対 MARK30 通信　ActiveX コントロール「Mk30Comm.OCX」

外部仕様書

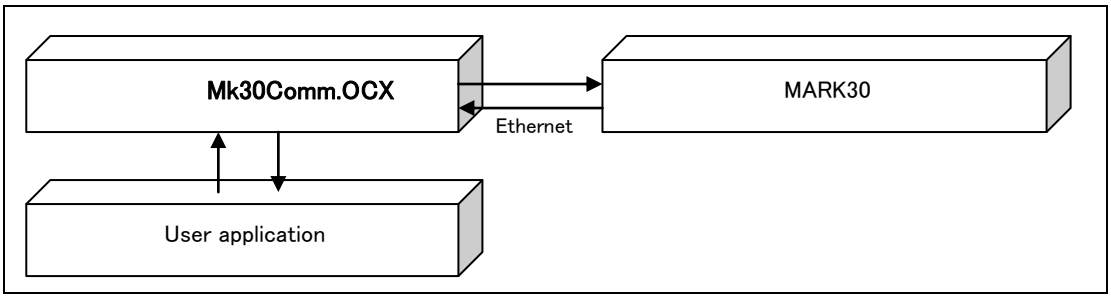
＜目次＞

1　概要

SODICK 社製 NC 加工機「MARK30」と WindowsNT4.0 を OS とする PC との通信を ActiveX の形態で実現し、PC から MARK30 のリモート制御を可能にする。

プログラム名	Mk30Comm.OCX
対応する NC 加工機	SODICK MARK30 形彫・ワイヤ
対応する OS	Microsoft WindowsNT 4.0 (SP3)
対応する PC	上記 OS が稼動している PC
対応言語	Microsoft VisualBasic 5.0 / 6.0 Microsoft Visual C++ 5.0
Ethernet 対応	PC と加工機とのインターフェースに Ethernet を利用する
プロトコル	Winsock ※　Winsock を起動するには、MARK30 に対し HTTP 接続されていなければならない。あらかじめ IE をユーザが起動するか User application で HTTP 接続する処理を行う必要がある。
機能	ファイル転送 ファイル削除 プログラム実行指示 加工履歴ファイルの取得 補正項ファイルの取得

＜構成＞



2　ファイル構成

・MK30COMM.OCX	ActiveX OCX 本体
・MK30COMM.EXP	エクスポート（.EXP）ファイル。 エクスポートされる関数やデータ項目に関する情報が含まれる。

3　セットアップ

3.1　MK30COMM.OCX のセットアップ手順

① MK30COMM.OCX、MK30COMM.EXP を適当なフォルダにコピーします
この 2 つのファイルは同じフォルダにコピーする必要があります。

② コマンドプロンプトを起動します。

③ REGSVR32.EXE を実行します。
2 つのファイルを C:¥TEMP にコピーした場合、
REGSVR32 C:¥TEMP¥MK30COMM.OCX
と実行します。

以上で OCX がクライアントで使用できるようになります。

3.2　セットアップに失敗する原因

セットアップに失敗する原因は様々ですが、基本的に以下の 2 点が主な原因です。

・REGSVR32.EXE がコントロールを確認することができない。

・必要な OLE サポート DLL ファイルが不足している。

セットアップに失敗する場合は、上記項目を確認してください。

3.3　バージョンアップ

バージョンアップ時も、「3.1MK30COMM.OCX のセットアップ手順」と同様の操作を行ないます。ただし、旧バージョンと同じフォルダにコピーしてください。アプリケーションからの参照設定に失敗する可能性があります。

その逆を行なう場合も同様です。

3.4　OLE サポート DLL について及び注意事項

ActiveX(OCX) コントロールを使用する場合は、コントロール以外に OLE サポート DLL が必要です。

MK30COMM.OCX の動作に必要な OLE サポート DLL を以下に示します。

MFC42.DLL、MSVCRT.DLL、OLEPRO32.DLL

OLE サポート DLL には、異なるバージョンがありますので、どのバージョンを使用しているのか、インストール先の管理に十分注意して下さい。

詳細については、MSDN 等、MFC 関連のドキュメントを参照してください。

4 プロパティ／メソッド／イベント

4.1 プロパティ／メソッド／イベント一覧

プロパティ

名称	機能
MachineStatus	MARK30 の機械状態をチェック
MyComputer	クライアント PC のコンピュータ名の参照
Password	クライアント PC にある共有フォルダにアクセスするためのユーザパスワードを設定
PowerOn	MARK30 のパワー状態をチェック
RasPassword	MARK30 にある共有フォルダにアクセスするためのユーザパスワードを設定
RasUserName	MARK30 にある共有フォルダにアクセスするためのユーザ名を設定
UserName	クライアント PC にある共有フォルダにアクセスするためのユーザ名を設定
FtpUserName	クライアント PC にある FTP フォルダにアクセスするためのユーザ名を設定
FtpPassword	クライアント PC にある FTP フォルダにアクセスするためのユーザパスワードを設定

メソッド

名称	機能
Connect	MARK30 への接続を行う
DeleteFile	MARK30 にあるファイルを削除する
Disconnect	MARK30 からの接続解除を行う
Execute	MARK30 へプログラム実行指示を行う
GetHistory	MARK30 にある最新の加工履歴ファイルを取得する
GetLastError	最後に発生したエラー番号を取得する（デバッグ向け）
GetOffset	MARK30 にある補正項ファイルを取得する
GetVersion	OCX のバージョン番号を取得する
LogEnd	通信ログを終了する（デバッグ向け）
LogStart	通信ログを開始する（デバッグ向け）
Pause	MARK30 への接続を一時的に解除・再開を行う
SendEmKey	MARK30 へ OFF/HALT/ACK キーを送信する
SendFile	MARK30 へファイル転送を行う
GetVoltage	電圧値を取得する
GetElectricity	電流値を取得する
GetFSpeed	F 値を取得する
GetFrequency	頻度値を取得する
GetMaxFrequency	頻度サーボ頻度数最大値を取得する
GetSigOnMaxFrequency	頻度サーボ頻度数最大値時の SIG 値を取得する
GetResistivity1	比抵抗 1 を取得する
GetResistivity2	比抵抗 2 を取得する
GetInch	Inch/mm の状態を取得する
GetDigit	digit 値を取得する
GetCoordSys	カレント座標系を取得する。
GetCoordOrg	座標系 nCoordSys の原点を取得する
GetMachCoord	機械座標値を取得する
GetCoord	座標値を取得する
GetCondition	加工条件を取得する
GetMachineKind	機種を取得する
GetConditionType	加工条件のタイプを取得する（33WS/非 33WS）
CreateNcCond	条件検索機能
GetHiVersion	HI（操作部）のバージョンを取得する
GetRunningFileName	実行中の NC ファイル名、パスタイプを取得する。サブプログラム実行中時は、サブプログラムの NC ファイル名を取得する。

GetRunningMainNcFileFirstLine	メイン NC プログラムの 1 ブロック目を取得する。
GetNcFileRunEventTime	前回と今回の加工時間を取得する。
FtpSendFile	FTP 経由で、NC ファイルを、PC から、機械の「メモリ」に Send する。
GetCurrentLogonUserInfo	機械側で、ログインされたユーザーの情報を取得する。
FarLogout	遠隔（PC 端）からログアウト。
FarLogon	遠隔（PC 端）からログイン。
GetUserLogonInfFile	ユーザーログイン、ログアウトの履歴 Log を取得する。
GetMachineLock	Machine Lock の状態を取得する。
GetMachineLock	Machine Lock の状態を取得する。
GetOffsetVar	Offset データを取得する
SetMacroVar	Macro 変数に、データをセットする
GetMacroVar	Macro 変数から、データを取得する
GetMacroVarRange	多数個マクロ変数から、データを取得する（最大連続の 100 個まで）（VC で使用できるインタフェース）
GetMacroVarRangeSafeArray	多数個マクロ変数から、データを取得する（最大連続の 100 個まで）（VB で使用できるインタフェース）
SetMacroVarRange	多数個マクロ変数に、データをセットする（最大連続の 100 個まで）（VC で使用できるインタフェース）
SetMacroVarRangeSafeArray	多数個マクロ変数に、データをセットする（最大連続の 100 個まで）（VB で使用できるインタフェース）
GetUserLogonInfFile	ユーザーログイン、ログアウト履歴の取得 LOG ファイル
GetRamFileList	RamList ファイル名のリストを取得する。
SendFileFromFolder	pcshare のサブディレクトリにあるファイルをマシンの Ram ディレクトリに送り、マシンの Ram リストを追加する。
SetActiveNCProgram	マシンの Ram リストのプログラムを実行ファイルとして選択する。
GetRemainLength	加工残量を取得する。（単位:mm、戻り値は double 型）
GetBlockRemainLength	1 ブロック残量を取得する。（単位:mm、戻り値は double 型）
GetPowerOnSeconds	機械の累積ソースオン時間を取得する。
GetWorkingSeconds	機械の累積加工時間を取得する。
GetActiveNCProgram	現在の NC プログラムを取得する。
DeleteAllFiles	NewFile を除く、Ram ディレクトリ内のすべてのファイルを削除する。
GetCutInfo	加工情報を取得する。
SetLnProDataProcessing	LnPro 用の加工パラメータのデータをセットする。
LoadHDiskFile	H_Disk のファイルを RAM にロードするようにして、ラン画面に移動
GetEleOfsSettings	電極補正関連の設定を読み取る
GetEleOfs	電極補正データを読み取る
SetEleOfs	電極補正データを書き込む

イベント

名称	機能
DeleteFileComplete	MARK30 にあるファイルを削除後、削除完了時に発生
DenyRemoteAccess	MARK30 が遠隔実行を認めない場合に発生
ExecuteComplete	MARK30 で NC プログラム実行終了時に発生
NcError	MARK30 でエラー・ハルト時に発生
NetworkError	ネットワークエラー時に発生
SendFileComplete	MARK30 へファイル転送後、転送完了時に発生
MachineStatusChanged	MARK30 から MachineStatus 変更する時に発生
EventLnProProcessResult	LnPro が関数実行後の時に、CimForce に通知するイベント
LoadHDiskFileComplete	メソッド LoadHDiskFile の実行終了時に発生

※　プロパティ／メソッド／イベントは、今後の機能拡張により変更、追加されることがあります。

4.2　プロパティ

MachineStatus

機能　MARK30 の機械状態をチェック

書式　*intMachineStatus = mMk30Comm.* **MachineStatus**

設定値　Integer 型

0	READY
1	実行中
2	HALT 中
3	ACK 待ち
（デフォルト値）	-1

用法　設計時：なし
実行時：参照のみ

解説

MyComputer

機能　クライアント PC のコンピュータ名の参照

書式　*mMk30Comm.* **MyComputer** *= strMyComputer*

設定値　文字列型
クライアントのコンピュータ名

（デフォルト値）	クライアントのコンピュータ名（ホスト名）

用法　設計時：なし
実行時：参照のみ

解説　・このプロパティは、MARK30 がクライアントコンピュータにアクセスする際に使われる
・クライアントのコンピュータ名（ホスト名）は OCX 起動時に自動的に取得される

Password

機能　　クライアント PC にある共有フォルダにアクセスするためのユーザパスワードを設定

書式　　*mMk30Comm*. **Password** = *strPasswd*

設定値　　文字列型
ユーザパスワード
（デフォルト値）　　“”

用法　　設計時：設定及び参照
実行時：設定及び参照

解説　　・このプロパティは、MARK30 がクライアントコンピュータにアクセスする際に使われる

PowerOn

機能　　MARK30 のパワー状態をチェック

書式　　*blnPowerOn = mMk30Comm*. **PowerOn**

設定値　　Boolean 型

True	POWER ON
False	POWER OFF
（デフォルト値）	True

用法　　設計時：なし
実行時：参照のみ

解説

RasPassword

機能　　MARK30 にある共有フォルダにアクセスするためのユーザパスワードを設定

書式　　*mMk30Comm*. **RasPassword** = *strPasswd*

設定値　　文字列型
ユーザパスワード
（デフォルト値）　　“enkaku”

用法　　設計時：設定及び参照
実行時：設定及び参照

解説　　・このプロパティは、MARK30 にある共有フォルダへのアクセスに使われる

RasUserName

機能　　MARK30 にある共有フォルダにアクセスするためのユーザ名を設定

書式　　*mMk30Comm*. **RasUserName** = *strUserName*

設定値　　文字列型
ユーザ名
（デフォルト値）　　“rasperson”

用法　　設計時：設定及び参照
実行時：設定及び参照

解説　　・このプロパティは、MARK30 にある共有フォルダへのアクセスに使われる

UserName

機能　　クライアント PC にある共有フォルダにアクセスするためのユーザ名を設定

書式　　*mMk30Comm*.**UserName** = *strUserName*

設定値　　文字列型
ユーザ名
（デフォルト値）　　現在ログオンしているユーザ名

用法　　設計時：設定及び参照
実行時：設定及び参照

解説　　・このプロパティは、MARK30 がクライアントコンピュータにアクセスする際に使われる

FtpUserName

機能　　FTP ユーザー名の設定

書式　　*mMk30Comm*.Ftp**UserName** = *strUserName*

設定値　　文字列型
FTP ユーザー名

用法

解説　　・FtpSendFile は呼び出し前に設定されている

FtpPassword

機能　　FTP ユーザーのパスワードを設定する

書式　　*mMk30Comm*.Ftp**Pasword** = *strPassword*

設定値　　文字列型
FTP ユーザーのパスワード　　現在ログインするときのユーザー名

用法

解説　　・FtpSendFile は呼び出し前に設定されている

4.3　メソッド

Connect

機能　MARK30 への接続を行う

書式　*intError = mMk30Comm.* **Connect**(*strMachineName*)

引数	strMachineName$	接続する機械名（コンピュータ名）文字列

戻り値　Integer 型

−1	接続成功
0	接続失敗

解説
- あらかじめ MARK30 に対し HTTP 接続されている必要がある
- 接続に失敗した場合、**GetLastError** メソッドを実行することにより、エラー番号が取得できる

DeleteFile

機能　MARK30 にあるファイルを削除する

書式　*intError = mMk30Comm.* **DeleteFile**(*strFileName*)

引数	strFileName$	削除するファイル名（拡張子を除いたもの）文字列

戻り値　Integer 型

−1	削除成功
0	削除失敗

解説
- このメソッドが完了した後、**DeleteFileComplete** イベントが発生する
- 戻り値は、引数に不正な文字列が使用された場合等、ファイル削除実行前に発生したエラーに対して返される
- ファイル削除実行時に発生したエラーは **DeleteFileComplete** イベントにおいて引数でエラー番号が渡される

Disconnect

機能　MARK30 との接続を解除する

書式　*mMk30Comm*. **Disconnect**( )

引数　なし

戻り値　なし

解説

Execute

機能　MARK30 へプログラム実行指示を行う

書式　*intError = mMk30Comm*. **Execute**(*strCommand*)

引数	strCommand$	実行指示文字列（"ENKAKU"）
戻り値	Integer 型	
	−1	実行成功
	0	実行失敗

解説
- このメソッドを実行する前に、ENKAKU.NC ファイルを MARK30 へ転送しておく必要がある
- このメソッドが完了した後、**ExecuteComplete** イベントが発生する
- 戻り値が 0 の場合、引数に不正な文字列が使用された場合等、プログラム実行指示前に発生したエラーに対して返される。
  この場合、**GetLastError** メソッドを実行することにより、エラー番号が取得できる
- プログラム実行指示時に発生したエラーは **ExecuteComplete** イベントにおいて引数でエラー番号が渡される
- ENKAKU.NC ファイルの内容は、MARK30 の RAM にある実体ファイルを Q コマンド形式で記述します。（例） QTEST(0.000,10.000);

GetHistory

機能　MARK30 にある最新の加工履歴ファイルを取得する

書式　*ret = mMk30Comm.* **GetHistory**(*strHistory*)

引数

strHistory$	取得した加工履歴ファイルの内容文字列（出力）

戻り値　Boolean 型

True	取得成功
False	取得失敗

解説
- このメソッドを実行する前に、**RasUserName**、**RasPassword** プロパティを設定する必要がある
- クライアント側で、NC 実行中はファイル読み込みはしてはいけない
- ファイル取得時の Max サイズは無制限とする
- ファイル読み込みは、FILE_SHARE_READ | FILE_SHARE_WRITE で行う
- ファイル取得時は、最近加工したもののみ取得する
- 取得失敗の場合、**GetLastError** メソッドを実行することにより、エラー番号が取得できる

GetLastError

機能　最後に発生したエラー番号を取得する（デバッグ向け）

書式　*ret = mMk30Comm.* **GetLastError**

引数　なし

戻り値　Integer 型
エラー番号

解説　戻り値の詳細は、＜4.6GetLastError メソッドエラー一覧＞を参照

GetVersion

機能　OCX のバージョン番号を取得する

書式　*strVer = mMk30Comm.* **GetVersion**( )

引数　なし

戻り値　CString 型
バージョン番号

解説　なし

LogEnd

機能　　通信ログを終了する（デバッグ向け）

書式　　*mMk30Comm.* **LogEnd**

引数　　なし

戻り値　　なし

解説　　・LogStart を実行した場合、必ずこのメソッドを実行してください

LogStart

機能　　通信ログを開始する（デバッグ向け）

書式　　*mMk30Comm.* **LogStart**

引数　　なし

戻り値　　BOOL 型

True	開始成功
False	開始失敗

解説　　・　ログファイルは、TEMP または TMP フォルダに作成される

GetOffset

機能　MARK30 にある補正項ファイルを取得する

書式　*ret = mMk30Comm.* **GetOffset**(*strOffset*)

引数		
	strOffset$	取得した補正項ファイルの内容文字列（出力）

戻り値		
	Boolean 型	
	True	取得成功
	False	取得失敗

解説
- このメソッドを実行する前に、**RasUserName**、**RasPassword** プロパティを設定する必要がある
- クライアント側で、NC 実行中はファイル読み込みはしてはいけない
- ファイル取得時の Max サイズは無制限とする
- ファイル読み込みは、FILE_SHARE_READ | FILE_SHARE_WRITE で行う
- 取得失敗の場合、**GetLastError** メソッドを実行することにより、エラー番号が取得できる

Pause

機能　MARK30 への接続を一時的に解除・再開を行う

書式　*mMk30Comm.* **Pause**(*blnPause*)

引数		
	blnPause	MARK30 への接続を一時的に解除・再開を指定 True （接続解除） False （接続再開）

戻り値　なし

解説
- このメソッドは、MARK30 と PC 間で通信中、MARK30 での人的作業が必要になったときに使用する
- このメソッドは、MARK30 からの通知を全て無視する機能である

SendEmKey

機能　MARK30 へ OFF/HALT/ACK キーを送信する

書式　*intError = mMk30Comm.* **SendEmKey**(*strKey$*)

引数　strKey$　MARK30 へ送信するキー文字列
“OFF”
“HALT”
“ACK”

戻り値　Integer 型
−1　送信成功
0　送信失敗

解説
- 戻り値が 0 の場合、引数に不正な文字列が使用された場合等、プログラム実行指示前に発生したエラーに対して返される。
  この場合、**GetLastError** メソッドを実行することにより、エラー番号が取得できる
- 送信したキーにより **NcError** イベントが発生する可能性がある

SendFile

機能　MARK30 へファイル転送を行う

書式　*intError = mMk30Comm.* **SendFile**(*strFileName*)

引数　strFileName$　送信するファイル名（拡張子を含む）文字列

戻り値　Integer 型
−1　送信成功
0　送信失敗

解説
- このメソッドを実行する前に、**UserName**、**Password** プロパティを設定する必要がある
- あらかじめクライアント PC で”PCSHARE”と共有がセットされたフォルダにあるファイルのみ転送が可能
- MARK30 に strFileName で指定したファイルがすでに存在する場合、MARK30 のファイルは上書きされる。ただし編集画面で開いているファイルはﾛｯｸされているため上書きできないため、エラーが発生する。
- このメソッドが完了した後、**SendFileComplete** イベントが発生する
- 戻り値は、引数に不正な文字列が使用された場合や送信ファイルが存在しなかった場合等、ファイル転送前に発生したエラーに対して返される
  この場合、**GetLastError** メソッドを実行することにより、エラー番号が取得できる
- ファイル転送時に発生したエラーは **SendFileComplete** イベントにおいて引数でエラー番号が渡される

GetVoltage

機能　MARK30 にある電圧値を取得する

書式　　*nVoltage = mMk30Comm.* **GetVoltage**()

引数　　無し

戻り値　　Integer 型　　取得した電圧値

解説　　・　電圧値(COM3(display_data3[9]))を取得する
戻り値：1 は 1V で表示

GetElectricity

機能　　MARK30 にある電流値を取得する

書式　　*nElectricity = mMk30Comm.* **Electricity**()

引数　　無し

戻り値　　Integer 型　　取得した電流値

解説　　・　電流値(COM3(display_data3[10]))を取得する
戻り値：1 は、0.1A で表示

GetFSpeed

機能　　MARK30 にある F 速度値を取得する

書式　　*nFSpeed = mMk30Comm.* **GetFSpeed**()

引数　　無し

戻り値　　Integer 型　　取得した F 速度値

解説　　・　F 速度値(COM3(display_data[0]))を取得する
戻り値について、メソッド GetInch と GetDigit で mm/inch と digit の値を取得して、1 は次の表のように転換して表示：

Digit	mm	Inch
0	0.001 mm/min	0.0001 inch/min
1	0.0001 mm/min	0.00001 inch/min
2	0.00001 mm/min	0.000001 inch/min
3	0.000001 mm/min	–

GetFrequency

機能　　MARK30 にある頻度値を取得する

書式　　*nFrequency = mMk30Comm.***GetFrequency**()

引数　　無し

戻り値　　Integer 型　　取得した頻度値

解説　　・　頻度値(COM3(display_data3[12]))を取得する
戻り値：1 は、1 cts/ms で表示

GetMaxFrequency

機能　　頻度サーボ頻度数最大値を取得する

書式　　*nMaxFrequency= mMk30Comm.* **GetMaxFrequency** ()

引数　　無し

戻り値　　Integer 型　　取得した頻度サーボ頻度数最大値

解説　　・　頻度サーボ頻度数最大値(COM3(display_data3[13]))を取得する
戻り値：1 は、1 cts/ms で表示

GetSigOnMaxFrequency

機能　　頻度サーボ頻度数最大値時の SIG 値を取得する

書式　　*nSigOnMaxFrequency = mMk30Comm.* **GetSigOnMaxFrequency** ()

引数　　無し

戻り値　　Integer 型　　取得した頻度サーボ頻度数最大値時の SIG 値

解説　　・　頻度サーボ頻度数最大値時の SIG 値(COM3(display_data3[14]))を取得する
戻り値：1 は、1 cts/ms で表示

GetResistivity1

機能　　比抵抗 1 を取得する

書式　　*nResistivity1 = mMk30Comm.* **GetResistivity1** ()

引数　　無し

戻り値　　Integer 型　　比抵抗 1

解説　　・　比抵抗 1 (COM3(display_data3[4]))を取得する
戻り値：1 は、1Ω・cm で表示

GetResistivity2

機能　　比抵抗 2 を取得する

書式　　*nResistivity2 = mMk30Comm.* **GetResistivity2** ()

引数　　無し

戻り値　　Integer 型　　比抵抗 2

解説　　・　比抵抗 2 (COM3(display_data3[5]))を取得する
戻り値：1 は、1Ω・cm で表示

GetInch

機能　　MARK30 にある inch/mm 状態を取得する

書式　　*nInch= mMk30Comm.***GetInch**()

引数　　無し

戻り値　　Integer 型　　取得した inch 値

解説　　・　inch/mm の状態(COM3(init_inch))を取得する
戻り値：0：mm、1:inch

GetDigit

機能　　MARK30 にある digit 値を取得する

書式　　*nFrequency = mMk30Comm.***GetDigit**()

引数　　無し

戻り値　　Integer 型　　取得した digit 値

解説　　・　digit 値(COM3(digit))を取得する
戻り値：0,1,2,3

GetCoordSys

機能　　カレント座標系を取得する。

書式　　*long GetCoordSys()*

引数　　無し

戻り値	long	カレント座標系

解説　　座標系は「nCoordSys と座標系の関係テーブル」を参照する。

GetCoordOrg

機能　　座標系 nCoordSys の原点を取得。

書式　　*long GetCoordOrg(int nCoordSys, double *pdblCoord)*

引数	入力 int	座標系： nCoordSys は「 nCoordSys と座標系の関係テーブル」を参照する。
	出力 double[8]	座標原点：配列の Index　0:X, 1:Y, 2:Z, 3:U, 4:V, 5:W, 6:UU, 7:VV

戻り値	long 型	
	−1	成功
	0	失敗

解説　　失敗した場合(戻り値 0)、**GetLastError** メソッドを実行することにより、エラー番号が取得できる。

GetMachCoord

機能　　機械座標値を取得する。

書式　　*long GetMachCoord(double *pdblCoord)*

引数	出力 double[8]	配列 Index　0:X, 1:Y, 2:Z, 3:U, 4:V, 5:W, 6:UU, 7:VV

戻り値	long 型	
	−1	成功
	0	失敗

解説　　失敗した場合(戻り値 0)、**GetLastError** メソッドを実行することにより、エラー番号が取得できる。

GetCoord

機能　　指定された座標系の座標値を取得する。

書式　　*long GetCoord(long CoordSys,　double *pdblCoord)*

引数　　入力　long CoordSys　　座標系

　　出力　double[8]　　対応座標値:配列 Index　0:X, 1:Y, 2:Z, 3:U, 4:V, 5:W, 6:UU, 7:VV

戻り値　　long 型

−1	成功
0	失敗

解説　　失敗した場合(戻り値 0)、**GetLastError** メソッドを実行することにより、エラー番号が取得できる。

GetCondition

機能　　加工条件を取得して文字列として返す

書式　　BOOLGetCondition(BSTR FAR* strCondition)

引数　　strCondition　　加工条件

戻り値

TRUE	成功
FALSE	失敗

解説　　異なる機種は条件データベースの定義が異なります:
データ定義は以下の通りです:

EDW:
{ON OFF IP HRP MAO SV V SF C PIK CTRL WK WT WS WP}
EDW 33WS
{ ON OFF IP HRP MAO SV V SF C PIK CTRL WK WT WS WP PC SK　}
EDM K3BL:
{ PL ON OFF IP SV S RS MP JS LNS STEP V HP PP C ALV OC LF JM LS LNM}
EDM:
{ PL ON OFF IP SV S UP DN JS LNS STEP V HP PP C ALV OC LF JM LS LNM}

GetMachineKind

機能　　機械タイプ情報を取得する

書式　　BOOL GetMachineKind(long FAR* mainKind,long FAR* subKind)

引数

	mainKind	0：EDW；1：EDM
	subKind	mainKind が 1 の時： 1：LQ33WS；その他：普通 mainKind が 0 の時： 1：K3BL, その他：K3BL 以外

戻り値

TRUE	成功
FALSE	失敗

解説　　機械タイプは 2 つの引数で構成されており、第 2 引数の意味は第 1 引数によって決まる,
mainKind が 1 で、EDW の場合、subKind が 1 の場合は、LQ33WS になる
mainKind が 0 で、EDM の場合、subKind が 1 の場合は、K3BL になる

GetConditionType

機能　　加工条件タイプを取得する（33WS / 33WS 以外）

書式　　*Short　GetConditionType()*

引数　　無

戻り値　　*Short　型*
0：旧加工条件（33WS 以外）
1：新加工条件（33WS）
2: SPW 加工条件

解説　　現在の条件検索のタイプを取得します。
戻り値　0：33　WS 以外　1：33WS　2:SPW

CreateNcCond

機能　　加工条件検索

書式　　short　CreateNcCond(　short nParamType,
　　short DatabaseType,
　　double* cnd,
　　double* mchcon,
　　double* mcb,
　　double* ncProc,
　　long* plnNcProcNum,
　　long* fin,
　　long* plnMTRL)

引数　　nParamType：　　入力と出力の構造の種類を決定する
DatabaseType：　　データベースの種類
Cnd：　　条件検索データ

以下、出力パラメータ：
Mchcon：　　加工条件コンテンツ
mcb：　　加工条件情報
ncProc：　　AIC コード/ SFCC コード/ TT、TB コード/ CSP コード
plnNcProcNum：　　AIC および SFCC、TT、TB、および CSP 出力コードの数
fin；　　//未使用
plnMTRL：　　出力の MTRL 値

戻り値　　short
1：データベースの検索成功
0：条件検索データの送受信タイムアウト（検索に失敗）
-1：条件検索の失敗
-2：データベースエラー
-3：HI、HtNC、LQ 条件検索エンジンの使用中
-4：条件検索で必要なファイルが見つからない
-5：条件検索で必要なファイルのオープンに失敗
-6：入力データが正しくないため、データベースに書き込むことができない
-7：Msg_cond.dat ファイルが見つからない

解説　　添付の「Mk30Comm 条件検索の手順.doc」を参照してください。

GetHiVersion

機能　　電源操作部のバージョンを取得する

書式　　ret = mMk30Comm. GetHiVersion (BSTR* pstrHiVersion)

引数　　pstrHiVersion　　電源操作部のバージョン

戻り値　　Integer 型
-1　　所得成功
0　　所得失敗

解説　　Windows システム API 関数の SysFreeString を使用して pstrHiVersion のメモリを解放する。

GetRunningFileName

機能　現在実行中プログラムのファイル名、パスを取得します。サブプログラムが実行中の場合、サブプログラム名を取得する。

書式　ret = mMk30Comm. GetRunningFileName (BSTR* pstrRunningFileName,long *pFilePathType)

引数

*pstrRunningFileName【out】	実行中のファイル名を取得します。サブプログラム実行中の場合はサブプログラム名を取得する。
*pFilePathType[out]	ファイルパスの種類。 0：メモリ 1：ディスク

戻り値　Integer 型

-1	所得成功
0	所得失敗

解説　加工していない時、戻り値は 0（失敗）
加工以外の NC ファイル実行時は，戻り値は 0（失敗）
Windows システム API 関数の SysFreeString を使用して pstrRunningFileName のメモリを解放する。

GetRunningMainNcFileFirstLine

機能　　実行中のメイン NC ファイルの最初の行内容を取得する。

格式　　ret = mMk30Comm. GetRunningMainNcFileFirstLine (BSTR* pstrFirstLine)

引数　　* pstrFirstLine【out】　　* pstrFirstLine【out】

戻り値　　Integer 型
−1　　所得成功
0　　所得失敗

解説　　加工していない時、戻り値は 0（失敗）
加工以外の NC ファイル実行時、戻り値は 0（失敗）
サブプログラムが実行中でも、メイン NC ファイルの最初の行の内容を読み込む
Windows システム API 関数 SysFreeString を使用して、pstrFirstLine のメモリを解放する

GetNcFileRunEventTime

機能　　機械の前回と今回の NC プログラムの加工時間の情報を取得する。

書式　　intError = mMk30Comm.GetNcFileRunEventTime(BSTR* pstrPrevEventTime,
BSTR*pstrCurrentEventTime)

引数　　pstrPrevEventTime【out】　　前回の NC プログラムの加工時間情報の文字列
pstrCurrentEventTime【out】　　今回の NC プログラムの加工時間情報の文字列

戻り値　　long 型
−1　　所得成功
0　　所得失敗

解説
- 電源起動後に NC ファイルが実行されない場合は、pstrPrevEventTime と pstrCurrentEventTime の両方が NULL になる。
- 電源起動後、初回の実行中、または一度だけ NC プログラムが実行された場合、pstrPrevEventTime は NULL で、pstrCurrentEventTime は NULL ではない。
- PstrPrevEventTime と pstrCurrentEventTime は、電源起動後、二度目の実行中、または 2 つ以上の NC プログラムが実行された場合、NULL ではなくなる。
- Windows システム API 関数を呼び出す SysFreeString を使用して、pstrPrevEventTime と pstrCurrentEventTime のメモリを解放する。

- pstrPrevEventTime と pstrCurrentEventTime の形式は次の通り。

  ####-##-## ##:##:##　　→1 行目：このメソッド（GetNcFileRunEventTime）が呼び出された日時。（機械の日付と時刻）

  ####-##-## ##:##:##　　→2 行目：加工開始時間

  @　　→3 行目：加工状態 0：実行中 1：加工終了

  *　　→4 行目：加工開始後の総時間(単位：秒)HALT、ERROR 時間を含む。

  *　　→5 行目：加工中の HALT 時間(単位：秒)

  *　　→6 行目：加工中の ERROR 時間(単位：秒)

```
$                       →7 行目：今回の加工中に発生したイベントの番号

                        イベント数が 800 以下の場合、値は 1 になり、8 行目からすべてのイベントを記録します。
                        イベント数が 800 より大きい場合、値は 1 より大きくなり、8 行目から直近の 800 イベントが記録されます。
&,####-##-## ##:##:##   →8 行目：各イベントの種類と時間。（TYPE、YYYY-MM-DD HH:MM:SS）
&,####-##-## ##:##:##
&,####-##-## ##:##:##
…
&,####-##-## ##:##:##
```

注：

1. pstrPrevEventTime と pstrCurrentEventTime の 2 つの変数を使用すると、1 つ目と 2 つ目の加工間の時間が正確に分かるので、時間統計に役立ちます。
2. メモリリークを防ぐ為に pstrPrevEventTime と pstrCurrentEventTime 2 つの文字列には最新の 800 イベントが保存されています。
3. 文字列の中の改行位置を「¥ x0d ¥ x0a」で示す。
4. ####-##-## ##:##:##：時間形式（例 2011-06-25 16:03:02）
5. ＊ ：時間を示す文字列は可変長。 1、23、343333、…など
6. @ ：加工状態 0：実行中 1：加工終了
7. $ ：加工中に発生したイベントの記録箇所(可変長)。番号は 1 から始まる。
   現在のイベント数が 800 より大きくない場合は、$ = 1 になります。
   現在のイベント数が 800 を超える場合、イベントの合計数 － $ 800 + 1 になる。
8. & ：イベントのタイプ。 以下の 4 種類：
   0: Ready 移行時
   1: 加工状態に移行時
   2: HALT 発生時
   3: Error 発生時

FtpSendFile

機能　　FTP を使用して、PC から機械のメモリディレクトリにファイルを転送する

書式　　intError = mMk30Comm. FtpSendFile(strFileName)

引数

strFileName$	転送するファイル名（拡張子を含む）

戻り値　　Integer 型

−1	送信成功
0	送信失敗

解説

- このメソッドを使用する前に、FtpUserName、FtpPassword プロパティを設定する必要がある
- 外部 PC には　FTP 共有ディレクトリ"PCSHARE"が存在する。
- strFileName で指定されたファイルが機械のメモリに存在する場合、元のファイルに上書きする。 ただし、ファイル名が編集画面のファイル名と一致するとエラーが発生。
- メソッドが終了すると、SendFileComplete イベントが実行される。
- 不正な文字やその他の文字列を含む文字列が引数に含まれると、戻り値はエラーになる。この場合、GetLastError メソッドを使用してエラー番号を取得する。
- ファイル転送中にエラーが発生した場合、エラー番号は SendFileComplete のパラメータで指定されます。

GetCurrentLogonUserInfo

機能　　機械側のログインユーザーに関する情報を取得する

書式　　ret = m_Mk30Comm. GetCurrentLogonUserInfo (BSTR* pstrUserLoginName , BSTR* pstrUserName, BSTR* pstrUserNo, long* pUserLevel)

引数

pstrUserLoginName	ログイン名
pstrUserName	ユーザ名
pstrUserNo	番号
pUserLevel	ユーザ制限レベル

戻り値　Long 型

−1	所得成功
0	所得失敗

解説
- Windows システム API 関数 SysFreeString を使用して、pstrUserLoginName, pstrUserName, pstrUserNo のメモリを解放する。
- pstrUserLoginName の戻り値が NULL の時は、機械側にユーザー名・ログイン名がないことを示す。
- この方法は[富士康 CAD/CAM 中心]ユーザー権限管理ソフトウェア専用です。

FarLogout

機能　　遠隔操作(PC)による機械のログアウト

書式　　ret = m_Mk30Comm. FarLogout (long *pError)

引数

pError	遠隔操作する時にエラーが発生すると、その種類を返す。

戻り値　Long 型

−1	成功
0	失敗

解説
- pError の値

  0: 正常
  1: PC 側と機械側は接続されていない
  2: 機械は管理者権限の操作をサポートしていない
  3: 遠隔操作する時に[設定] − [フラグ] − [SEIKAnet]の ″ネットワークからの操作″画面で、 ″機械操作の実行″が許可されている場合はログアウトできない
  4: 機械の権限レベルが最高の時、緊急処理時に外部 PC の権限変更ができる
  外部 PC が機械にログインすると、その時ログインしている外部 PC はログアウトする。

  遠隔操作でログイン中にネットワークが切断した場合、機械からログアウトする為には、機械の最高レベルのユーザーで登録し、ログアウトボタンから行う。

  新しいユーザー名とパスワードを入力すると、現在のユーザーはログアウトし、その後、新しいユーザーでログインされる。
- この方法は【富士康 CAD/CAM 中心】ユーザー権限管理ソフトウェア専用です。

FarLogon

機能　遠隔操作(PC)で機械にログインする。

書式　ret = m_Mk30Comm. FarLogon (LPCTSTR strUserNo ， LPCTSTR strPasswrod, long *pError)

引数

strUserNo	番号
strPasswrod	パスワード
pError	遠隔ログインが失敗したときのエラーの種類を返します。

返り値　Long 型

−1	成功
0	失敗

解説

・ *pError の値
0: 正常
1: PC 側と機械は接続が確立されない
2: 機械は管理者権限の操作をサポートしていない
3: ［設定］－［フラグ］－［SEIKAnet］の ″ネットワークからの操作″画面で、 ″機械操作の実行″が許可されている場合ログアウトできない。
4: 番号またはパスワードが、機械の登録ファイル UserPermission.dat と一致しない
5：ユーザー管理者権限のファイルが見つからない

外部 PC は機械にログインし、別の PC は機械に再度ログインできる。
遠隔操作でログイン中にネットワークが切断した場合、機械からログアウトする時は、機械の最高レベルのユーザーで登録する。

・ この方法は【富士康 CAD/CAM 中心】ユーザー権限管理ソフトウェア専用です。

GetUserLogonInfFile

機能　ユーザー管理者権限の、ログインとログアウトのログを取得する。

書式　intError = m_Mk30Comm. GetUserLogonInfFile (BSTR lpLocalFileFullPath)

引数

lpLocalFileFullPath	外部 PC のファイルのフルパス

戻り値　long 型

−1	成功
0	失敗

解説

・ このメソッドは、API 関数 FtpGetFile を使用して、FTP 共有ディレクトリの DAT フォルダのファイルを取得します。
・ この方法は【富士康 CAD/CAM 中心】ユーザー権利管理ソフトウェア専用です。

GetMachineLock

機能　　Machine Lock の状態を取得する。

書式　　long　GetMachineLock()

引数　　無

戻り値　　long 型　　Machine Lock の状態を取得する。

解説　　・　MachineLock の値を取得する (COM3(display_data3[16]))
・　-1 戻る場合、MachineLock 状態取得失敗。

GetOffsetVar

機能　　指定番号の Offset データを取得する

書式　　long nRet = GetOffsetVar(long nOffsetNumber, double * dblOffsetValue)

引数　　入力 nOffsetNumber　　オフセットの番号
マクロ変数の番号 = nOffsetNumber + 10000
出力 dblOffsetValue　　取得したオフセットデータのポインタ (double *)

戻り値　　long 型
-1　　成功
0　　失敗

解説　　nOffsetNumber の範囲: 0 ～ 9999
nOffsetNumber + 10000 → Macro 変数の番号

SetMacroVar

機能　　指定番号の Offset データをセットする

書式　　long nRet = SetMacroVar(long nMacroNumber, double dblMacroValue)

引数　　入力 nMacroNumber　　マクロ変数の番号

出力 dblMacroValue　　セットしたいオフセットデータ

戻り値　　long 型
-1　　成功
0　　失敗

解説　　nMacroNumber の範囲: 0 ～ 199999

GetMacroVarRange

機能　　多数個マクロ変数から、データを取得する（最大連続の 100 個まで）
（VC で使用できるインタフェース）

書式　　long nRet = GetMacroVarRange(long nStartNo, long nEndNo, double * dblMacro)

引数　　入力 nStartNo　　開始のマクロ番号

入力 nEndNo　　終了のマクロ番号

输出 dblMacro　　取得したマクロデータの配列

戻り値　　long 型
>0　　取得したマクロデータの Count 数
0　　失敗

解説　　a) 0 <= nStartNo < 200000
b) 0 <= nEnd < 200000
c) nEndNo < nStartNo + 100
d) nRet : 1~100, 取得したマクロ変数の個数
0, 失敗
失敗の時、GetLastError で error 番号取得できます。
LastError = 17, nStartNo, or nEndNo が <0 or >=200000
LastError = 5, nStartNo と nEndNo の間隔は、正しくありません。
正しい範囲は、 0 < nEndNo − nStartNo < 99

GetMacroVarRangeSafeArray

機能　　多数個マクロ変数から、データを取得する（最大連続の 100 個まで）
（VB で使用できるインタフェース）

書式　　long nRet = GetMacroVarRange(long nStartNo, long nEndNo, VARIANT* varMacro)

引数　　入力 nStartNo　　開始のマクロ番号

入力 nEndNo　　終了のマクロ番号

输出 varMacro　　取得したマクロデータの配列

戻り値　　long 型
>0　　取得したマクロデータの Count 数
0　　失敗

解説　　a) 0 <= nStartNo < 200000
b) 0 <= nEnd < 200000
c) nEndNo < nStartNo + 100
d) nRet : 1~100, 取得したマクロ変数の個数
0, 失敗
失敗時、GetLastError で error 番号取得できます。
LastError = 17, nStartNo, or nEndNo が <0 or >=200000
LastError = 5, nStartNo と nEndNo の間隔は、正しくありません。
正しい範囲は、 0 < nEndNo − nStartNo < 99

SetMacroVarRange

機能	多数個マクロ変数に、データをセットする（最大連続の 100 個まで） （VC で使用できるインタフェース）	
書式	long nRet = SetMacroVarRange(long nStartNo, long nEndNo, double * dblMacro)	
引数	入力 nStartNo	開始のマクロ番号
	入力 nEndNo	終了マクロ番号
	输出 dblMacro	セットしたいマクロデータの配列
戻り値	long 型	
	>0	セットしたマクロデータの Count 数
	0	失敗
解説	a) 0 <= nStartNo < 200000 b) 0 <= nEnd < 200000 c) nEndNo < nStartNo + 100	

SetMacroVarRangeSafeArray

機能	多数個マクロ変数に、データをセットする（最大連続の 100 個まで） （VC で使用できるインタフェース）	
書式	long nRet = SetMacroVarRange(long nStartNo, long nEndNo, VARIANT* varMacro)	
引数	入力 nStartNo	開始マクロ番号
	入力 nEndNo	終了マクロ番号
	输出 varMacro	セットしたいマクロデータの配列
戻り値	long 型	
	>0	セットしたマクロデータの Count 数
	0	失敗
解説	a) 0 <= nStartNo < 200000 b) 0 <= nEnd < 200000 c) nEndNo < nStartNo + 100	

GetUserLogonInfFile

機能	ユーザーログイン、ログアウト履歴の取得 LOG ファイル	
書式	intError = m_Mk30Comm. GetUserLogonInfFile (BSTR lpLocalFileFullPath)	
引数	lpLocalFileFullPath	PC ローカルファイルのフルパス
戻り値	long 型	
	−1	成功
	0	失敗

解説

- GetUserLogonInfFile の機能は、API 関数 FtpGetFile を内部的に使用して、電源側の FTP 共有ディレクトリの DAT で指定されたファイルを取得します。
- この方法は、Foxconn CAD / CAM センターのユーザー権利管理ソフトウェア専用です。

GetRamFileList

機能　RamList ファイル名のリストを取得する

書式　*long GetRamFileList(IVARIANT*pvarList)*

引数　出力 :　ocx から安全な配列を保持するために使用されるラッパーオブジェクト
VARIANT*pvarList

戻り値　long 型
リスト内のファイル数を返す
ファイルが無い時は 0 を返す

解説　CIMForce の要件に従って増加
使用事例　C#:

```
Object obj = new Object();
Object obj2 = new System.Runtime.InteropServices.VariantWrapper(obj);
if (0 != this.axMk30Comm1.GetRamFileList(ref obj2))
 {
        Array aaa = (Array)obj2;
        this.listBox1.Items.Clear();

        foreach (String x in aaa)
        {
            this.listBox1.Items.Add(x);
        }
}
```

SendFileFromFolder

機能　pcshare のサブディレクトリにあるファイルをマシンの Ram ディレクトリに送り、マシンの Ram リストを追加する。

書式　*long SendFileFromFolder(LPCTSTR strFolder, LPSTR strFileName)*

引数　入力 :
LPCTSTR strFolder
LPCTSTR strFileName

戻り値　-1 : 成功
0 : 失敗

解説　CIMForce の要件に従って増加
使用事例　C#:

```
this.axMk30Comm1.SendFileFromFolder(txtFolder.Text, txtFileName.Text);
```

SetActiveNCProgram

機能　　マシンの Ram リストのプログラムを実行ファイルとして選択する。

書式　　*long SetActiveNCProgram(LPSTR strFileName)*

引数　　入力 :
LPCTSTR strFileName

戻り値　　-1:成功
0:失敗

解説　　CIMForce の要件に従って増加
使用事例　C#:

```
object obj = this.listBox1.SelectedItem;
if (null != obj)
        this.axMk30Comm1.SetActiveNCProgram(obj.ToString());
```

GetRemainLength

機能　　加工残量を取得する。

書式　　*double GetRemainLength( )*

引数　　なし

戻り値　　加工残量, double 型

解説　　CIMForce の要件に従って増加
使用事例　C#:

```
this.txtRemainLength.Text =    this.axMk30Comm1.GetRemainLength().ToString();
```

GetBlockRemainLength

機能　　1ブロックのブロック残量を取得する。

書式　　*double GetBlockRemainLength( )*

引数　　なし

戻り値　　1 ブロック残量, double 型

解説　　CIMForce の要件に従って増加
使用事例　C#:

```
this.txtBlockRemainLength.Text =     this.axMk30Comm1.GetBlockRemainLength().ToString();
```

GetPowerOnSeconds

機能　　機械の累積ソースオン時間を取得する。

書式　　*ULING GetPowerOnSeconds()*

引数　　なし

戻り値　　ソースオン時間 [秒]

解説　　CIMForce の要件に従って増加
メンテ画面で使用
使用事例　C#:

```
double t1 =    this.axMk30Comm1.GetPowerOnSeconds();
```

GetWorkingSeconds

機能　　機械の累積加工時間を取得する。

書式　　*ULING GetWorkingSeconds()*

引数　　なし

戻り値　　加工時間［秒］

解説　　CIMForce の要件に従って増加
メンテ画面で使用

使用事例　C#:

```
double t2 =   this.axMk30Comm1.GetWorkingSeconds();
```

GetActiveNCProgram

機能　　現在の NC プログラムを取得する

書式　　*long GetActiveNCProgram(BSTR *strProgramName)*

引数　　なし

戻り値　　-1：成功
0：失敗

解説　　CIMForce の要件に従って増加
現在のNCプログラムとは、機械側のENTキーを押して実行するプログラムのこと。

使用事例:　C#:

```
String man = "";
this.axMk30Comm1.GetActiveNCProgram(ref man);
```

DeleteAllFiles

機能　　NewFile を除く、Ram ディレクトリ内のすべてのファイルを削除する。

書式　　*long DeleteAllFiles()*

引数　　なし

戻り値　　-1：成功
0：失敗

解説　　CIMForceの要件に従って増加

使用事例:　C#：

```
this.axMk30Comm1.DeleteAllFiles();
```

GetCutInfo

機能　　加工情報を取得する。

書式　　*long GetCutInfo(BSTR *strDirMain , BSTR *strNameMain , BSTR*strDirCurrent, BSTR*strNameCurrent,,VARIANT*offset)*

引数　　出力パラメータ ：

BSTR*strDirMain	加工中のメインファイルファイルディレクトリ
BSTR*strNameMain	加工中のメインファイル名
BSTR*strDirCurrent	現在の加工プログラムのファイルディレクトリ
BSTR*strNameCurrent	現在の加工プログラムファイル名
VARIANT*offset	現在の加工のオフセット

戻り値　　-1：成功
0：失敗

解説　　CIMForceの要件に従って増加

使用事例:　C#：

```
string strDirMain = "";
string strDirCur = "";
string strNameMain = "";
string strNameCur = "";
Object obj = new Object();
Object obj2 = new System.Runtime.InteropServices.VariantWrapper(obj);

this.axMk30Comm1.GetCutInfo(ref strDirMain, ref strNameMain, ref strDirCur,
                            ref    strNameCur, ref obj2);
```

SetLnProDataProcessing

機能　　LnPro 用の加工パラメータのデータをセットする。

書式　　***long SetLnProDataProcessing(LPCTSTR　strDataParameter);***

引数　　入力パラメータ :

LPCTSTR ***strDataParameter***	参照ドキュメント“Design-CIMForce_LNPro-Interface.xlsx”

戻り値　　-1 : 成功
0 : 失敗

解説　　この関数は、CimForceによって呼び出され、パラメータを渡し、LnProのさまざまな関数を実行する。

使用事例:　C# :

string ***strDataParameter*** = xml書式文字列;
this.axMk30Comm1. ***SetLnProDataProcessing*** (***strDataParameter***);

LoadHDiskFile

機能　　機械側の Disk のファイルを、RAM にロードして、加工画面に切り替わる

書式　　*intError = mMk30Comm.* **LoadHDiskFile** (*strFileName*)

引数

strFileName$	ロードするファイル名（拡張子を含む）文字列

戻り値　　Integer 型

-1	送信成功
0	送信失敗

解説
- ・このメソッドを実行する前に、設定・フラグ・SEIKAnet の「ネットワークからの操作」画面で、「メモリ」領域の書き込み・削除」を許可する必要がある。
- ・機械の RAM に strFileName で指定したファイルがすでに存在する場合、RAM のファイルは上書きされる。ただし、編集画面で開いているファイルはロックされるため上書きできず、エラーが発生する。
- ・このメソッドが完了した後、**LoadHDiskFileComplete** イベントが発生する。
- ・戻り値は、引数に不正な文字列が使用された場合や送信ファイルが存在しなかった場合等、ファイル転送前に発生したエラーに対して返される。
- ・ファイル転送時に発生したエラーは **LoadHDiskFileComplete** イベントにおいて引数でエラー番号が渡される。

GetEleOfsSettings

機能　　電極補正関連の設定を取得する。

書式　　*long nRet = mMk30Comm.GetEleOfsSettings (VARIANT* nSettings)*

引数　　出力

nSettings	電極補正関連の設定

戻り値　long 型

−1	送信成功
0	送信失敗

解説　　・　SAFEARRAY 型の引数 varSettings から、電極補正関連の設定を取得する。

nSettings[0]: 電極補正画面で表示された軸 (0:XYZU, 1:XYZ, 2:XYZW, 3:XYZ, その他:XYZU)
nSettings[1]: フラグ・[29]回転軸入力表示種類
nSettings[2]: 機械・[4]工具収納可能本数
nSettings[3]: 操作・[38]電極補正基準登録パターン
nSettings[4]: 操作・[39]電極補正有効パターン数
nSettings[5]: 動作・[2]ディジット
nSettings[6]: 動作・[2]初期状態 インチ
nSettings[7]: 電極補正で、表示している補正パターン。電極補正画面表示されない場合、0にセット。

GetEleOfs

機能　　機械側の電極補正画面で、電極の補正量と基準登録データを読み取って、PC 側に保存。

書式　　*long nRet = GetEleOfs (long nPattern)*

引数　　入力

nPattern	指定したパタン番号

戻り値　long 型

−1	送信成功
0	送信失敗

解説　　・　PC 側の共有フォルダ”PCSHARE”は、書き込み権限をセットする必要がある。
・　Ocx の UserName、Password 属性をセットする必要がある。
・　HI 側で、機械の共有フォルダ”NCPROHD”に、ファイル tool_ocx_output_standard#.txt, tool_ocx_output#.txt を作成する； Ocx が、機械側の共有フォルダ”NCPROHD”から、ファイル tool_ocx_output_standard#.txt, tool_ocx_output#.txt を、PC の”PCSHARE”共有フォルダにコピーする。
#は、パターン番号、値の範囲は、1～6。

SetEleOfs

機能　　PC 側の電極の補正量と基準登録データを、機械へ書き込む。

書式　　*long nRet = SetEleOfs (long nPattern)*

引数　　入力

nPattern	指定したパタン番号

戻り値　　long 型

−1	送信成功
0	送信失敗

解説

- PC 側の共有フォルダ”PCSHARE”は、書き込み権限をセットする必要がある。
- Ocx の UserName、Password 属性をセットする必要がある。
- HI 側で、PC の”PCSHARE”共有フォルダから、tool_ocx_output_standard#.txt , tool_ocx_output#.txt を機械側にコピーする;作成する; tool_ocx_output_standard#.txt, tool_ocx_output#.txt を解析して、指定したパタンの電極補正データを更新して、電極補正画面にもセットする。
  #は、パタン番号、値の範囲は、1～6。

電極補正ファイル tool_ocx_output_standard#.txt, tool_ocx_output#.txt の説明:

1. #は、パタン番号、値の範囲は、1～6。

   操作・[38]電極補正有効パターン数<6 の場合、＃の範囲は、1～操作・[38]電極補正有効パターン数。

2. tool_ocx_output_standard#.txt は、基準登録のデータを記入してください

   a) 基準登録データは、1 行のみからなるデータファイルです。

   b) データフォーマット

   ツール No., X 機械座標, Y 機械座標, Z 機械座標[, U/W 機械座標]

   XYZ 軸の単位は、mm です。

   U、W 軸は、角度で、単位は、度です。

   今後、U、W 軸は長さの場合、mm 単位で、HI で対応する必要があります。

   注: U/W 機械座標は、無い場合もあります。

   メソッド GetEleOfsSettings によって、nSettings[0]から、U、W 表示するかを判断します。

   例 1:XYZU 軸データあった場合

   2, +63.0002, +58.0010,+39.0003,+56.0005

   例 2:XYZ 軸データのみあった場合

   2, +63.0002, +58.0010,+39.0003

3. tool_ocx_output#.txt は、電極補正のデータ

   a) 電極補正のデータは、最大で 999 個からなるツールデータを記入してください。

   b) データフォーマット

   ツール No., X 機械座標, Y 機械座標, Z 機械座標[, U/W 機械座標]

   XYZ 軸の単位は、mm です。

   U、W 軸は、角度で、単位は、度です。

   今後、U、W 軸は長さの場合、mm 単位で、HI で対応する必要があります。

   注: U/W 機械座標は、無い場合もあります。

   メソッド GetEleOfsSettings によって、nSettings[0]から、U、W 表示するかを判断します。

   例 1: XYZW 軸データあった場合

   1,+0.0001,−0.0001,+0.0002,0.0005

   2,+0.0002,−0.0010,+0.0003,0.0005

3,+0.0002,−0.0010,+0.0003,0.0005

|

○, +0.0002,−0.0010,+0.0003,0.0005

例 2:XYZU 軸データのみあった場合

1,+0.0001,−0.0001,+0.0002

2,+0.0002,−0.0010,+0.0003

3,+0.0002,−0.0010,+0.0003

|

○, +0.0002,−0.0010,+0.0003

DeleteFileComplete

機能	MARK30 にあるファイルを削除後、削除完了時に発生		
書式	*mMk30Comm_* **DeleteFileComplete**(*intError,Cstring strError*)		
引数	intError	エラー番号	ファイル削除時にエラーが発生した場合はそのエラー番号、正常な場合は 19 が渡される
引数	Cstring strError	エラーメッセージ	ファイル削除時にエラーが発生した場合はそのエラーメッセージ、正常な場合は“Mark30 のラム内の指定されたプログラムを削除しました”が渡される
解説	・ **DeleteFile** メソッド実行後、このイベントが発生する		

DenyRemoteAccess

機能	MARK30 が遠隔実行を認めない場合に発生
書式	*mMk30Comm_* **DenyRemoteAccess**( )
引数	なし
解説	・各メソッドを実行後、MARK30 が遠隔実行を認めない場合に発生する

ExecuteComplete

機能	MARK30 で NC プログラム実行終了時に発生		
書式	*mMk30Comm_* **ExecuteComplete**(*intError,Cstring strError*)		
引数	intError	エラー番号	プログラム実行指示時にエラーが発生した場合はそのエラー番号、正常な場合は 166 が渡される
引数	Cstring strError	エラーメッセージ	プログラム実行指示時にエラーが発生した場合はそのエラーメッセージ、正常な場合は“プログラム終了”が渡される
解説	・ **Execute** メソッド実行後、このイベントが発生する		

NetworkError

機能　　ネットワークエラー時に発生

書式　　*mMk30Comm_* **NetworkError**(*intError*)

引数　　intError　　ネットワークエラー番号 0:Connct 時エラーが発生
1:Dsiconnect 時エラーが

解説　　・ **Connect** メソッド実行後、このイベントが発生する

NcError

機能　　MARK30 でエラー・ハルト時に発生

書式　　*mMk30Comm_* **NcError**(*strError*)

引数　　strError$　　エラー・ハルト文字列

解説　　・E00166 の場合、このイベントは発生しない

MachineStatusChanged

機能　　MARK30 の機械状態を変わる時に発生

書式　　*mMk30Comm.**MachineStatusChanged(long oldValue, long newValue)***

引数　　oldValue　　変わる前の状態

引数　　NewValue　　現在の状態

解説　　newValue,と oldValue の値:

- 0:READY
- 1:実行中
- 2:HALT 中
- 3:ACK 待ち
- −1:初期値

SendFileComplete

機能	MARK30 へファイル転送後、転送完了時に発生		
書式	*mMk30Comm_* **SendFileComplete**(*intError,Cstring strError*)		
引数	intError	エラー番号	ファイル転送時にエラーが発生した場合はそのエラー番号、正常な場合は 17 が渡される
引数	Cstring strError	エラーメッセージ	ファイル転送時にエラーが発生した場合はそのエラーメッセージです、正常な場合は“Mark30 のラムに正常に書き込みました”が渡される
解説	・ **SendFile** メソッド実行後、このイベントが発生する。		

EventLnProProcessResult

機能	这是一个事件，当 LnPro 执行完成某个功能后，通知 CimForce 执行结果。	
書式	*mMk30Comm_ ventLnProProcessResult(CString strDataParameter);*	
引数	strDataParameter	通知 CimForce 执行结果。
解説	・这是一个事件，当 LnPro 执行完成某个功能后，通知 CimForce 执行结果。 参照文档“Design-CIMForce_LNPro-Interface.xlsx”	

LoadHDiskFileComplete

機能　メソッド LoadHDiskFile の実行終了時に発生

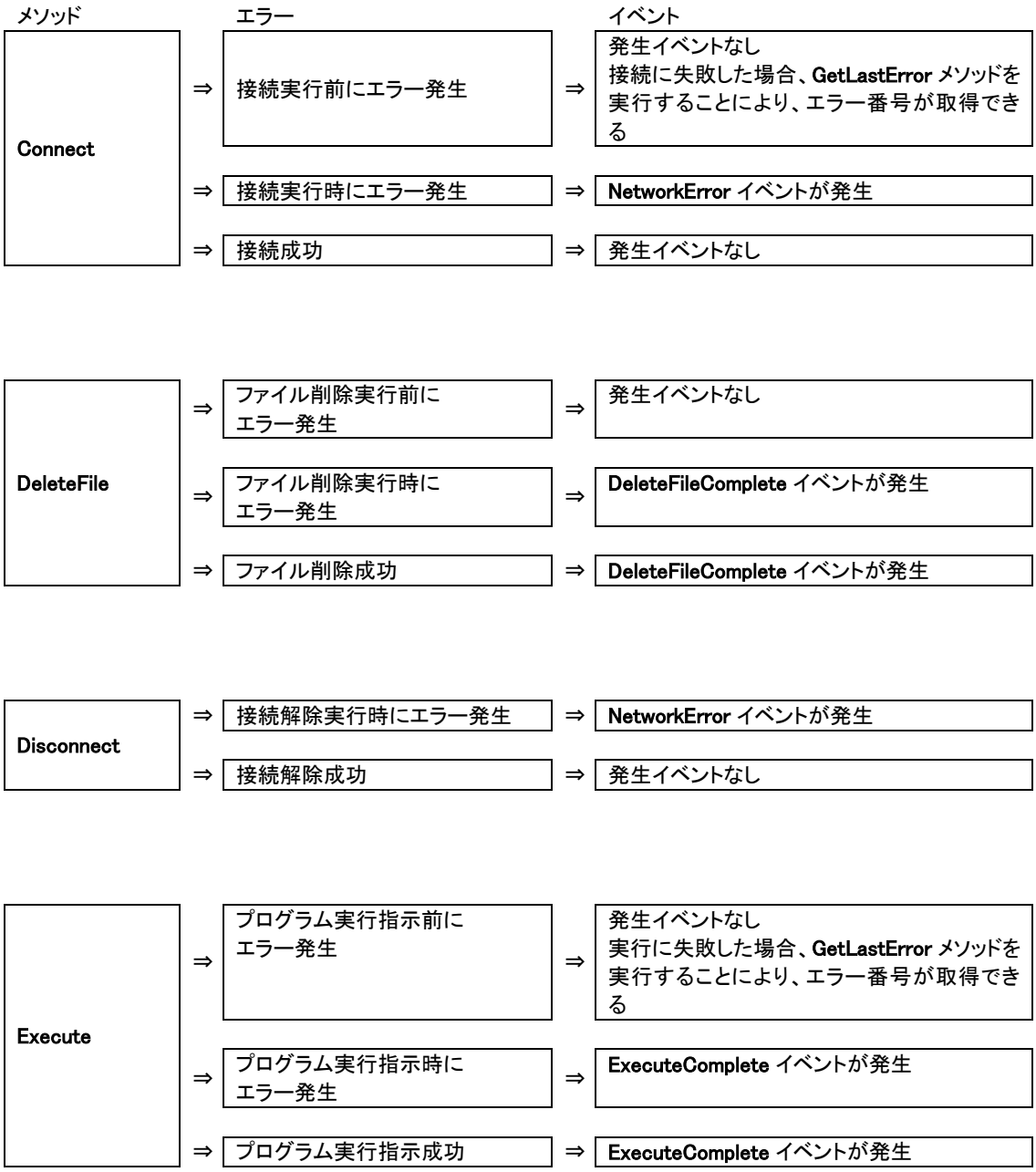
書式　*mMk30Comm_* **LoadHDiskFileComplete** (*intError,Cstring strError*)

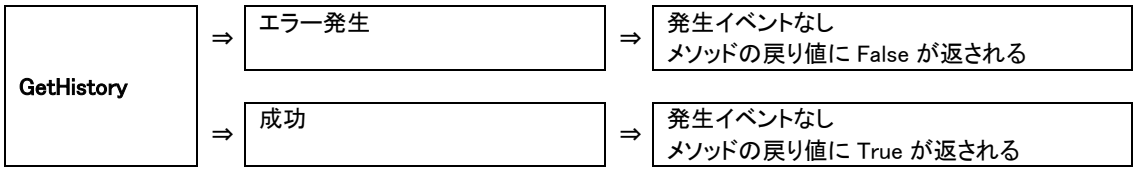
引数	intError	エラー番号	ファイルロード時にエラーが発生した場合はそのエラー番号、正常な場合は 49 が渡される
引数	Cstring strError	エラーメッセージ	未使用

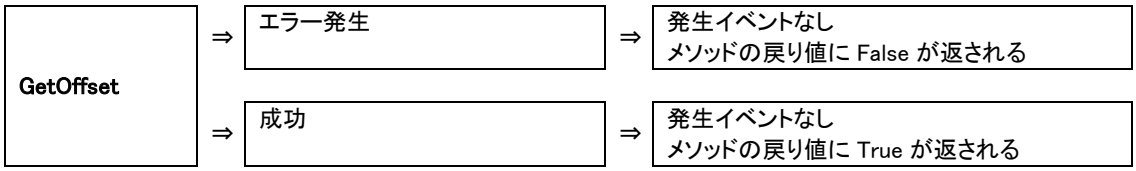
解説　・**LoadHDiskFile** メソッド実行後、このイベントが発生する。
・intError の説明:
49 – 正常にロードされた。
0 – 設定・フラグ・SEIKAnet の「ネットワークからの操作」画面で、「メモリ」領域の書き込み・削除」は、許可されない。
1 – LoadHDiskFile ファイルロード中、2 回目 LoadHDiskFile をコールする場合。
2 – 機械側 Busy 中の状態。
3 – D:ドライブの空き容量が不足。
4 – RAM にロード失敗。(例えば、H_Disk にファイルない)
5 – 編集中のファイルを、ロード禁止。
6 – ファイルを RAM にロードしたが、加工モードに切り替わる失敗。
加工モードに切り替わる失敗の例:
・MDI で、ロード、セーブダイアログ表示されている時;
・編集で、ロード、セーブダイアログ表示されている時;
・メッセージボックスが表示されている時;
・データバックアップ中の時;
・データリストア中の時;
・バージョンアップ中の時;
・UTY/LORAN 編集中の時;
・ …
上記以外の値 – 未知のエラー。

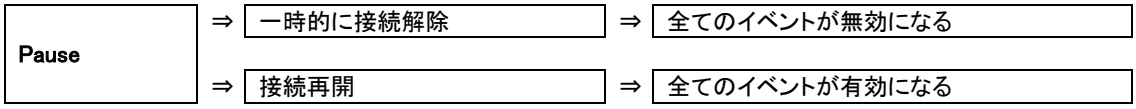
4.5　メソッドとイベントの関係

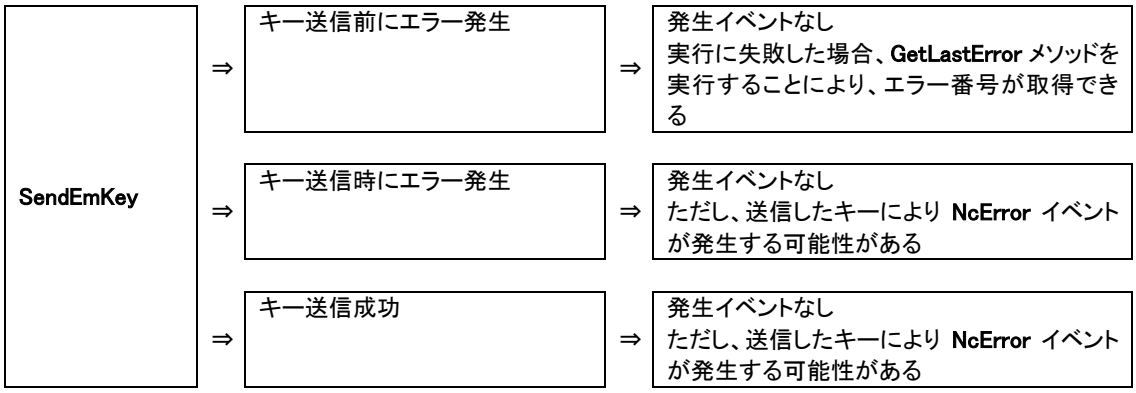
実行したメソッドにより、発生するイベントが異なってくる。その関係を以下に示す。

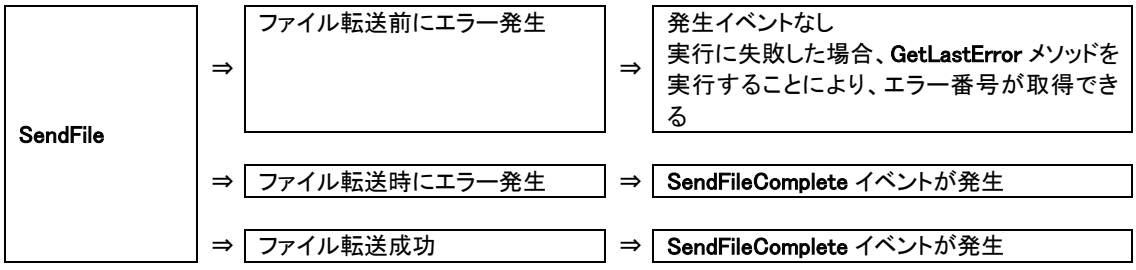



GetHistory
エラー発生
発生イベントなし
メソッドの戻り値に False が返される
成功
発生イベントなし
メソッドの戻り値に True が返される


GetOffset
エラー発生
発生イベントなし
メソッドの戻り値に False が返される
成功
発生イベントなし
メソッドの戻り値に True が返される


Pause
一時的に接続解除
全てのイベントが無効になる
接続再開
全てのイベントが有効になる


SendEmKey
キー送信前にエラー発生
発生イベントなし
実行に失敗した場合、GetLastError メソッドを実行することにより、エラー番号が取得できる
キー送信時にエラー発生
発生イベントなし
ただし、送信したキーにより NcError イベントが発生する可能性がある
キー送信成功
発生イベントなし
ただし、送信したキーにより NcError イベントが発生する可能性がある


SendFile
ファイル転送前にエラー発生
発生イベントなし
実行に失敗した場合、GetLastError メソッドを実行することにより、エラー番号が取得できる
ファイル転送時にエラー発生
SendFileComplete イベントが発生
ファイル転送成功
SendFileComplete イベントが発生

4.6 GetLastError メソッドエラーー覧

GetLastError メソッド実行時に返されるエラーは以下の通りである。

戻り値	説明
0	エラーは発生していません
1	未使用
2	ソケットにコネクトできませんでした
3	未使用
4	未使用
5	メソッドに指定したパラメータに問題があります
6	ファイルがオープンできませんでした
7	MARK30 の共有フォルダにあるファイルがオープンできませんでした
8	クライアント PC のコンピュータ名が取得できませんでした
9	クライアント PC にある共有フォルダにアクセスするためのユーザ名が指定されていません
10	未使用
11	タイムアウト
12	電源に既に接続しました
13	接続しませんので、Disconnect できません
14	LN データ準備中ためデータを読み込めません
15	Heartbeat の acknowledge(応答)がありません。
16	Heartbeat 閉じています。(socket が異常に disconnect された時)
17	マクロデータを取得する時、マクロ変数の番号は、範囲外です。(nStartNo, or nEndNo が <0 or >=200000)
18	共有フォルダにアクセスできません。(機械の"NCPROHD"か、PC の"PCSHARE"共有フォルダかへのアクセスできません。)
19	LN BUSY 中のため、電極補正データ書き込めません。
20	電極補正データ編集中のため、電極補正データ読み取り・書き込みできません。
21	補正がかかっているため、電極補正データ書き込めません。
22	電極補正データ読み取り失敗。
23	電極補正データ書き込み失敗。
24	パターン番号が操作[39] 電極補正有効ﾊﾟﾀｰﾝ数より大きい。

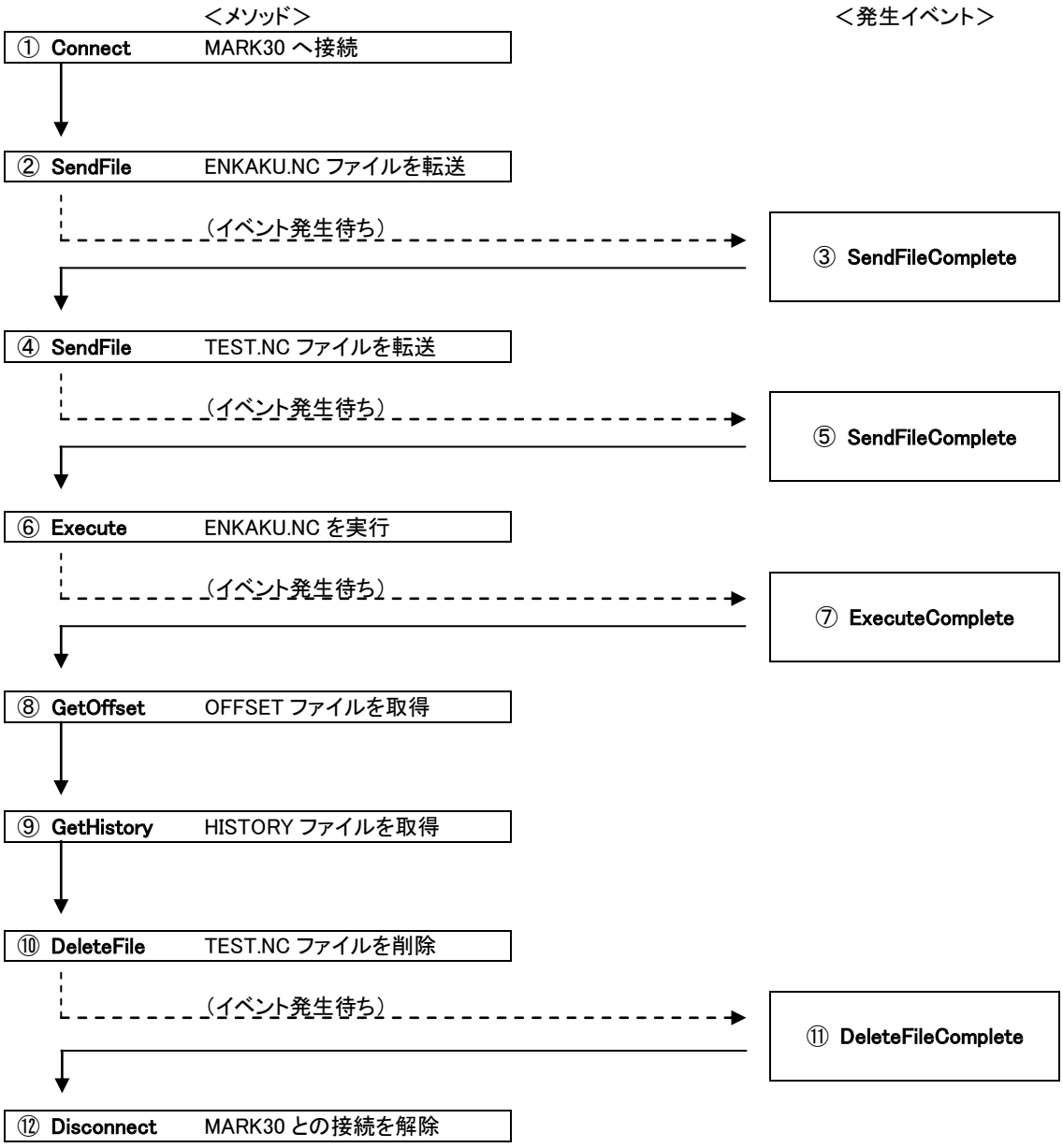
4.7 nCoordSys と座標系の関係テーブル

nCoordSys	0	1	2	3	4	5	6	7	8	9
座標系	54	55	56	57	58	59	154	155	156	157
nCoordSys	10	11	12	13	14	15	16	17	18	19
座標系	158	159	254	255	256	257	258	259	354	355
nCoordSys	20	21	22	23	24	25	26	27	28	29
座標系	356	357	358	359	454	455	456	457	458	459
nCoordSys	30	31	32	33	34	35	36	37	38	39
座標系	554	555	556	557	558	559	654	655	656	657
nCoordSys	40	41	42	43	44	45	46	47	48	49
座標系	658	659	754	755	756	757	758	759	854	855
nCoordSys	50	51	52	53	54	55	56	57	58	59
座標系	856	857	858	859	954	955	956	957	958	959

5　シーケンスの一例

以下に MK30COMM.OCX を使用した、シーケンスの一例を示します。

以下の例は、TEST.NC をクライアント PC から MARK30 へ転送、実行、削除を実現する例です。



※　②の前に、ENKAKU.NC を作成する必要があります。ファイルの内容は、④で転送するファイル（実体ファイル）を Q コマンド形式で記述します。（例） QTEST(0.000,10.000);

6 特記事項

- GetOffset メソッドは、NC 実行中に実行しないで下さい。
- DeleteFile メソッドは、NC 実行中に実行しないで下さい。また、このメソッドで失敗するような場合は、処理の直前で WAIT 処理（2 秒以上）を加えてみてください

以上