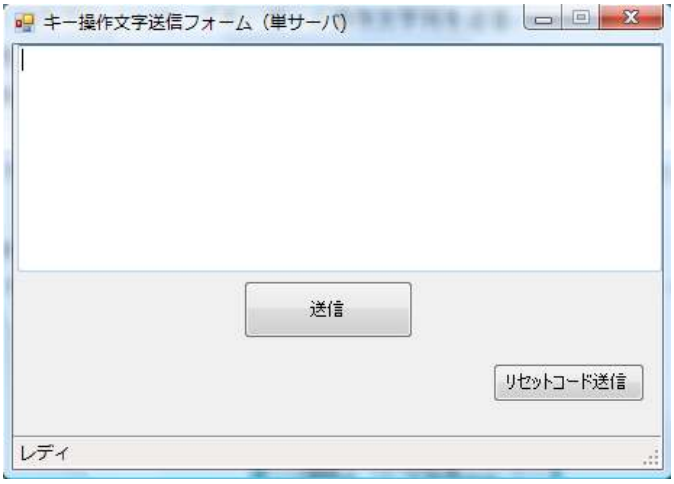


OK をクリックすると M10_Client.EXE と同じフォルダに設定ファイルが作成されます。

M10_Client.ini 例
[Setting]
Ip=192.168.1.20
Port=12001

M10 が実行しているコンピュータの IP アドレスに ping で接続できなければデータを送信できません。もしデータ送信が正しく行われないようであれば、最初に ping で確認してください。

そして、次のウィンドウが開きます。



キー操作文字列の指定
テキストフィールドにキーボード操作文字列を設定して、「送信」ボタンをクリックすると M10 が実行しているコンピュータで指定キーボード操作が再現されます。

もっとも簡単な指定は、文字入力操作です。入力したい文字をテキストフィールドに入力してみてください。

例： M10_Client 実行

送信ボタンを押すと、M10 側で”M10_Client 実行“とそのまま入力されます(M10 側コンピュータではテキストエディターなどを実行して、キーボード入力を受けてる状態にしてください)

次に長めの文字列（評価版では最大 256 バイト、製品版では 0x7ffff バイト ）を設定してみてください。

例： .Net Framework Runtime V3.0 以上が必要です。実行すると次のウィンドウが現れます。

ファンクションキー(Tab, Enter, BS, Del, F1 等)は、¥02¥xx(xx は 16 進数の仮想キーコード)という書式で指定できます。稀に文字コードと仮想キーコードが衝突することがあります。その場合、文字コードが仮想キーコードとして解釈されてしまい、意図していたキー操作として再現されません。そのような状況では文字ではなく直接仮想キーコードを指定することで意図した操作を再現できます。

仮想キーコード

キーボードにはメーカによってキー数、キー種類などが異なります。Windows はメーカ毎に異なるキーボードに対応するのではなく、キーの仮想キーコードを定義しキーボード（ドライバー）側で仮想キーコードで生成して Windows に渡しています。ノート PC の fn キーは、キー数が少ないキーボードで fn と他のキーの組み合わせで仮想キーコードを生成します。キー数が多いキーボードでは PageUp, PageDown, Home, End キーを独立させることができます。それぞれのキーを押すと対応する仮想キーコードが生成されます。一方、キー数の少ないノート PC では、fn+[他のキー]の組み合わせで PageUp, PageDown, Home, End などの仮想キーコードが生成されます。仮想キーコードを利用することで物理的なキー操作に関係なく、キーを定義しています。

M10 では、以下のキーは次のコードで指定できます。

Enter	¥0d
Tab	¥09
上矢印	¥02¥26
下矢印	¥02¥28
左矢印	¥02¥25
右矢印	¥02¥27
ALT+下矢印	¥0e
PrintScrn	¥02¥2c
プログラム実行	¥06(プログラムパス)¥06
アイドリング(2.5 秒)	¥16
Alt	¥04 (押し下げ/押し上げ)
Shift	¥03(押し下げ/押し上げ)
Ctrl	¥05(押し下げ/押し上げ)
MENU	¥02¥12
Windows キー	¥02¥5b or ¥02¥5c

¥02¥xx 形式で仮想キーコードを指定しますが、Tab, Enter のように¥02 を使わずに¥xx で指定できるキーもあります。 ALT+下矢印 (¥0e)、プログラム実行、アイドリング、Alt, Shift, Ctrl は M10 が独自に定義したものです。

例：
¥0e¥0dy （ ALT+下矢印, Enter, y キーの操作 ）

ALT+下矢印に新規ウィンドウを開くという機能が割り当てられたプログラムに出力すると、ALT+下矢印で新規ウィンドウを開き、確認メッセージに対して Enter して、その後の はい、いいえ に対して y を入力する操作になります。

例：
ALT+下矢印¥02¥25¥02¥25¥02¥25¥08¥02¥23¥0d 新規ウィンドウ¥0d 開く

テキストエディターには、次の様に出力されます

ALT 下矢印
新規ウィンドウ
開く

“ALT+下矢印“が出力されてから、左矢印が 3 回（**¥02¥25¥02¥25¥02¥25**）、その後 BS(**¥08**) で”+”が削除され、END(**¥02¥23**)でカーソルが行末に移動します。Enter で改行、”新規ウィンドウ“が入力されてから　Enter、そして、”開く“が入力されます。

例：
ALT+下矢印¥02¥25¥02¥25¥02¥25¥08¥02¥23¥0d 新規ウィンドウ¥0d 開く¥02¥2c

上と同じですが、最後に画面のスナップショットをクリップボードにコピーします。

例：
¥06http://www.google.co.jp¥06¥16¥16¥16¥16¥16 リビッグ¥0d

既定のブラウザで http://www.google.co.jp を開きます。¥06 で囲まれた文字列はキーボード操作ではなく、ブラウザに渡される文字です。その後 12.5 秒（ **¥16** が 5 個）待ってから、検索文字として“リビッグ”を入力をして検索を実行します。

例:
¥03¥05¥02¥70¥05¥03　(SHIFT+CTRL+F1)

Alt, Shift, Ctrl を表す **¥04, ¥03, ¥05** は、押し下げ、押し上げを再現します。この例は、 **¥03** で Shift 押し下げ、**¥05** で Ctrl 押し下げ、**¥02¥70** で F1 の押し下げ/押し上げ、**¥05** で Ctrl 押し上げ、**¥03** で Shift 押し上げとなります。この例のように、**¥03、¥04、¥05** は、必ず、押し下げ/押し上げを再現するように 2 度使います。

ALT+DOWN は次のコードでも再現できます。
¥04¥02¥28¥04

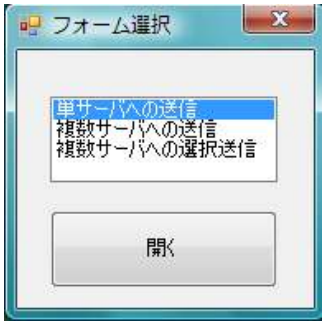
M10 は受け取った文字は IME の状態に関係なく指定通りに出力します。¥17 で IME を ON/OFF することはできますが、仮に IME が ON になっていたとしても、指定文字は　IME に影響されることなくそのまま出力されます（ IME が ON の状態で　a を入力しても a として出力されます。“あ”と変換されることはありません　）

リセットコード送信

文字列指定に誤りがあったり、また、文字列送信に異常があると M10 の内部状態がおかしな状態になることがあります。文字列送信で 10 文字送ると指定したけれども、9 文字しか送らないと M10 は残りの 1 文字を受け取るまで待ち続けます。その状態で次の文字列を送信してしまうと、M10 側では正しく文字列を解釈できません。

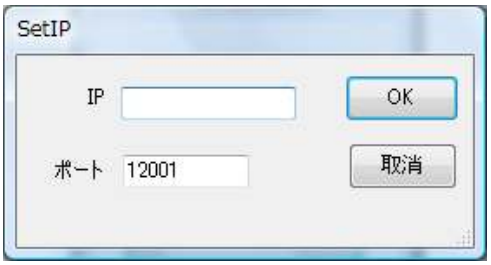
“リセットコード送信”ボタンは、M10 の内部状態をリセットして、新たに文字列を受け入れる状態にします。

複数サーバへの送信



M10_Client では、同時に複数のサーバに同時にキーボード操作文字列を送信したり、、複数のサーバから送信する対象を選択してキーボード操作文字列を送信することができます。

“複数サーバへの送信”　を選択して[開く]をクリックします。もし、M10 が実行しているコンピュータの IP とポートが指定されていなければ次のウィンドウが開きます。



ここでは 1 つの M10 コンピュータしか指定できません。IP を入力してから OK をクリックすると M10_Client.EXE と同じフォルダに設定ファイルが作成されます。

M10_Client.ini 例
[Servers]
Count = 1

[Server1]
Name = Server1
Ip = xxx.xxx.xxx <- 指定した IP
Port =yyy <- 指定したポート

複数の M10 コンピュータを指定するには、設定ファイルを手動で以下の通り設定してください。

設定ファイル

1. 設定ファイル名は m10_client.exe と同じフォルダの m10_client.ini
2. [Servers]セクションの Count で サーバ数を指定
3. 指定したサーバ数に応じて [Server1], [Server2] …,[ServerN]セクションで名前、 IP とポートを指定
4. ポートを指定しなければ既定はポート 12001

設定ファイル例：

[Servers]
Count = 3　← 3 つのサーバの情報が含まれる

[Server1]　<- サーバ 1
Name = Computer1
Ip = xxx.xxx.xxx

[Server2] <- サーバ 2
Name = NoteBook
Ip = xxx.xxx.xxx
Port=xxx

[Server3] <- サーバ 3
Name = NetBook
Ip = xxx.xxx.xxx

“複数サーバへの送信“では、設定ファイルのサーバ全てにキー操作文字列を送信します。”
複数サーバへの選択送信“では、設定ファイルのサーバがリストボックスに表示されます。
送信するサーバを指定して送信できます。

Visual Studio 2008　C#プロジェクト

M10 クライアントはユーザ側で作成可能です。M10 にはクライアントを作成するための.NET ライブラリ M10Lib.DLL とクライアントサンプルを収めた C#プロジェクトを添付していますので、このライブラリを利用することでクライアント開発の負担を軽減できます。

M10 ディスク

<クライアント>
M10_Client.exe
M10Lib.DLL
　　<M10_Client>　M10_Client プロジェクト
　　<M10_Client_Simple>　M10_Client_Simple プロジェクト
　　<M10_Client_Simple_WFP>　M10_Client_Simple_WFP プロジェクト

M10LIB

M10 サーバーとやり取りをするためのクラスライブラリファイルです。

M10 とやり取りするアプリケーションでは M10LIB.DLL ライブラリファイルへの参照を追加して利用します。

添付のサンプルプロジェクトは、すべて M10LIB を利用します。

M10 クライアントライブラリ - M10Lib.DLL

ライブラリ内のクラスのネームスペースは M10 です。

M10.Connection クラス public class Connection : IDisposable

M10 サーバーとのネットワーク接続を確立して、指定されたキーボード操作文字列を M10 サーバへ送るためのクラスです。複数のインスタンスを作成して、複数サーバにキーボード操作文字列を送信することが可能です。

プロパティ

No	int	読取のみ	インスタンスに割り当てる任意の番号。
Name	sring	読取のみ	インスタンスに割り当てる任意の名前
Port	In	読取/書込	接続する M10 の待ち受けるポート番号
ServerIP	sring	読取/書込	接続する M10 が実行しているコンピュータの IP アドレス
Status	bool	読取のみ	接続が確立されているかどうかを表すブール値
errCount	int	読取のみ	接続確立に失敗した回数
ReadTimeOut	int	書込のみ	M10 からの処理完了信号待ちのタイムアウト 単位はミリセカンド

コンストラクタ

Connection(int no, string name, int port, string IP)

作成するインスタンスの No & Name と接続する M10 サーバの Port & ServerIP を指定します。
コンストラクタでは渡された引数を保存するのみです。接続はしません。

Connection(Server svr)

Server の no, name, port, IP プロパティで Connection オブジェクトを作成します。
コンストラクタでは渡された引数を保存するのみです。接続はしません。

メソッド

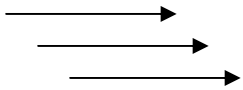
void SendString(string data)

data で指定したキーボード操作文字列を M10サーバーに送ります。もし、M10サーバとの接続

が確立されていなければ、接続してから送信します。古いバージョンとの互換性を保つためのメソッドです。内部では **uint SendString(string data, bool bWait, bool bCloseOnSent)**を **SendString(data, false, false)** で呼び出しています。

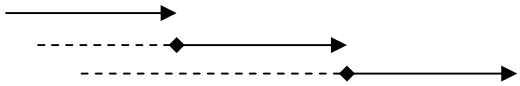
uint SendString(string data, bool bWait, bool bCloseOnSent)
data で指定したキーボード操作文字列を M10 サーバーに送ります。もし、M10 サーバとの接続が確立されていなければ、接続してから送信します。bWait を true にすると、M10 側でキーボード操作の再生が完了するまで制御を戻しません。false にすると、data を送信後、すぐに制御がもどります。bCloseOnSent を true にすると、data の送信後、M10 との接続を切断します。

bWait = false



SendStringは、すぐにM10に文字列を送信． 送信済みの文字列がM10側で再生されている最中でも、次の文字列は M10に送られます。

bWait = true



SendStringは、すでにM10で再生中の文字列があれば送信しません（破線の部分は送信待ちを現します）。M10で前の文字列の再生が完了したら、次の文字列が送信されます。

SendStringは、送信が完了するまで呼び出し側に制御を戻さないため、bWaitを true にする場合には、下の非同期版 **SendStringAsync**を利用するとアプリケーションのリスポンスが改善します。 また、bWaitをtrueにすると、M10側での再生終了の通知のため、クライアント側にM10から合図が送られます。このため外部からデータを受けることになるクライアント側では指定ポートを開ける必要があるかもしれません（セキュリティソフトによって、クライアント側から開始した通信ではデータを受けることを問題視しないこともありますし、また、明示的に許可を求めるものもあります。送信後に指定ポートの解除を求められた場合、内容を確認してM10関連のものであれば解除してください）。もし、セキュリティソフトを利用していてクライアント側のポートを開けることができなければ、bWaitには false を指定してください。

M10 の合図を永久に待ち続けるわけにはいかないため既定では送信 1 バイト当り 0.75 秒（100

バイト送信した場合、タイムアウトは 75 秒になります）でタイムアウトします。ただし文字列に¥16 が含まれる場合、 ¥16 の個数 x 3 秒が追加されます。このメソッドを呼び出す前に ReadTimeOut プロパティを設定することで、呼び出し側がタイムアウト値を指定できます。ReadTimeOut プロパティは **SendString/SendStringAsync** の呼出し毎にリセットされますので、設定する場合は呼び出す毎に設定してください。

```
conn.ReadTimeOut = 10000;       //10 秒（指定単位はミリ秒）
SendString(…)  or SendStringAsync(…)
// conn.ReadTimeOut は既定値に戻る

conn.ReadTimeOut = 15000;       // 15 秒
SendString(…)  or SendStringAsync(…)
```

戻り値： Connectionオブジェクトは **SendString / SendStringAsync**で文字列を送信する毎に 送信IDを割り当てます（0から 1 つずつ増加します）。**SendString / SendStringAsync**は、送信に割り当てられた送信IDを返します。

uint SendStringAsync(string data, bool bWait, bool bCloseOnSent)

SentString の非同期版です。data を送る処理を開始して（専用スレッドを開始して）すぐに制御を戻します。実際の送信は、**SendStringAsync** の呼び出し側とは独立して行われます。処理が完了すると OnSendCompleted イベントで通知します。**SendStringAsync** を呼び出す側では、呼び出す前にイベントハンドラーを登録することで、送信処理の完了を知ることができます。

bWait を true にすると、M10 側のキーボード操作再生処理が完了するまで、イベントハンドラーは呼び出されません。M10 から処理完了の信号待ちタイムアウトについては SendString を参照してください。。

bWait を false にすると、送信後キー操作再生の完了を待たずにハンドラーが呼び出されます。

bCloseOnSent を true にすると、イベントハンドラーを呼び出す前に M10 との接続を切断します。

戻り値： Connectionオブジェクトは **SendString / SendStringAsync**で送信する毎に 0から送信IDを割り当てます。**SendString / SendStringAsync**の送信IDを返します。

備考: **SendStringAsync**はスレッドプールを使います。スレッドプールは1プロセスあたり25スレッドという制限があるようなので、その制限にかからないようご注意ください（**SendStringAsync** を連続して呼び出すと、呼び出した順番で送信が行われます。5回呼び出し、1回目の呼び出しがなんらかの理由で M10からの処理終了信号を待ち続けると、その間、2回目以降の呼び出し4つは、1回目の呼び出しが完了するまで待ち続けます。この状態でさらに **SendStringAsync**を呼び出すと、さらに待ちスレッドが増えます　）。

イベント

public event NotifySendCompletedHandler **OnSendCompleted**

public delegate void NotifySendCompletedHandler**(object sender,**
OnSendCompletedEventArgs **e);**

SendStringAsyncで開始した送信処理が完了したときに生成されるイベントです。
OnSendCompletedEventArgsに終了の状態が設定されます。以下を含みます。

uint id　　…　送信処理が完了した送信ID
bool status　　…　送信成功ならば　true, 失敗ならば false
string msg　　.. status　が　falseの場合、失敗した説明
int connNo　　..　送信処理をしたConnectionオブジェクトの　No
string connName …送信処理をしたConnectionオブジェクトの　Name

void SendReset()

M10 内部の状態をリセットするコードを送信します。もし、M10 サーバとの接続が確立されていなければ、接続してから送信します。

void Dispose()　/　void CloseNet()

M10 サーバとの接続を閉じます。

M10.Server クラス　public class Server

M10 サーバに関する情報(Port & IP)を保持するクラスです。コンストラクタとプロパティしかありません。

プロパティ

No	int	読取のみ	インスタンスに割り当てる任意の番号。
Name	sring	読取/設定	インスタンスに割り当てる任意の名前
Port	In	読取/設定	接続する M10 の待ち受けるポート番号
IP	sring	読取/設定	接続する M10 が実行しているコンピュータの IP アドレス

コンストラクタ

Server(int no, string name, int port, string IP)

M10.Servers クラス　　public class Servers : IEnumerable<Server>

アプリケーションと同じフォルダにある設定ファイル（ アプリケーション名ファイル+.INI(拡張子)）から、設定されているM10サーバのPort & IPを読み取ります。設定ファイルでは複数のサーバの Port & IP を指定可能です。利用するプログラムは、Serversが読み取った複数のサーバ情報を配列構文 [] で取得できます。 設定ファイルから読み取ったサーバの番号、名前、port、IPを Serverクラスのインスタンスとして返します。

設定ファイル例：

[Servers]
Count = 3　　← 3 つのサーバの情報が含まれる

[Server1]　<- サーバ 1
Name = Computer1
Ip = xxx.xxx.xxx

[Server2] <- サーバ 2
Name = NoteBook
Ip = xxx.xxx.xxx
Port=xxx

[Server3] <- サーバ 3
Name = NetBook
Ip = xxx.xxx.xxx

プロパティ

Count	int	読取のみ	読み取ったサーバ情報数

コンストラクタ

Servers()
インスタンス構築時に設定ファイルを読み込みます。

インデクサ []
読み取ったサーバ情報を配列構文で取得できます。戻り値の型は　Server です。また、Servers インスタンスに foreach 構文を適用することで、Serverインスタンスを取得することもできます。

M10.XmlServers クラス　　public class XmlServers : Servers

機能はServersと同じですが、読み取る対象が .INIファイルではなく XMLファイルです。

XML 設定ファイルの書式例

```
<Servers Count="3">
        <Server no="1">
                <name>111</name>
                <ip>192.168.2.78</ip>
        </Server>
        <Server no="2">
                <name>222</name>
                <ip>192.168.2.80</ip>
        </Server>
        <Server no="3">
                <name>333</name>
                <ip>192.168.2.81</ip>
                <port>10000</port>
        </Server>
</Servers>
```

プロパティ

Count	int	読取のみ	読み取ったサーバ情報数

コンストラクタ

XmlServers()

インスタンス構築時に設定ファイルを読み込みます。

インデクサ []

読み取ったサーバ情報を配列構文で取得できます。戻り値の型は　Server　です。また、Servers インスタンスに foreach 構文を適用することで、Serverインスタンスを取得することもできます。

M10.ServerSetting　public class ServerSetting

アプリケーションと同じフォルダにある設定ファイル（ アプリケーション名ファイル+.INI(拡張子)）から、設定されているM10サーバのPort & IPを読み取ります。設定ファイルでは１つのサーバを指定します。

設定ファイル例
[Setting]
Ip=192.168.1.20
Port=12001

プロパティ

M10Server	Server	読取のみ	読み取ったサーバ情報

コンストラクタ

ServerSetting()

インスタンス構築時に設定ファイルを読み込み、Sever インスタンスを作成します。

M10_Client_Simple プロジェクト

M10Lib を利用した簡単な Windows フォームアプリです。

1. Form1_Load で 設定ファイルを読み込み Connection オブジェクトを作成します
2. [送信]ボタンのクリックで、テキストボックスの文字列を Connection の SendString で送信します
3. Form1_Closing で Connection のリソースを解放します。

プロジェクトのプロパティービルドー条件付コンパイルシンボルで ASYNC_CALL を定義すると非同期で送信が行われます。SEND_WAIT を定義すると送信待ちオプションが有効になります。

M10Lib.DLL への参照が必要です。

M10_Client_Simple_WPF プロジェクト

M10_Client_Simple とほぼ同じですが、WPF 版です。UI 以外の処理は、M10_Client_Simple と同じです。

プロジェクトのプロパティービルドー条件付コンパイルシンボルで ASYNC_CALL を定義すると非同期で送信が行われます。SEND_WAIT を定義すると送信待ちオプションが有効になります。

M10Lib.DLL への参照が必要です。

M10_Client プロジェクト

M10_Client のソースです。次のファイルが含まれます。

1 つのクラスファイル

M10controller.cs(SingleConnectionController, MultipleConnectionController)

6 つのフォーム(UI)

Form1.cs
Form2.cs
Form3.cs
M10ClientForm.cs
SelectForm.cs
SetIP.cs

SelectForm は、起動直後に表示されるフォームです。どのフォームで送信するかを選択するためのフォームです。

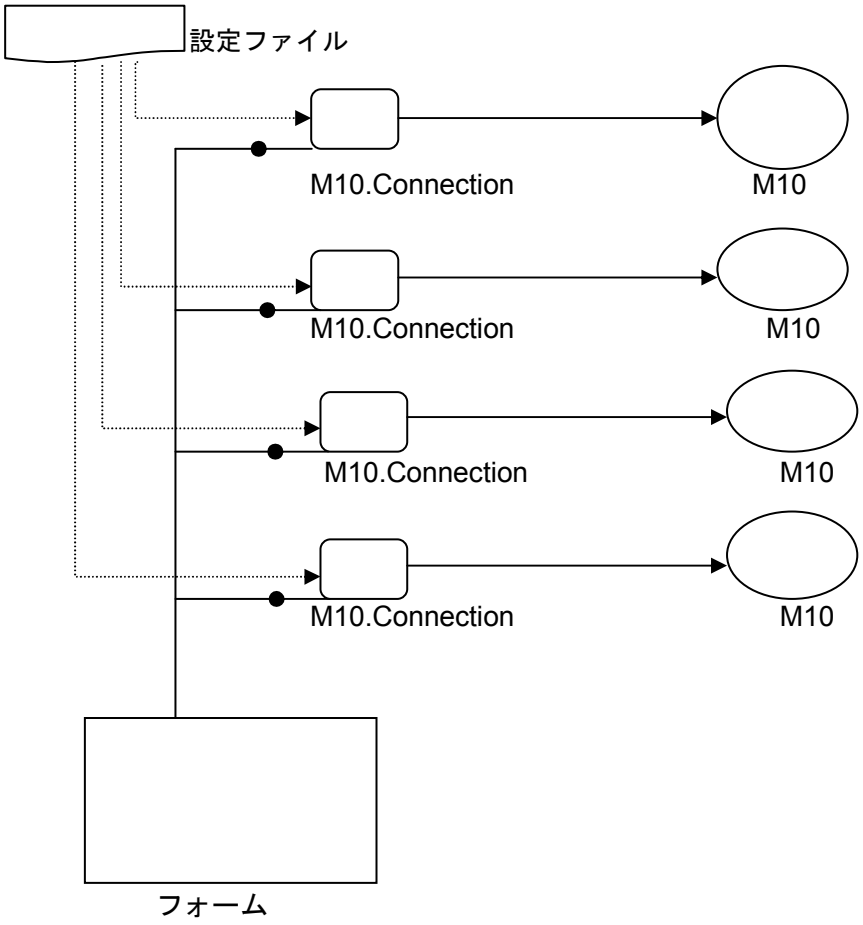
M10ClientForm は Form1,Form2,Form3 のベースフォームで、Form1,Form2,Form3 が共通して持つ virtual メソッドを定義します。

M10Controller.cs は m10conn, m10servers, m10setting クラスから必要なオブジェクトを作成して、Form1,Form2,Form3 とやり取りします。

プロジェクトのプロパティービルドー条件付コンパイルシンボルで ASYNC_CALL を定義すると非同期で送信が行われます。SEND_WAIT を定義すると送信待ちオプションが有効になります。

M10Lib.DLL への参照が必要です。

Form1.cs (複数サーバにキーボード操作文字列を送信　)



a. 設定ファイルに指定されているサーバ(IP + Port)分の Connection オブジェクト作成 (MultipleConnectionController)
b. フォームで指定されたキーボード操作文字列を 全ての Connection オブジェトに渡して送信(Form1)

Servers クラスから派生する **XmlServers** クラスを使うと XML 設定ファイルを利用します。

M10controller.cs の 2 行

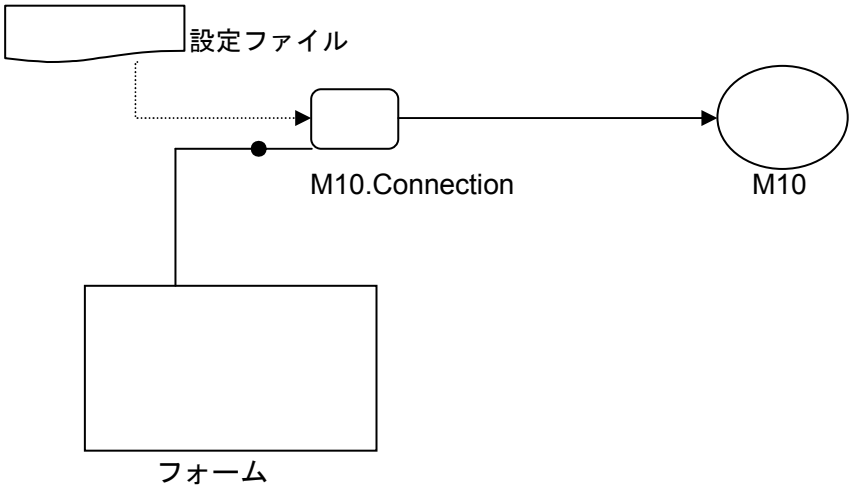
```
public MultipleConnectionsController()
 {
      M10.Servers svrs = null;

      try
      {
           //設定ファイル読み込み
           svrs = new M10.Servers();

           // XML設定ファイルを利用する
           svrs = new M10.XmlServers();
      }
      catch( Exception ex )
      {
           MessageBox.Show(ex.Message, M10ClientUtil.AppName,
                                MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

           return;
      }
```

Form2.cs (単サーバにキーボード操作文字列を送信　)

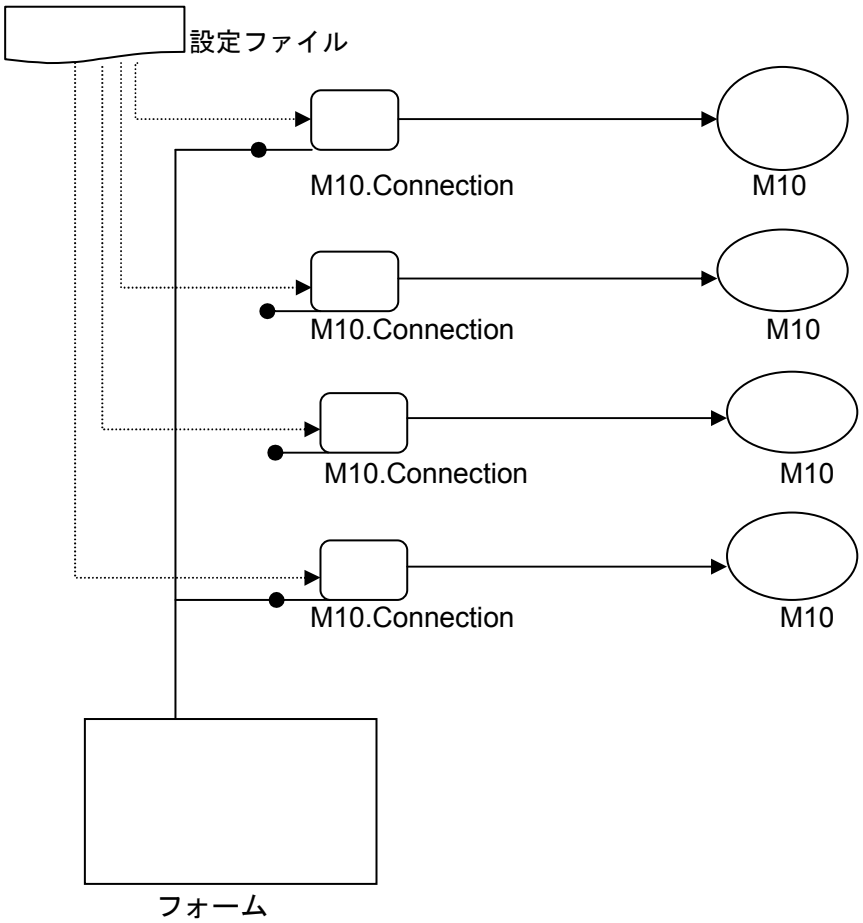


a. 設定ファイルに指定されているサーバ(IP + Port)の Connection オブジェクト作成 (SingleConnectionController)
b. フォームで指定されたキーボード操作文字列を Connection オブジェトに渡して送信

ServerSetting クラスで設定ファイルの[Setting] IP,Port を処理します。

Form3.cs (複数のサーバから送信対象を選択して操作文字列を送信　)

Form1.cs とほぼ同じですが、フォームのリストボックスに設定ファイルのサーバ名を表示。リストボックスで選択したサーバにのみ送信



a. 設定ファイルに指定されているサーバ(IP + Port)分の Connection オブジェクト作成
b. フォームで指定されたキーボード操作文字列を、リストボックスで選択されたサーバへの Connection オブジェトに渡して送信

Form1.cs と大きく異なる点は、Form1 では初めて文字列を送信するときに、M10 サーバとの接続が確立され、フォームを閉じるまで接続は維持されます。一方、Form3.cs では文字列を送信する度に、M10 サーバとの接続を確立し、送信後、接続を切断します（リストボックスで接続対象が変更可能なため、接続を維持せずに都度切断します）

仮想キーコード表

M10 ですべての仮想キーコードが利用できるわけではありません。特に SHIFT, CTRL, ALT の組み合わせは仮想キーコードでは指定できません。

以下は、参考のための仮想キーコード表です。
http://api.farmanager.com/en/winapi/virtualkeycodes.html

16 進コード	キー操作
01	Left mouse button
02	Right mouse button
03	Control-break processing
04	Middle mouse button (three-button mouse)
05	**Windows 2000/XP/2003/Vista/2008/7:** X1 mouse button
06	**Windows 2000/XP/2003/Vista/2008/7:** X2 mouse button
07	Undefined
08	BACKSPACE key
09	TAB key
0A-0B	Reserved
0C	CLEAR key
0D	ENTER key
0E-0F	Undefined
10	SHIFT key
11	CTRL key
12	ALT key
13	PAUSE key
14	CAPS LOCK key
15	Input Method Editor (IME) Kana mode
15	IME Hanguel mode (maintained for compatibility; use **VK_HANGUL**)

15	IME Hangul mode
16	Undefined
17	IME Junja mode
18	IME final mode
19	IME Hanja mode
19	IME Kanji mode
1A	Undefined
1B	ESC key
1C	IME convert (Reserved for Kanji systems)
1D	IME nonconvert (Reserved for Kanji systems)
1E	IME accept (Reserved for Kanji systems)
1F	IME mode change request (Reserved for Kanji systems)
20	SPACEBAR
21	PAGE UP key
22	PAGE DOWN key
23	END key
24	HOME key
25	LEFT ARROW key
26	UP ARROW key
27	RIGHT ARROW key
28	DOWN ARROW key
29	SELECT key
2A	PRINT key
2B	EXECUTE key
2C	PRINT SCREEN key for Windows 3.0 and later
2D	INS key
2E	DEL key
2F	HELP key

30	0 key
31	1 key
32	2 key
33	3 key
34	4 key
35	5 key
36	6 key
37	7 key
38	8 key
39	9 key
3A-40	Undefined
41	A key
42	B key
43	C key
44	D key
45	E key
46	F key
47	G key
48	H key
49	I key
4A	J key
4B	K key
4C	L key
4D	M key
4E	N key
4F	O key
50	P key
51	Q key

52	R key
53	S key
54	T key
55	U key
56	V key
57	W key
58	X key
59	Y key
5A	Z key
5B	Left Windows key (Microsoft Natural Keyboard)
5C	Right Windows key (Microsoft Natural Keyboard)
5D	Applications key (Microsoft Natural Keyboard)
5E	Reserved
5F	Computer Sleep key
60	Numeric keypad 0 key
61	Numeric keypad 1 key
62	Numeric keypad 2 key
63	Numeric keypad 3 key
64	Numeric keypad 4 key
65	Numeric keypad 5 key
66	Numeric keypad 6 key
67	Numeric keypad 7 key
68	Numeric keypad 8 key
69	Numeric keypad 9 key
6A	Multiply key
6B	Add key
6C	Separator key
6D	Subtract key

6E	Decimal key
6F	Divide key
70	F1 key
71	F2 key
72	F3 key
73	F4 key
74	F5 key
75	F6 key
76	F7 key
77	F8 key
78	F9 key
79	F10 key
7A	F11 key
7B	F12 key
7C	F13 key
7D	F14 key
7E	F15 key
7F	F16 key
80H	F17 key
81H	F18 key
82H	F19 key
83H	F20 key
84H	F21 key
85H	F22 key
86H	F23 key
87H	F24 key
88-8F	Unassigned
90	NUM LOCK key

91	SCROLL LOCK key
92	NEC PC-9800 kbd definitions: '=' key on numpad
92	Fujitsu/OASYS kbd definitions: 'Dictionary' key
93	Fujitsu/OASYS kbd definitions: 'Unregister word' key
94	Fujitsu/OASYS kbd definitions: 'Register word' key
95	Fujitsu/OASYS kbd definitions: 'Left OYAYUBI' key
96	Fujitsu/OASYS kbd definitions: 'Right OYAYUBI' key
97-9F	Unassigned
A0	Left SHIFT key
A1	Right SHIFT key
A2	Left CONTROL key
A3	Right CONTROL key
A4	Left MENU key
A5	Right MENU key
A6	**Windows 2000/XP/2003/Vista/2008/7:** Browser Back key
A7	**Windows 2000/XP/2003/Vista/2008/7:** Browser Forward key
A8	**Windows 2000/XP/2003/Vista/2008/7:** Browser Refresh key
A9	**Windows 2000/XP/2003/Vista/2008/7:** Browser Stop key
AA	**Windows 2000/XP/2003/Vista/2008/7:** Browser Search key
AB	**Windows 2000/XP/2003/Vista/2008/7:** Browser Favorites key
AC	**Windows 2000/XP/2003/Vista/2008/7:** Browser Start and Home key
AD	**Windows 2000/XP/2003/Vista/2008/7:** Volume Mute key
AE	**Windows 2000/XP/2003/Vista/2008/7:** Volume Down key
AF	**Windows 2000/XP/2003/Vista/2008/7:** Volume Up key
B0	**Windows 2000/XP/2003/Vista/2008/7:** Next Track key
B1	**Windows 2000/XP/2003/Vista/2008/7:** Previous Track key
B2	**Windows 2000/XP/2003/Vista/2008/7:** Stop Media key
B3	**Windows 2000/XP/2003/Vista/2008/7:** Play/Pause Media key

B4	**Windows 2000/XP/2003/Vista/2008/7:** Start Mail key
B5	**Windows 2000/XP/2003/Vista/2008/7:** Select Media key
B6	**Windows 2000/XP/2003/Vista/2008/7:** Start Application 1 key
B7	**Windows 2000/XP/2003/Vista/2008/7:** Start Application 2 key
B8-B9	Reserved
BA	**Windows 2000/XP/2003/Vista/2008/7:** For the US standard keyboard, the ';:' key
BB	**Windows 2000/XP/2003/Vista/2008/7:** For any country/region, the '+' key
BC	**Windows 2000/XP/2003/Vista/2008/7:** For any country/region, the ',' key
BD	**Windows 2000/XP/2003/Vista/2008/7:** For any country/region, the '-' key
BE	**Windows 2000/XP/2003/Vista/2008/7:** For any country/region, the '.' key
BF	**Windows 2000/XP/2003/Vista/2008/7:** For the US standard keyboard, the '/?' key
C0	**Windows 2000/XP/2003/Vista/2008/7:** For the US standard keyboard, the '`~' key
C1-D7	Reserved
D8-DA	Unassigned
DB	**Windows 2000/XP/2003/Vista/2008/7:** For the US standard keyboard, the '[{' key
DC	**Windows 2000/XP/2003/Vista/2008/7:** For the US standard keyboard, the '¥|' key
DD	**Windows 2000/XP/2003/Vista/2008/7:** For the US standard keyboard, the ']}' key
DE	**Windows 2000/XP/2003/Vista/2008/7:** For the US standard keyboard, the 'single-quote/double-quote' key
DF	Used for miscellaneous characters; it can vary by keyboard.
E0	Reserved
E1	OEM specific
E2	**Windows 2000/XP/2003/Vista/2008/7:** Either the angle bracket key or the backslash key on the RT 102-key keyboard
E3-E4	OEM specific
E5	**Windows 95/98/Me, Windows NT/2000/XP/2003/Vista/2008/7:** IME PROCESS key
E6	OEM specific
E7	**Windows 2000/XP/2003/Vista/2008/7:** Used to pass Unicode characters as if they were keystrokes. The VK_PACKET key is the low word of a 32-bit Virtual Key value used for

	non-keyboard input methods. For more information, see Remark in KEYBDINPUT , SendInput , WM_KEYDOWN , and WM_KEYUP
E8	Unassigned
E9	Only used by Nokia.
EA	Only used by Nokia.
EB	Only used by Nokia.
EC	Only used by Nokia.
ED	Only used by Nokia.
EE	Only used by Nokia.
EF	Only used by Nokia.
F0	Only used by Nokia.
F1	Only used by Nokia.
F2	Only used by Nokia.
F3	Only used by Nokia.
F4	Only used by Nokia.
F5	Only used by Nokia.
F6	Attn key
F7	CrSel key
F8	ExSel key
F9	Erase EOF key
FA	Play key
FB	Zoom key
FC	Reserved for future use.
FD	PA1 key
FE	Clear key